კობა გელაშვილი

ალექსანდრე გამყრელიძე

დაპროგრამების საფუძვლები (C++)

(სალექციო კურსი)

თავი 1: შესავალი

- C++-ის ისტორია
- რა არის პროგრამა?
- კომპიუტერში მთელი რიცხვების წარმოდგენა
- C++-ის ერთი ნაწილი, რაც თანაკვეთაშია მათემატიკასთან
- როგორი უნდა იყოს იდეალური პროგრამა

C++-ის ისტორია

პროგრამირების ენა C შეიქმნა 1972 წელს პროგრამისტ დენის რიჩის (Dennis Ritchie) მიერ. თუმცა, თუ მოვინდომებთ პროგრამაში მიმდინარე დროის დაბეჭდვას, C დაბეჭდავს 1970 წლის პირველი იანვრიდან გასულ დროს წამებში, რაც მიჩნეულია C- ის ერის დასაწყისად. ახალ ენას ასეთი სახელი ავტორმა უწოდა იმის გამო, რომ ენას რომელსაც იგი მანამდე იყენებდა, ერქვა B.

1979 წელს ბიარნ სტრაუსტრუპმა (Bjarne Stroustrup) დაიწყო ამ ენის გაუმჯობესებული ვერსიის შექმნა (Bell Labs-ში). ახალი ენის თავდაპირველი სახელწოდება იყო „C კლასებით“, 1983 -სი მას C++ ეწოდა.

რა არის პროგრამა?

პროგრამა შედგება ორი მთავარი ნაწილისგან: მონაცემებისა და ინსტრუქციებისგან.

უმცირესი მონაცემი, რომლის აღქმაც შეუძლია კომპიუტერს, არის 0 ან 1, ან უფრო ზუსტად "-" ან "+", რისი ჩაწერა და წაკითხვაც მას შეუძლია ფიზიკური მეხსიერების უმცირეს ერთეულში. ეს იმის გამო, რომ ჯერ-ჯერობით ელექტრონულ სქემებს მხოლოდ ორ მდგრად მდგომარეობაში შეუძლიათ ყოფნა. მონაცემთა ამ უმცირეს ერთეულს ეწოდება ბიტი (ინგლისური ტერმინი bit წარმოადგენს ”binary digit” -ის ანუ ორობითი ციფრის შემოკლებას). ნებისმიერი მონაცემი, რომელიც მუშავდება კომპიუტერის მიერ, წარმოიდგინება ნულებისა და ერთების კომბინაციით და მის დასამახსოვრებლად საჭიროა ბიტების გარკვეული რაოდენობა. შემდეგ, შედარებით მარტივი ტიპის მონაცემებისგან შესაძლებელია უფრო რთული კონსტრუქციების შექმნა, მათგან კიდევ უფრო რთულის და დახვეწილის და ა.შ.

C++-ში ჩაშენებულია რამდენიმე მარტივი საბაზო ტიპი მონაცემებისთვის. თითოეული ტიპისთვის გამოყოფილია ბიტების მკაცრად დადგენილი რაოდენობა. ამათგან აიგება ყველა დანარჩენი. ზოგადად, მონაცემები და ინსტრუქციები ანალოგიურია მათემატიკის სიმრავლეებისა და ფუნქციებისა. ამიტომ, ვიდრე C++ -ის მონაცემთა ტიპებსა და ფუნქციებზე ვისაუბრებთ, ჯერ უნდა გავარკვიოთ რა ცოდნა შეგვიძლია გამოვიყენოთ მათემატიკიდან. მანამდე მოკლედ შევეხოთ კომპიუტერში რიცხვების წარმოდგენის საკითხს.

კომპიუტერში მთელი რიცხვების წარმოდგენა

სიმარტივისთვის განვიხილოთ მხოლოდ მთელი რიცხვების წარმოდგენის საკითხი. იმისათვის, რომ კომპიუტერმა მთელ რიცხვებზე არითმეტიკული ოპერაციები განახორციელოს, მათ ტოლი რაოდენობის ბიტები უნდა ჰქონდეს გამოყოფილი. მათემატიკაში არის ერთი სიმრავლე მთელი რიცხვებისა, ხოლო C++ -ში არსებობს რამდენიმე ასეთი სიმრავლე, თითოეული სხვადასხვა დიაპაზონს მოიცავს, კონკრეტულ ამოცანაში შეგვიძლია ისე შევარჩიოთ გამოყენებული მთელი რიცხვები, რომ მათთვის რაც შეიძლება ნაკლები მეხსიერების (მაშასადამე პროგრამული დროის) დათმობა გახდეს საჭირო. ყველაზე მცირე დიაპაზონის მთელი რიცხვები არის 0 და 1. თუ ამოცანაში ასეთი რიცხვებია საჭირო, მაშინ პროგრამაში ვიყენებთ ეგრეთ წოდებულ ბულის ტიპის ცვლადებს (**`bool`**), შემდეგ არის ძალიან მოკლე მთელი რიცხვების სიმრავლე (**`char`**), თითოეული ცვლადი ამ სიმრავლიდან იკავებს 8 ბიტს, ანუ ერთ ბაიტს. **`char`** ტიპი (სიმრავლე)

ორობითი	ათობითი
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	−1
1110	−2
1101	−3
1100	−4
1011	−5
1010	−6
1001	−7
1000	−8

კონკრეტული მიზნით შეიქმნა, **ASCII** კოდების შესანახად. ამჟამად ეს სისტემა კვლავ ინტენსიურად გამოიყენება, თუმცა უკვე მოძველებულად ითვლება. შემდეგი არის მოკლე მთელი რიცხვების სიმრავლე (**short**), თითოეული ცვლადი ამ სიმრავლიდან იკავებს 16 ბიტს, ანუ 2 ბაიტს, შემდეგ

int, long, long long.

ზოგადად, ვთქვათ გამოყოფილია N ბიტი. მათ შორის ერთი (უფროსი) ბიტი გვიჩვენებს რიცხვის ნიშანს: თუ ნიშანთვისების ბიტში დგას 0 – რიცხვი დადებითია ან ნულია, თუ კი ეს ბიტი შეიცავს 1 –ს – რიცხვი უარყოფითია (უარყოფითი რიცხვებისთვის, ნიშანთვისების ბიტი მოქმედებს აგრეთვე რიცხვის სიდიდეზეც). არაუარყოფითი რიცხვები წარმოიდგინება, ჩვეულებრივ, ორობითი ჩანაწერით დიაპაზონში 0-იდან $2^{N-1}-1$ -მდე. უარყოფითი რიცხვის ორობით ჩანაწერს მივიღებთ თუ 2^N -ს დავაკლებთ ამ უარყოფითის აბსოლუტურ მნიშვნელობას. ამ სისტემას ეწოდება two's complement system.

ახლა ეს სისტემა ყველაზე გავრცელებულია. მართალია მთავარი თანრიგი განსაზღვრავს ნიშანს, მაგრამ ეს არ ნიშნავს რომ მთელი რიცხვი და მისი მოპირდაპირე რიცხვი მხოლოდ ამ თანრიგით განსხვავდებიან. მაგალითად, შეგვიძლია ვნახოთ ყველა 4 ბიტიანი მთელი რიცხვი ზედა ცხრილში ან რამდენიმე 8 ბიტიანი მთელი რიცხვი შემდეგ ცხრილში:

მთავარი ბიტი									
0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**=**	**127**
0	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**=**	**126**
0	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**=**	**2**
0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**=**	**1**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**=**	**0**
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**=**	**−1**
1	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**=**	**−2**
1	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**=**	**−127**
1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**=**	**−128**

8-ბიტიანი მთელები

უფრო კონკრეტულად, შეგიძლიათ იხილოთ http://en.wikipedia.org/wiki/Two's_complement.

C++-ის ერთი ნაწილი, რაც თანაკვეთაშია მათემატიკასთან

ნებისმიერ ენას ბევრი საერთო აქვს მათემატიკასთან, რადგან გარკვეული აზრით მათემატიკა თვითონ არის ენა. ეს მსგავსება განსაკუთრებით საგრძნობი გახდა C -ის შექმნის (და მის საფუძველზე ახალი ენების განვითარების) შემდეგ, რადგან ამ ენაში ყველა პროგრამა ფუნქციების კრებულია, ხოლო მონაცემები წარმოადგენენ სპეციფიკურ სიმრავლეებს, რომლებიც განუყოფელია მათზე განსაზღვრული ოპერაციებისგან. მართალია, მათემატიკური და C++ -ში გამოყენებული აღნიშვნები საგრძნობლად განსხვავდება, მაგრამ თუ მათემატიკის ელემენტების მცოდნე ადამიანი ერთხელ გაერკვევა განსხვავებებში და მსგავსებებში, მაშინვე აღმოაჩენს, რომ გარკვეული საბაზო ცოდნა უკვე გააჩნია C++ -ში.

სიმრავლეები. ჯერ კიდევ სასკოლო პროგრამებში ჩვენ ვხვდებით ნატურალური რიცხვების N , მთელი რიცხვების Z , რაციონალური რიცხვების Q და ნამდვილი რიცხვების R სიმრავლეებს. C++ ენაში "ჩაშენებულია" ორი მათგანი: მთელი რიცხვების და ნამდვილი რიცხვების (ვუწოდებთ ნამდვილ რიცხვებს, თუმცა რეალურად მისი ელემენტები არის სასრული ათწილადები) სიმრავლეები. თითოეული მათგანის რამდენიმე ვერსიაა შემოთავაზებული, რაც საშუალებას აძლევს პროგრამისტს მაქსიმალურად დაზოგოს კომპიუტერული მეხსიერება და დრო. თითოეულ რეალიზაციას აქვს თავისი დიაპაზონი, თანრიგების რაოდენობა (რასაც სისტემა გამოყოფს ასეთი რიცხვების ჩაწერისთვის) და მათემატიკური ოპერატორები, რომლებიც ხელახლა გადაიტვირთება თითოეული რეალიზაციისთვის. მათემატიკური ოპერატორების გადატვირთვა ნიშანვს, რომ ერთი და იგივე ოპერატორი სხავასხვა სიმრავლეზე აღინიშნება ერთნაირად, მაგრამ მოქმედებს განსხვავებული წესით. ამის გაკეთება აუცილებელია, რადგან, შესაძლოა ორი მთელი რიცხვის შეკრების შედეგი დამოკიდებული იყოს იმაზე, თუ როგორაა რეალიზებული ისინი (ანუ რომელი სიმრავლის ელემენტებად განვიხილავთ მათ). მაგალითად, თუ a არის **char** ტიპის ცვლადი (ერთბაიტიანი წარმოდგენით) რომლის მნიშვნელობაა 127, მაშინ შეკრების ოპერატორი, გადატვირთული ამ სიმრავლეზე, a+a -ის შედეგად მოგვცემს -2-ს. ხოლო a++ -ის შედეგად -128-ს. თუ a იქნებოდა **int** ტიპის (ან სხვა რაიმე ტიპის, რომელიც ერთ ბაიტზე მეტს გამოყოფს რიცხვის შესანახად), მაშინ შეკრების შედეგები იქნებოდა ისეთი, რასაც მიჩვეული ვართ. სხვა მაგალითი არის გაყოფის ოპერაცია "/". თუ ორ მთელ რიცხვს გავყოფთ ერთმანეთზე, გაყოფის შედეგიც მთელია (წილადი ნაწილი იგნორირდება), თუ ნამდვილ რიცხვებს გავყოფთ ერთმანეთზე იგივე (ოღონდ სხვა ტიპზე გადატვირთული ოპერატორის) საშუალებით, მაშინ შედეგი იქნება ისეთი, როგორსაც მიჩვეული ვართ.

როგორც ვხედავთ, C++ -ში მოცემული ორი მთელი რიცხვისთვის ერთი და იგივე არითმეტიკული ოპერატორების მოქმედების შედეგი შესაძლოა განსხვავდებოდეს ერთმანეთისაგან, რადგან მთელ რიცხვთა აბსტრაქტულ სიმრავლეს შეესაბამება რამდენიმე რეალიზაცია. იგივე მიზეზის გამო მათ ერთი და იგივე სახელი ვერ ერქმევათ (იგივე შეგვიძლია ვთქვათ ნამდვილ რიცხვებზეც). ამ ენაში სიმრავლის და საერთოდ ნებისმიერი რამის სახელი ისეა შერჩეული, რომ მაქსიმალურად შეესაბამებოდეს შინაარსს (სახელსა და შინაარსს შორის შინაგანი კავშირის შენარჩუნება, სხვათა შორის პროგრამირების კარგი სტილის ერთ-ერთი რეკომენდაციაა): **bool**, **char**, **short**, **int**, **long**, **long long**. იგივეა ნამდვილი რიცხვებისთვის, რომლის სხვადასხვა რეალიზაციებიდან შეგვიძლია აღვნიშნოთ **float**, **double**, **long double**.

მათემატიკაში, ბევრი გავრცელებული სიმრავლე განიმარტება სხვა, უფრო მარტივი სიმრავლეების საფუძველზე. ანალოგიური ვითარებაა C++ -ში, სადაც არსებობს უკვე განმარტებული ტიპებიდან ახალი ტიპების შექმნის მექანიზმი, მაგალითად კლასების გამოყენებით.

სიმრავლის ელემენტები. უახლოეს რამდენიმე მაგალითში ვიგულისხმოთ, რომ მთელ რიცხვთა სიმრავლის წარმოდგენისთვის ვსარგებლობთ 32 ბიტით, ანუ ვიყენებთ **int** ტიპს, ნამდვილი რიცხვების წარმოდგენისთვის ვიყენებთ **float** ტიპს. კიდევ ერთხელ დავაზუსტოთ, რომ ტიპი არის სიმრავლე და მასზე განსაზღვრული ოპერატორები (არა მხოლოდ არითმეტიკული, მაგალითად, სხვადასხვა ტიპზე ძალიან გავრცელებულია მინიჭების "=" და შედარების '==" ოპერატორები) . განვიხილოთ კარგად ცნობილი მათემატიკური ჩანაწერი: $x \in R$ და $j \in Z$. C++ -ში მიკუთვნებისთვის არ ხდება სპეციალური კვანტორის გამოყენება და ვწერთ უბრალოდ:

```
int j;
float x;                                                    (1)
```

რომელია უფრო მოხერხებული? ალბათ ბოლოს შემოღებული, რადგან მისი განზოგადება უფრო ადვილია. მაგალითად, თუ გვინდა C++ -ში ჩავწეროთ ფაქტი "ნამდვილი x ცვლადის მნიშვნელობა არის 12.95-ის ტოლი", (1) სტრიქონს შეცვლით სულ ოდნავ:

```
float x = 12.95;          // C ენაში მიღებული  ინიციალიზაციის ფორმა      (2)
```

ან

```
float x(12.95);           // C++ ენში მიღებული  ინიციალიზაციის ფორმა
```

თუმცა შეგვიძლია მათემატიკური ჩანაწერის ორი ($x \in R,\ x = 12.95$) წინადადების ანალოგიურად გამოვიყენოთ ორი შეტყობინება (`statement`):

```
float x;
x = 12.95;
```

პროგრამის ფრაგმენტებთან დაკავშირებით ხშირად გამოიყენება ტერმინი შეტყობინება და არა წინადადება, რადგან პროგრამული კოდი განკუთვნილია კომპილერისთვის, (2) შეტყობინების შედეგს წარმოადგენს ის, რომ მეხსიერებაში x ცვლადისთვის დაიჯავშნება მონაკვეთი (იმდენი ბიტი, რამდენიც ზოგადად გამოიყოფა კონკრეტული სისტემის მიერ **float** ტიპის ცვლადისთვის), და ამავე დროს ამ მონაკვეთში ჩაიწერება 12.95. ადამიანისთვის (2) წარმოადგენს ცვლადის აღწერას, ხოლო პროგრამისთვის განაცხადს კონკრეტული ცვლადისთვის საჭირო მეხსიერებაზე.

სავარჯიშო: როგორ ჩავწეროთ, რომ

1. x არის გრძელი ნამდვილი რიცხვი?
2. y არის ნამდვილი რიცხვი მნიშვნელობით 27238.32?
3. k არის მოკლე მთელი რიცხვი მნიშვნელობით 456?
4. k არის ძალიან მოკლე მთელი რიცხვი მნიშვნელობით 45?
5. t არის ბულის ტიპის რიცხვი ჭეშმარიტი მნიშვნელობით ?

ფუნქციები. ვნახოთ როგორ ხდება ფუნქციებთან დაკავშირებული ჩანაწერების კონვერტირება C++ -დან მათემატიკაში და პირიქით. მათემატიკური ჩანაწერი $f(x): R \to Z$ ნიშნავს რომ $f(x) \in Z$ ანუ $\text{int } f(x)$; და $x \in R$ ანუ $\text{float } x$. ამ ორი ფაქტის გაერთიანება გვაძლევს შესაბამის C++ –ის ჩანაწერს: $\text{int } f(\text{float } x)$;

სავარჯიშო: რას ნიშნავს:

6. $function(z): Z \to Z$?
7. $F(y): Z \to R$?
8. $F(x, y): Z^2 \to R$?
9. $\text{int } T(\text{float } a)$?

გარდა მსგავსებისა, მნიშვნელოვანია განსხვავების ცოდნა. მათემატიკაში, თუ ორ ფუნქციას აქვს ტოლი განსაზღვრის და მნიშვნელობათა სიმრავლეები და ისინი ერთი და იმავე

არგუმენტისთვის იღებენ ტოლ მნიშვნელობებს, მაშინ თვითონაც ითვლებიან ტოლად. პროგრამირებაში, გასათვალისწინებელია ალგორითმი, რომლითაც ხდება ამ ფუნქციების მნიშვნელობების გამოთვლა. თუ განსხვავებულია ალგორითმი, ფუნქციებიც განსხვავებულად ითვლებიან. მაგალითად, ფუნქციები $4x$ და $x+x+x+x$, განსხვავდებიან თავისი შესრულების სისწრაფით.

როგორი უნდა იყოს იდეალური პროგრამა?

არსებობს კრიტერიუმები, რომლებსაც უნდა აკმაყოფილებდეს იდეალური პროგრამა. ზოგჯერ მათი ერდროულად შესრულება შეუძლებელია, ზოგჯერ განზრახ ხდება საჭირო რომელიმე მათგანის დარღვევა, მაგრამ ზოგადად ითვლება, რომ იდეალური პროგრამა უნდა იყოს:

- მრავალჯერადი, გამოყენების თვალსაზრისით;
- ადვილად გაუმჯობესებადი გადაკეთების თვალსაზრისით და მარტივი ექსპლოატაციაში;
- საიმედოდ დაწერილი (მაგალითად, ნაკლებად დამოკიდებული კონკრეტულ სისტემაზე);
- ადვილად გასარჩევი;
- კარგად დოკუმენტირებული (ანუ ახლდეს ყველა საჭირო კომენტარი და ახსნა-განმარტება).

იდეალური პროგრამების შესაქმნელად აუცილებელია იმ გამოცდილების გამოყენება, რაც დაგროვდა საუკეთესო პროგრამისტების მიერ და რაც კონდენსირებულია სტილების და პარადიგმების სახით. ჩვენს კურსში ყურადღებას გავამახვილებთ რამდენიმე მომენტზე, რასაც პროგრამირების კარგი სტილი გვირჩევს: კომენტარების გამოყენებაზე და პროგრამის ფრაგმენტების შეწევაზე.

C++ –ში კოდის მრავალჯერადი განმეორების იდეა ეფუძნება ფუნქციების და კლასების გამოყენებას. სტანდარტული ბიბლიოთეკის სახით ენა თავაზობს პროგრამისტებს უამრავ ფუნქციას და კლასს. კერძოდ, C++ –ის ყველა პროგრამა იყენებს სტანდარტულ შეტანა–გამოტანის კლასებს.

უმარტივესი C++ –პროგრამის სტრუქტურა შემდეგია:

```
#include <iostream>    // მიმართვა შეტანა–გამოტანის
using namespace std;   // სტანდარტულ კლასებზე

// პროგრამის მთავარი ფუნქცია
int main()
{
      // მონაცემებზე განაცხადი
      // და შესრულებადი ინსტრუქციები
}
```

მაგალითად, ქვემოთ მოყვანილი პროგრამა იპოვის და დაბეჭდავს ორი მთელი რიცხვის ჯამს:

```
#include <iostream>
using namespace std;

int main()
{
   int number1(9), number2(-4), sum;
   sum = number1 + number2;
   cout<<"Sum = "<<sum<<endl;
   return 0;
}
```

თავი 2: პროგრამირების კარგი სტილი

- რას წარმოადგენს სტილი
- კომენტარები
- პროგრამული კოდის კომენტირება. ცვლადის სახელი როგორც კომენტარის ფორმა
- კოდის ფორმატირება წანაცვლების საშუალებით
- სიცხადე და სიმარტივე
- როგორი უნდა იყოს იდეალური პროგრამა
- პროგრამები, ამოცანის დასმიდან შესრულებამდე

რას წარმოადგენს სტილი

სტილი დაპროგრამების მნიშვნელოვანი ნაწილია, რადგან მასში კონცენტრირებულია უმდიდრესი გამოცდილება, რაც დააგროვეს საუკეთესო პროგრამისტებმა. ეს ცოდნა გვიცავს უამრავი ძნელად წარმოსადგენი შეცდომისა და დროის უაზრო ხარჯვისგან. ამიტომ ვიწყებთ კარგი სტილის გაცნობას თავიდანვე. დაპროგრამების კარგი სტილის გამოყენების გარეშე მარტივი და ადვილად წასაკითხი პროგრამის შექმნა თითქმის წარმოუდგენელია.

გავრცელებული შეხედულების საწინააღმდეგოდ, სინამდვილეში პროგრამისტი დროის უმეტეს ნაწილს ხარჯავს არა პროგრამების დაწერაზე, არამედ არსებული პროგრამების გამართვაზე, ექსპლუატაციაზე და გაუმჯობესებაზე. რაც დრო გადის, ტიპურ გამოყენებით პროგრამებში სტრიქონთა საშუალო რაოდენობა სულ უფრო იზრდება. მაგალითად, 1980–იდან 1990 წლამდე საშუალო ზომის გამოყენებით ამოცანაში სტრიქონების რაოდენობა გაიზარდა 23 000-დან 1 200 000-მდე. შესაბამისად, რთულდება კარგი სტილის დაცვით დაწერილი პროგრამული კოდის გარჩევაც კი. მაგალითად, ერთ-ერთ კონფერენციაზე მენეჯერთა 74%-მა განაცხადა, რომ მათ უხდებათ ისეთ სისტემებთან მუშაობა, რომელთა ექსპლუატაცია მხოლოდ კონკრეტულ ადამიანებს შეუძლიათ, რადგან მათ გარდა ვერავინ ვერაფერს არკვევს. პროგრამული პროდუქტების (software) უმეტესობა ეფუძნება უკვე არსებულ პროგრამულ პროდუქტებს. ამიტომ არსებითი მნიშვნელობა აქვს იმას, რომ შექმნილი პროგრამა იყოს მაქსიმალურად გასაგები ნებისმიერი მომხმარებლისათვის.

ზოგიერთი პროგრამისტი თვლის, რომ პროგრამის დანიშნულებაა მხოლოდ კომპიუტერის უზრუნველყოფა ინსტრუქციების კომპაქტური კრებულით. ასეთ პროგრამებს აქვს ორი ნაკლი:

- გასამართად ძნელია, რადგან დროის გარკვეული პერიოდის შემდეგ ავტორსაც უჭირს მისი გაგება.
- ძნელია პროგრამის მოდიფიცირება და გაუმჯობესება, რადგან პროგრამისტს სჭირდება საკმაო დრო იმის გასარკვევად, თუ რას აკეთებს პროგრამა.

კომენტარები

იდეალურ შემთხვევაში, პროგრამა ემსახურება ორ მიზანს:

- უზრუნველყოს კომპიუტერი ინსტრუქციების კრებულით;
- უზრუნველყოს პროგრამისტი ნათელი, გასაგები ენით დაწერილი აღწერებით იმის შესახებ, თუ რას აკეთებს პროგრამა.

საბედნიეროდ, ორივე ამ მიზნის მიღწევა შესაძლებელია ერთდროულად. ამის საშუალებას იძლევა **კომენტარები**, რომლებიც უკეთდება ინსტრუქციებს აუცილებლობის

შემთხვევაში. კომენტარი არ კომპილდება, ამიტომ გავლენას არ ახდენს პროგრამის შესრულებაზე, იგი საჭიროა პროგრამისტებისთვის და არა კომპიუტერისთვის.

პროგრამა აუცილებლად უნდა შეიცავდეს კომენტარებს. დანერგილი პროგრამა, რომელიც არ შეიცავს კომენტარებს, შენელებული მოქმედების ნაღმს წააგავს, რომელიც თავის წამს ელის. ადრე თუ გვიან ვიღაც აღმოაჩენს შეცდომას ამ პროგრამაში, ან კიდევ ვინმე შეეცდება მის გადაკეთებას და გაუმჯობესებას, კომენტარების არარსებობა კი მის სამუშაოს ძალზე გაართულებს.

C++ -ის სტანდარტულ ვერსიაში (ANSI/**ISO C++ Standard**) შესაძლებელია ორი სახის კომენრტარის გამოყენება:

- C ენაში მიღებული კომენტარები, რომლებიც მოთავსებულია /* და */-ს შორის;
- ე.წ. ორმაგსლეშიანი კომენტარი, რომელიც ერთსტრიქონიანია და ვრცელდება ორმაგი სლეშის (//) მარჯვნივ.

C++ -ის ზოგიერთ გაფართოებაში, მაგალითად C++/CLI-ში, რომელსაც იყენებს Visual Studio, შესაძლებელია ე.წ. სამსლეშიანი კომენტარების გამოყენება, რაც უკავშირდება ინტრგრირებული XML დოკუმენტაციის გამოყენებას და მდგომარეობს შემდეგში: პროგრამული პროდუქტის შემქმნელი ვალდებულია შექმნას კარგად კომენტირემული პროგრამული კოდი, და შექმნას აგრეთვე დოკუმენტაცია, სადაც დაწვრილებით იქნება აღწერილი ამ პროდუქტთან დაკავშირებული ყველა ასპექტი. დოკუმენტაციის მომზადება ითვლება ყველაზე მომაბეზრებელ და დამღლელ ეტაპად პროგრამული პროდუქტის მომზადების პროცესში. სამმაგსლეშიანი კომენტარები საშუალებას იძლევა, რომ კომპილირების პროცესში კომენტარებისგან შეიქმნას პროდუქტის დოკუმენტაცია XML-ის სხვადასხვა ფორმატში. ამ ლექციაში ამ საკითხს არ შევეხებით მეტად, რადგან ეს C++ -ის სტანდარტულ ვერსიას ნაკლებად ეხება.

იმისთვის, რომ შევქმნათ პროგრამა, ნათლად უნდა წარმოვიდგინოთ, თუ რის გაკეთება გვინდა და ჩავწეროთ მარტივად და გასაგებად. შემდეგ ეს ყველაფერი შეიძლება კიდევ ერთხელ შემოწმდეს და "გადაითარგმნოს" კომპიუტერულ პროგრამად, ხოლო ჩვენი ჩანაწერები გამოვიყენოთ კომენტარებად. რადგან პროგრამა სხვადასხვა ნაწილისგან შედგება, კომენტარებიც სხვადასხვანაირია.

როგორც წესი, პროგრამას ყოველთვის უკეთდება საწყისი კომენტარების ბლოკი, რომელიც რამდენიმე პუნქტისგან შედგება. ზოგ შემთხვევაში ყველა მათგანის მოყვანა არც არის აუცილებელი. ეს პუნქტებია:

- **სათაური.** პირველი კომენტარი უნდა შეიცავდეს პროგრამის დასახელებას. აქვე შეგიძლიათ ჩაურთოთ მოკლე აღწერა იმისა, თუ რას აკეთებს პროგრამა. თქვენ შეიძლება გქონდეთ ყველაზე საუკეთესო პროგრამა, რომელიც მსოფლიო მნიშვნელობის ამოცანებს წყვეტს, მაგრამ პროგრამა გამოუსადეგარი იქნება, თუ არავის ეცოდინება, რას აკეთებს იგი.
- **ავტორი.** მონაცემები თქვენს შესახებ. თუ ვინმე დააპირებს თქვენი პროგრამის შეცვლას, შესაძლებლობა ექნება ინფორმაციისათვის ან დახმარებისათვის თქვენ მოგმართოთ.
- **მიზანი.** რისთვის დაწერეთ ეს პროგრამა?
- **გამოყენება.** ამ განყოფილებაში მოკლედ აღწერეთ, როგორ უნდა მართონ პროგრამა. იდეალურ შემთხვევაში ყველა პროგრამას უნდა ახლდეს დოკუმენტთა კრებული, რომელშიც აღწერილი იქნება, თუ როგორ გამოვიყენოთ იგი.
- **საცნობარო ინფორმაცია**. არსებული პროგრამების სხვადასხვა ფრაგმენტის თქვენს მიერ გამოყენება (კოპირება) წარმოადგენს დაპროგრამების გავრცელებულ ფორმას, და სრულიად კანონიერ ფორმასაც, თუ თქვენ არ არღვევთ ამ დროს საავტორო უფლებებს. ამ პუნქტში უნდა მიუთითოთ ავტორი ნებისმიერი ნაშრომისა, რომელითაც თქვენ ისარგებლეთ (რომლის ფრაგმენტების კოპირებაც მოახდინეთ).

- **ფაილების ფორმატი**. ჩამოთვალეთ ის ფაილები, რომლებსაც თქვენი პროგრამა კითხულობს ან ქმნის და აღწერეთ მოკლედ მათი ფორმატი.
- **შეზღუდვები**. ჩამოთვალეთ ნებისმიერი შეზღუდვები ან დასაშვები საზღვრები, მაგ. ასეთი ტიპისა "პროგრამა არ ამოწმებს შეცდომებს მონაცემებში".
- **ცვლილებების ქრონოლოგია**. ეს განყოფილება შეიცავს სიას, სადაც ნაჩვენებია ვინ შეცვალა პროგრამა, როდის და რა ცვლილებები იქნა მასში შეტანილი.
- **შეცდომების გასწორება**. თუ პროგრამა აღმოაჩენს შეცდომას, აღწერეთ რას მოიმოქმედებს ამ დროს იგი. მაგალითად, დაუყოვნებლივ შეწყვეტს მუშაობას, თუ მიგვითითებს შეცდომის შესწორების გზას.
- **შენიშვნები**. აქ შეიტანეთ სპეციალური კომენტარები ან სხვა ინფორმაცია, რომელიც ზემოთ არ გქონდათ მოყვანილი.

კომენტარებში ახსნილი უნდა იყოს ყველაფერი, რაც პროგრამისტმა უნდა იცოდეს პროგრამის შესახებ, მაგრამ ზედმეტი არაფერი. დარწმუნებული უნდა იყოთ, რომ ყველაფერი, რაც შეგაქვთ კომენტარებში, საჭიროა.

ტექსტურ რედაქტორებში (მაგალითად, Word) ჩვეულებრივი ტექსტის ბეჭდვის დროს ჩვენ შეგვიძლია შევცვალოთ შრიფტის ზომა, **გავამუქოთ,** *დავხაროთ,* რათა ტექსტის სხვადასხვა ნაწილი გამოვყოთ. პროგრამის ტექსტის, ანუ კოდის შემთხვევაში კი უნდა შემოვიფარგლოთ მხოლოდ ფიქსირებული ზომის შრიფტით. თუმცა, მაინც შეგვიძლია გარკვეული ეფექტების შექმნა კომენტარების წერის დროს. მაგალითად:

```
//////////////////////////////////////////////////////
//              გაფრთხილება: ეს არის გამაფრთხილებელი          //
//                        გზავნილის მაგალითი, რომელიც          //
//                     პროგრამისტის ყურადღებას მიიქცევს        //
//////////////////////////////////////////////////////
```

ან:

```
//>>>>>>>>>>>>>>>>       სექციის სათაური      <<<<<<<<<<<<<<<<<< //
```

ან უბრალოდ

```
int counter;          // მთვლელი
```

ნებისმიერ პროგრამაში, ფაქტიური კოდი (ანუ კოდი კომენტარების გარეშე) შედგება ორი ნაწილისაგან: ცვლადები და შესრულებადი ინსტრუქციები. ცვლადები გამოიყენება იმისთვის, რომ შევინახოთ პროგრამის მიერ გამოყენებული მონაცემები, ხოლო ინსტრუქციები მიუთითებს კომპიუტერს, თუ რა მოქმედებები უნდა ჩაატაროს მონაცემებზე.

პროგრამული კოდის კომენტირება. ცვლადის სახელი როგორც კომენტარის ფორმა

სერიოზულ ამოცანებში, როგორც წესი, პროგრამული კოდის წერას იწყებენ ამოცანის საფუძვლიანი გააზრების შემდეგ, როდესაც უკვე მოფიქრებულია საკმაოდ დეტალური გეგმა. გეგმის პირველ, ყველაზე უხეშ ვარიანტს ჯერ ჩვეულებრივ სალაპარაკო ენაზე წერენ, ხოლო რაც უფრო იხვეწება იგი, მით უფრო ხშირად გამოერევა მასში მათემატიკური აღნიშვნები და პროგრამირების ენისთვის ჩვეული კონსტრუქციები და შეტყობინებები. ასეთ სინთეზს (სალაპარაკო ენისა პროგრამირების ენასთან) ფსევდოკოდს უწოდებენ. როცა ფსევდოკოდი იმდენად დეტალური გახდება, რომ მხოლოდ მისი "გადათარგმნა" არის საჭირო პროგრამირების ენაზე, მაშინ იწყება პროგრამული კოდის წერა. მაგრამ, როგორც ზემოთ აღვნიშნეთ, ადრე თუ გვიან საჭირო ხდება პროგრამის გადაკეთება და გაუმჯობესება, ან კიდევ

სადმე სხვაგან დანერგვა. ისევ საჭირო რომ არ შეიქმნას მთელი სამუშაოს თავიდან გამეორება, ამ მომენტისთვის აუცილებელია იმ გეგმის და ფსევდოკოდის შენარჩუნება, რომლის საფუძველზეც შეიქმნა პროგრამა. როგორც წესი, ფსევდოკოდი ილექება კომენტარებში, კომპილერი მას არ აღიქვამს, მაგრამ კვალიფიციურ პროგრამისტს კომენტარებიდან შეუძლია აღადგინოს ყველა ის სირთულე და პრობლემა, რაც გადაიჭრა პროგრამის წერის პროცესში.

მაქსიმალური სიცხადის მიღწევის მიზნით, ძალიან გავრცელებულია ცვლადების სათაურებში მათი შინაარსის ასახვა - კარგად შერჩეული სახელი გარკვეული აზრით უნდა ითავსებდეს კომენტარის ფუნქციას. განვიხილოთ რამდენიმე მაგალითი.

პროგრამირებაში, **ცვლადი** არის ადგილი კომპიუტერის მეხსიერებაში, გამოყოფილი რაიმე სიდიდის (მნიშვნელობის) შესანახად. ამ ადგილს C++ აიგივებს ცვლადის სახელთან. სახელი შეიძლება იყოს ნებისმიერი სიგრძის და ისე უნდა შეირჩეს, რომ მისი შინაარსი გასაგები იყოს (სინამდვილეში სიგრძის ზღვარი არსებობს, მაგრამ იგი დიდია და რეალურად ამ შეზღუდვას ვერ ვგრძნობთ). საჭიროა ყველა ცვლადი, რომელთაც პროგრამაში ვიყენებთ, წინასწარ ჩამოვთვალოთ ანუ აღვწეროთ. ეს ჩამონათვალი კომპილერისთვის წარმოადგენს განაცხადს მეხსიერების გამოყოფაზე. ცვლადებზე განაცხადის საკითხი დაწვრილებით იქნება განხილული ერთ-ერთ შემდეგ ლექციაში. შემდეგი განაცხადი, რომლის მათემატიკური ჩანაწერია $p,q,r \in Z$, C++-ის აცნობებს, რომ ჩვენ ვაპირებთ სამი `p`, `q` და `r` მთელი (**`int`**) რიცხვის გამოყენებას:

```
int p,q,r;
```

მაგრამ სრულიად გაუგებარია, რისთვის? ეს სამი ცვლადი ნებისმიერ რამეს შეიძლება აღნიშნავდეს. შემოკლებულ ჩანაწერებს თავი უნდა ავარიდოთ. ზედმეტი შემოკლება ძნელად გასაგებს ხდის აღნიშვნებს. ამისგან განსხვავებით, შემდეგი განაცხადი:

```
int account_number;
int balance_owed;
```

ცვლადების სახელების საშუალებით აშკარად მიგვანიშნებს, რომ საქმე გვაქვს საბუღალტრო აღრიცხვის პროგრამასთან. მაგრამ ჩვენ უფრო მეტი ინფორმაციის მიღებაც შეგვიძლია თუ კომენტარებსაც გამოვიყენებთ. მაგალითად:

```
int  account_number;  // საბანკო ანგარიშის ნომერი
int  balance_owed;    // დავალიანება (ლარებში)
```

ვწერთ რა კომენტარებს ყოველი განაცხადის შემდეგ, ჩვენ სინამდვილეში ვქმნით მინი ლექსიკონს, სადაც განვსაზღვრავთ ყოველი ცვლადის სახელის მნიშვნელობას. ძალზე მნიშვნელოვანია პროგრამაში გამოყენებული ერთეულების მითითება (კმ, სთ, $ და ა.შ.), განსაკუთრებით მაშინ, თუ გვიწევს პროგრამის გადაკეთება, ან მისი ფორმატის შეცვლა.

ზოგადად, პროგრამისტმა ის ფაქტორები უნდა გაითვალისწინოს, რაც კარგ პროგრამას ახასიათებს. მაგალითად, იდეალურ შემთხვევაში პროგრამა უნდა იყოს ადვილად აღსაქმელი, კომპაქტური, სწრაფად უნდა სრულდებოდეს, უნდა საჭიროებდეს მინიმალურ მეხსიერებას და ა. შ. სამწუხაროდ, რეალურ სიტუაციებში ყველა მიზანი ერთდროულად ვერ მიიღწევა და უნდა ვიცოდეთ, თუ რა არის ძირითადი. ნებისმიერ შემთხვევაში, პროგრამა უნდა იყოს რაც შეიძლება ადვილად აღსაქმელი, თუნდაც ამისთვის, გონიერების ფარგლებში, სხვა მიზნების მიღწევის გზაზე ნაწილობრივი კომპრომისების გაკეთება გახდეს საჭირო. პროგრამები თავისი ბუნებით ძალიან რთულია. ყველაფერი, რასაც გააკეთებთ ამ სირთულის შესამცირებლად, გააუმჯობესებს თქვენს პროგრამას. უნდა გვახსოვდეს, რომ რთულ პროგრამას შესაძლოა გაუთვალისწინებელი ეფექტები ახლდეს (ზოგიერთ ჩრდილოვან ეფექტებს მოგვიანებით განვიხილავთ).

კომპიუტერისთვის სულერთია, თუ პროგრამის როგორ ვერსიას შექმნის პროგრამისტი: ადვილად გასარჩევს თუ რთულს და ჩახლართულს. კარგი კომპილერი ორივე ვერსიას

შეუსაბამებს მანქანურ კოდს. გასაგებად დაწერილი კოდით მხოლოდ პროგრამისტი ისარგებლებს.

კოდის ფორმატირება წანაცვლების საშუალებით

იმისათვის, რომ პროგრამა ადვილად გასაგები იყოს, პროგრამისტების უმრავლესობა წანაცვლებულად წერს პროგრამის გარკვეულ ნაწილებს. C++ -ის პროგრამების გაფორმების ზოგადი წესია წავანაცვლოთ ერთი დონით ყოველი ახალი შიგა ბლოკი. ერთ დონეზე შეიძლება მოთავსდეს მხოლოდ ერთმანეთისაგან დამოუკიდებელი ბლოკები. ძალიან მოკლედ, ბლოკს ვუწოდებთ პროგრამული კოდის ფრაგმენტს, რომელიც მოთავსებულია ფიგურულ ფრჩხილებში. განვიხილოთ მაგალითი, რომელშიც სამი განსხვავებული დონე შეგვხვდება.

```
//////////////////////////////////////////////////////////
//  ავტორი:
//  პროგრამა: პროგრამა გამოიცნობს  უდრის თუ არა რიცხვი 5-ს
//////////////////////////////////////////////////////////

#include <iostream>
using namespace std;
int main()
{
   int a(5);                //განაცხადი მთელ რიცხვზე და ინიციალიზაცია 5-ით
   if( a == 5 )
   {
      cout<<"Hello,\n";     // ეკრანზე გამოჩნდება გზავნილი Hello,
      cout<<"a = 5\n\n";    // და შემდეგ სტრიქონში გზავნილი a = 5
   }
   else
   {
     cout<<"I'm Program\n";         // ეკრანზე დაიბეჭდება I'm Program და
     cout<<"I know, a isn't 5\n\n";  // შემდეგ სტრიქონში: I know, a isn’t 5
   }
   return 0;
}
```

კომენტარებიდან გასაგებია, თუ რას აკეთებს პროგრამა. ვნახოთ, თუ როგორ აკეთებს ამას.

კომენტარის ბლოკი განმარტავს პროგრამის დანიშნულებას და შეიცავს ცნობას ავტორის შესახებ. შემდეგი სტრიქონები: `#include <iostream>` და `using namespace std;` ნიშნავს მიმართვას სტანდარტული ბიბლიოთეკის შეტანა გამოტანის კლასებზე (ჩვენ პროგრამას სხვა ინფორმაცია არ სჭირდება სტანდარტული ბიბლიოთეკიდან). ბიბლიოთეკების ჩართვა აუცილებელია, რათა გამოვიყენოთ უკვე არსებული პროგრამების ფრაგმენტები. ეს სტრიქონები მოთავსებულია ყველაზე მარცხნივ, პირველ დონეზე, ანუ ყოველგვარი წანაცვლების გარეშე.

მომდევნო სტრიქონში იგივე დონეზე მოთავსებულია მთავარი ფუნქცია `main()`. რადგან იგი ყველა პროგრამაში გვხვდება, ამიტომ მას არ ვუკეთებთ კომენტარს. მის შიგნით მოთავსებული ნაწილი მთლიანად მას ეკუთვნის და ის მეორე დონეა. ანალოგიურად, მეორე დონის ინსტრუქციის **if**-ის შიგნით მოთავსებული ინსტრუქციების შესრულების საკითხი მთლიანად **if** -ის შედეგზეა დამოკიდებული და ამიტომ წანაცვლების მესამე დონეა.

`main()`-ის ტანში პირველი სტრიქონია `a` მთელი რიცხვის აღწერა:

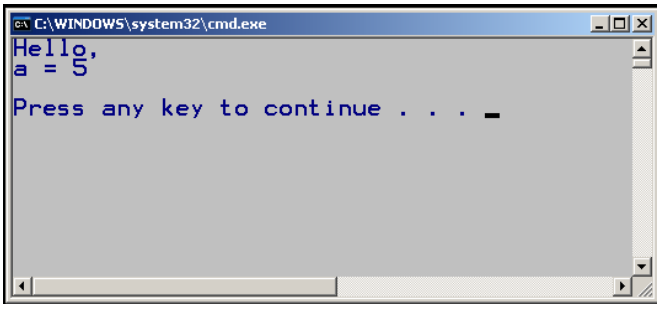
```
    int a(5);
```

რაც აცნობებს კომპილერს, რომ პროგრამაში მონაწილეობს მთელი რიცხვი, სახელით a, რომელიც ეკუთვნის მთელი რიცხვების **int** სიმრავლეს. **int** რეზერვირებული სახელია, ამ ენაში მისი სხვა მიზნით გამოყენება აკრძალულია. a ცვლადზე ფრჩხილებში მიდგმული კონსტრუქცია a(5) მიუთითებს კომპილერს, რომ მან ჩაწეროს მეხსიერებაში მითითებული მნიშვნელობა.

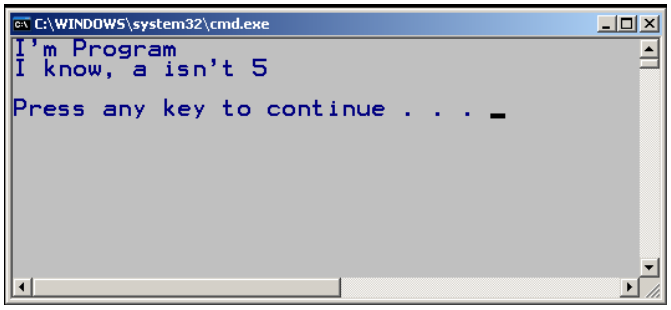
შემდეგ რამდენიმე სტრიქონს იკავებს **განშტოების** შეტყობინება **if**(a == 5) შესაძლო შედეგებითურთ. მისი შესრულების წესი მარტივია: შემოწმდება პირობა a==5 (უუდრის?) და თუ პირობა ჭეშმარიტია (სწორია), შესრულდება პირველ ფიგურულ ფრჩხილებში მოთავსებული ყველა შეტყობინება (ანუ, თუ a ტოლია 5-ის, შესრულდება cout<<"Hello,\n"; და cout<<"a = 5\n\n";), ხოლო თუ a არ უდრის 5-ს (ე. ი. პირობა მცდარია) – მაშინ შესრულდება **else** სიტყვის შემდეგ ფიგურულ ფრჩხილებში მოთავსებული ყველა შეტყობინება.

ყურადღება მივაქციოთ ჩანაწერს a == 5. ეს არის შედარება ტოლობაზე - ტოლია თუ არა? ასეთი შედარება C++-ში ჩაიწერება ზედიზედ ორი ტოლობის ნიშნით. არავითარ შემთხვევაში არ შეიძლება " == " ნიშნის შეცვლა " = "-ით ან " = = "-ით.

პროგრამაში a -ს თავიდან მივანიჭეთ 5, შედარება a == 5 ჭეშმარიტია და პროგრამა გამოიტანს შედეგს:



შევცვალოთ main()-ის მეორე სტრიქონი ასე: int a(9); რაც მიანიჭებს a-ს მნიშვნელობას 9. პირობა a==5 იქნება მცდარი, და პროგრამა გამოიტანს შედეგს



არსებობს წანაცვლების ორი ძირითადი სტილი. პირველი – მოკლე ფორმაა:

```
int main()
{
   int a =9;
   if( a == 5 ) {
      cout<<"Hello,\n";
      cout<<"a = 5\n\n";
   }
   else {
     cout<<"I'm Program\n";
     cout<<"I know, a isn't 5\n\n";
   }
   return 0;
}
```

მეორე სტილი ფიგურულ ფრჩხილებს ცალკე სტრიქონებზე სვამს:

```
int main()
{
   int a =9;
   if( a == 5 )
  {
      cout<<"Hello,\n";
      cout<<"a = 5\n\n";
   }
   else
   {
     cout<<"I'm Program\n";
     cout<<"I know, a isn't 5\n\n";
   }
   return 0;
}
```

ორივე ფორმატი ხშირად გამოიყენება.

წანაცვლებაში, გამოტოვებული სიმბოლოების (space) რაოდენობა პროგრამისტზეა დამოკიდებული. მიღებულია ორი, ოთხი ან რვა space-ით წანაცვლება. გამოკვლევებმა აჩვენეს, რომ ოთხი ინტერვალით წანაცვლების შემთხვევაში პროგრამა უკეთ იკითხება.

ზოგიერთი რედაქტორი, UNIX Emacs editor,Borland C++ და Microsoft Visual C++ internal editors შეიცავს საშუალებებს, რაც ავტომატურად ახდენს თქვენს პროგრამაში კოდის ტექსტის წანაცვლებას.

სიცხადე და სიმარტივე

პროგრამა უნდა იკითხებოდეს, როგორც ტექნიკური დოკუმენტი. ის დაყოფილი უნდა იყოს სექციებად და პარაგრაფებად. პარაგრაფი უნდა დაიწყოთ შესაბამისი კომენტარით და გამოყოთ ეს კომენტარი სხვა პარაგრაფისგან თავისუფალი სტრიქონით. მაგალითად, ვხსნით ამოცანას: სიბრტყეზე მოცემულია ორი წერტილი A და B. A -ს კოორდინატებია x1 და y1, B -სი – x2 და y2. ჩვენ გვინდა A და B წერტილებს მნიშვნელობები გავუცვალოთ.

```
/* არცთუ საუკეთესო პროგრამა */
temp = x1;
x1 = x2;
x2 = temp;
temp = y1;
y1 = y2;
y2 = temp;
```

უკეთესი ვერსია იქნება:

```
/*
* ადგილი გავუცვალოთ (swap) A და B-ს
*/

/* X კოორდინატების გაცვლა */
temp = x1;
x1 = x2;
x2 = temp;

/* Y კოორდინატების გაცვლა */
temp = y1;
y1 = y2;
```

```
y2 = temp;
```

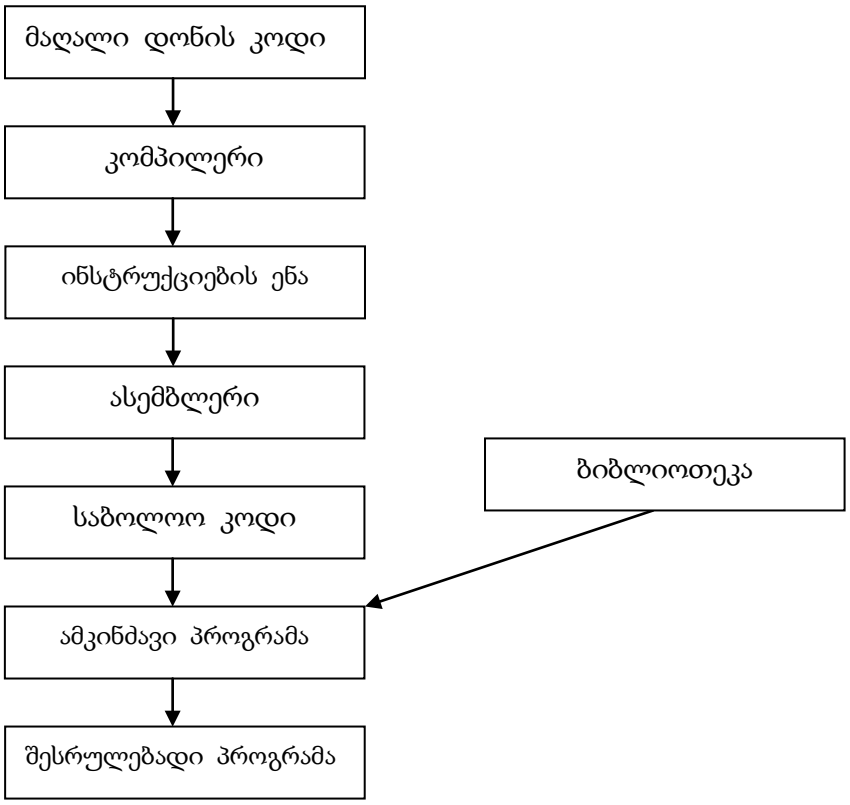
პროგრამა მარტივი უნდა იყოს. ზოგადი წესები ასეთია:

- ეცადეთ, თქვენს პროგრამას არ ჰქონდეს რთული ლოგიკა. დაყავით ცალკეულ პროცედურებად და შეამცირეთ სირთულის ხარისხი.
- გრძელ ინსტრუქციებს თავი აარიდეთ. უმჯობესია გრძელი წინადადება რამდენიმე მოკლეთი შეცვალოთ (ისევე, როგორც სალაპარაკო ენაში). პროგრამა ასე უფრო ადვილი აღსაქმელი იქნება.

და ბოლოს, ყველაზე მნიშვნელოვანი წესი: ეცადეთ თქვენი პროგრამა იყოს რაც შეიძლება მარტივი და ნათელი, თუნდაც თქვენ მოგიხდეთ ზოგიერთი აქ მოყვანილი წესის დარღვევა. მიზანი – ნათელი და გასაგები პროგრამის დაწერაა და ეს წესები იმისთვისაა, რომ დაგეხმაროთ ამ მიზნის მიღწევაში.

პროგრამები, ამოცანის დასმიდან შესრულებამდე

მაღალი დონის პროგრამები C++ ენაზე იწერება კლავიატურაზე მოთავსებული სიმბოლოების გამოყენებით. რადგან კომპიუტერი რეალურად ასრულებს (უშვებს) მხოლოდ დაბალი დონის, მანქანურ კოდებში დაწერილ პროგრამას, ამიტომ, C++ –ის პროგრამები განიცდის მთელ რიგ გარდაქმნებს, ვიდრე მათი შესრულება (გაშვება) მოხდება.



პროგრამაზე მუშაობა იწყება ამოცანის დასმით ჩვეულებრივ სალაპარაკო ენაზე, რაც შეიძლება იყოს ზეპირი ან წერილობითი, შესაძლოა დიაგრამების, ცხრილების, გრაფიკების, ფორმულების ან სხვა რაიმე მოდელის გამოყენებით. მას შემდეგ, რაც პროგრამისტი მოიფიქრებს ამოცანის გადაჭრის გზას, ანუ ალგორითმს, იგი ტექსტური რედაქტორის დახმარებით ქმნის **საწყის ფაილს**, რომელიც შეიცავს **საწყის კოდს** (source code). საწყის კოდს პროგრამა სახელად **კომპილერი** გადაიყვანს საბოლოო ფაილში (object file). შინაარსობრივად ესაა ფაილი, რომლის შექმნაც წარმოადგენდა ჩვენს მიზანს. შემდეგ კიდევ ერთი პროგრამა (linker) ერთად აკინძავს საბოლოო ფაილს და ყველა იმ ფუნქციას, რომლებსაც ის იძახებს სტანდარტული თუ სხვა სახის ბიბლიოთეკებიდან. შედეგად მიიღება შესრულებადი პროგრამა (executable program) მანქანური ენის ინსტრუქციების კრებული. ამჟამად, პროგრამები ამოცანის დასმიდან შესრულებამე ყველა ფაზას გადიან ე.წ. ინტეგრებულ გარემოში (IDE), რომელიც შეიცავს ტექსტურ რედაქტორს, სხვადასხვა მენიუს, მართვის ღილაკებს და ა.შ.. ჩვენ შემთხვევაში ესაა ***Visual Studio***. ასეთ გარემოს ბევრი ღირსება გააჩნია. ერთ-ერთი ისაა, რომ პროგრამისტს არ სჭირდება იზრუნოს საწყისი კოდის გარდაქმნებზე და საჭირო მომენტში საჭირო პროგრამები (კომპილერი და ამკინძავი) თვითონ იწყებენ მუშაობას.

თავი 3: ზოგადი ცნობები ცვლადების შესახებ

- მოკლედ პროგრამის ძირითადი ელემენტების შესახებ
- ცვლადი: მეხსიერება, სახელი, ზომა
- ცვლადის ინიციალიზაციის ორი მარტივი გზა, მინიჭების შეტყობინება და რეზერვირებული სიტყვა **`const`**
- მარტივი გამოსახულებები
- **`bool`** ტიპი, **`if-else`** კონსტრუქცია

მოკლედ პროგრამის ძირითადი ელემენტების შესახებ

პროგრამის ძირითადი ელემენტები არის თავსართი (სათაურების) ფაილები, განაცხადები ცვლადებზე, ფუნქციები და შესრულებადი შეტყობინებები.

თავსართი ფაილების ჩართვა ხდება პრეპროცესორის `#include` დირექტივით. როდესაც ბიბლიოთეკა ჩაშენებულია C++-ში, მაშინ თავსართი ფაილის სახელი, როგორც წესი, თავსდება მეტობის და ნაკლებობის ნიშნებისგან გაკეთებულ ფრჩხილებში. მაგალითად:

```
#include<iostream>
```

როდესაც ისეთ ბიბლიოთეკას ვრთავთ, რომელიც არაა ჩაშენებული C++-ში, მაშინ თავსართი ფაილის სახელი, როგორც წესი, თავსდება ორმაგ ბრჭყალებში. მაგალითად:

```
#include "my_math.h"
```

თავსართი ფაილების სახელები მჭიდრო კავშირშია მათ შინაარსთან. მაგალითად, `iostream` ნიშნავს შეტანა-გამოტანის (`input-output`) ნაკადების (`stream`) თავსართ (`header`) ფაილს.

ცვლადებზე განაცხადი კეთდება მათთვის მეხსიერების გარკვეული მონაკვეთის "დაჯავშვნის" მიზნით, რომელშიც შეინახება ამ ცვლადის მიმდინარე მნიშვნელობები. უნდა გავითვალისწინოთ, რომ თუ განაცხადის გაკეთების მომენტში არ მივანიჭებთ ცვლადს გარკვეულ მნიშვნელობას (ანუ არ მოვახდენთ მის **ინიციალიზაციას),** სისტემა მის მნიშვნელობად აღიქვამს იმას, რაც ამ მომენტისთვის წერია მეხსიერების ამ მონაკვეთში - C++ ენაში მეხსიერების მონაკვეთის გამოყოფა არ ნიშნავს მის დამუშავებას ან გასუფთავებას.

ჩვეულებრივი სალაპარაკო ენის წინადადებებს C++-ში შეესაბამება შეტყობინებები (`statements`). და რატომ აქაც წინადადებები არა? იმიტომ, რომ წინადადებები განკუთვნილია ადამიანებისთვის, ხოლო შეტყობინებები კომპილერისთვის, რომელსაც, თავის მხრივ, შეტყობინებები გადაჰყავს მანქანურ ინსტრუქციებში (იმედი გვაქვს, ასეთ განმარტებას რასიზმის გამოვლინებად არ ჩათვლით). სასვენი ნიშნების საქმე C++-ში შედარებით მარტივადაა, აქ ყოველი შეტყობინება მთავრდება წერტილ-მძიმით, რაც ნიშნავს ამ შეტყობინების დასასრულს, დამთავრებას. საზოგადოდ, შეგვიძლია ერთ სტრიქონში რამდენიმე შეტყობინების ჩაწერა, მაგრამ ეს არაა კარგი სტილი, პროგრამა ადვილად წასაკითხი რომ იყოს, სჯობს ერთ სტრიქონში მხოლოდ ერთი შეტყობინება დავწეროთ.

მეორეს მხრივ, ზოგიერთი სხვა ენისგან განსხვავებით, C++ –ში ხშირად გვხვდება შეტყობინებები, რომლებიც იკავებენ რამდენიმე სტრიქონს.

ცვლადი: მეხსიერება, სახელი, ზომა

განაცხადები კეთდება როგორც ცვლადებზე, ასევე ფუნქციებზე. ერთის მხრივ, განაცხადი წარმოადგენს მოცემულ ამოცანაში გამოყენებული ცვლადების და ფუნქციების აღწერას. მაგალითად, თუ რაიმე გამოყენებითი ამოცანის მათემატიკურ მოდელს ვაპროგრამებთ და პროგრამაში უნდა გამოვიყენოთ მათემატიკურ მოდელში აღწერილი ფუნქცია $f(x): R \to Z$ და ცვლადი $j \in Z$, მაშინ ისინი უნდა აღვწეროთ პროგრამაშიც C++ ენის სინტაქსის დაცვით:

```
int f (float x);
int j;
```

მეორეს მხრივ, რაც ადამიანისთვის წარმოადგენს ფუნქციის და ცვლადის აღწერას, იგივე ჩანაწერი კომპილერისთვის წარმოადგენს განაცხადს მეხსიერების მონაკვეთზე, რომელშიც მოთავსდება მოცემული ფუნქციის და ცვლადის შესახებ ყველა საჭირო ინფორმაცია. მეხსიერების მოცემული მონაკვეთები გაიგივდება შესაბამის ფუნქციასთან და ცვლადთან, ანუ დაიჯავშნება მათთვის. პროგრამის იმ ბლოკის დასრულებამდე, რომელშიც ეს განაცხადებია გაკეთებული, მეხსიერების ეს ნაწილი სისტემის მიერ სხვა მიზნით არ იქნება გამოყენებული.

მოცემულ მარტივ მაგალითში ფუნქციის და ცვლადის სახელად თითო ასო არის გამოყენებული, რაც არაა შესაბამისობაში პროგრამირების კარგ სტილთან (შინაარსს ვერ გამოხატავს). უმეტეს შემთხვევაში სახელები საკმაოდ გრძელია და ხშირად შედგება რამდენიმე ერთმანეთთან შეერთებული სახელისგან. ასეთი სახელებისთვის, პროგრამირების კარგი სტილი იძლევა შემდეგ რეკომენდაციებს:

- სახელები შევაერთოთ ზედა რეგისტრის ტირეთი (ან ტირეებით, თუ რამდენიმეა), მაგალითად `balance_owed`. ამ სტილს ძირითადად C -პროგრამისტები იყენებენ;
- ყოველი ახალი სახელი დავიწყოთ მთავრული ასოთი, მაგალითად `balanceOwed, bankBill.` ზოგჯერ პირველი ასოც მთავრულია. ეს საკმაოდ მოხერხებულია და ამიტომ ბევრი პროგრამისტი იყენებს ასეთ სტილს, რომელსაც გასაგები მიზეზების გამო პროგრამისტების სლენგზე ეწოდება “camel” (აქლემი).

რა ტიპისაც არ უნდა იყოს ცვლადი, მისი სახელის შედგენისას უნდა გავითვალისწინოთ შემდეგი შეზღუდვები:

ცვლადების სახელებში გამოიყენება ასოები, ციფრები და ზედა რეგისტრის ტირეს სიმბოლო (”_”). სახელის ციფრით დაწყება არ შეიძლება. საზოგადოდ, ზედა რეგისტრის ტირეთი იწყება სისტემური სახელები. სტანდარტული ბიბლიოთეკების ფუნქციებში ხშირად გამოიყენება ამ სიმბოლოთი დაწყებული სახელები, ამიტომ არ არის რეკომენდებული ცვლადის სახელის ზედა რეგისტრის ტირეთი დაწყება. რადგან C++ განასხვავებს ზედა და ქვედა რეგისტრის სიმბოლოებს, ამიტომ კომპილერისთვის `Gia, gia, GIA` სხვადასხვა სახელებია.

C++ ენის სტანდარტული კონსტრუქციები ეფუძნება რამდენიმე ხშირად გამოყენებად სახელს, მაგალითად, **`int, while, for, float`** და სხვა, რომელთაც C++ ენის რეზერვირებული (მომსახურე) სიტყვები ეწოდება. მათი გამოყენება არ შეიძლება ცვლადების სახელებად.

მაგალითად, ცვლადის სახელი **შეიძლება იყოს:**

```
sashualo              /* ყველა გაზომვის საშუალო მნიშვნელობა */
pi                    /* პი რიცხვის მნიშვნელობა მეათასედის სიზუსტით */
studentebis_raoden    /* ჯგუფში სტუდენტების რაოდენობა */
```

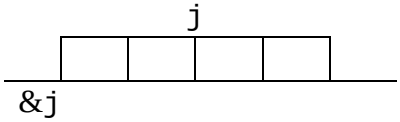
ცვლადის სახელი **არ შეიძლება** იყოს შემდეგი:

```
3_gazomva             // იწყება ციფრით
fasi$                 // შეიცავს სიმბოლო $-ს
studentebis raoden    // შეიცავს ჰარს
int                   // რეზერვირებლი სიტყვაა
```

თავი უნდა ავარიდოთ ერთ პროგრამაში ცვლადებისთვის ერთმანეთის მსგავსი სახელების გამოყენებას, მაგალითად:

```
total                 // ერთ ჯერზე შეტანილი ცვლადების ჯამი
totals                // ყველა ცვლადის ჯამი
```

სხვადასხვა ტიპის ცვლადებს არა მარტო სახელების შედგენის წესები აქვთ საერთო. ნებისმიერი ტიპის ცვლადს აქვს მისამართი და სიგრძე. ზოგადად, განაცხადის გაკეთების მომენტში ცვლადის სახელი გაიგივდება მისთვის გამოყოფილ მეხსიერების მონაკვეთთან. თვითონ მეხსიერება შეიძლება გავაიგივოთ სხივთან, რომელიც დაყოფილია ერთბაიტიან მონაკვეთებად, და ეს მონაკვეთები დანომრილია ერთმანეთის მომდევნო არაუარყოფითი რიცხვებით. მაგალითად, თუ `j` არის მთელი ტიპის ცვლადის სახელი, მაშინ ამ ცვლადის მისამართს, ანუ `j`-სთან გაიგივებული მონაკვეთის დასაწყისს გავიგებთ C++ -ის ოპერატორ ”&”-ის გამოყენებით, - `&j` წარმოადგენს `j` ცვლადის მისამართს.



ნახაზზე `j`-ის მონაკვეთი გაყოფილია ოთხ ნაწილად იმის მისათითებლად, რომ მთელი რიცხვების წარმოდგენისთვის გამოყენებულია 32-ბიტი. თუმცა ეს დამოკიდებულია სისტემაზე. ზოგადად, ცვლადისთვის საჭირო მეხსიერების სიდიდეს გავიგებთ C++ -ის ოპერატორ **`sizeof`** -ის საშუალებით. მაგალითად, თუ დავბეჭდავთ **`sizeof`**(`j`) სიდიდეს, დაიბეჭდება სისტემის მიერ `j` ცვლადისთვის გამოყოფილი ბაიტების რაოდენობა. იგივე ოპერატორი შეგვიძლია გამოვიყენოთ უშუალოდ ტიპებისთვის. მაგალითად, დავბეჭდოთ და ვნახოთ რისი ტოლია **`sizeof`**(**`int`**), **`sizeof`**(**`double`**) და ა.შ.. თუ ვიცით რამდენი ბაიტია გამოყოფილი ცვლადისთვის, შეგვიძლია გამოვთვალოთ რა დიაპაზონის და რა სიზუსტის მიღწევა შეიძლება ამ ტიპის საშუალებით. მაგალითად, თუ **`int`** ტიპის ცვლადისთვის ოთხი ბაიტია გამოყოფილი, მაშინ ყველაზე დიდი მთელი (**`int`**) რიცხვი არის $2^{31}-1$ (გავიხსენოთ რიცხვების წარმოდგენის `two's complement` სისტემა). მაგრამ ამის ანგარიშის ან დამახსოვრების აუცილებლობა არ არის, რადგან

```
#include <cmath>
```

დირექტივა საშუალებას გვაძლევს გამოვიყენოთ ამ ბიბლიოთეკაში აღწერილი მუდმივი სიდიდეები. მაგალითად, თუ ჩართულია ეს ბიბლიოთეკა, კომპილერისთვის `INT_MAX` მუდმივი არის ზუსტად უდიდესი **`int`** მთელი რიცხვი, რომლის წარმოდგენაც შესაძლებელია **`sizeof`**(**`int`**) რაოდენობა ბაიტებში. ანგარიში ჩვენ არ გვჭირდება. თუ რამდენიმე წლის შემდეგ **`sizeof`**(**`int`**) გაიზრდება, შესაბამისად გაიზრდება `INT_MAX`-იც.

ყურადღება უნდა მივაქციოთ განსხვავებას ”&” და ”&&” ოპერატორებს შორის. პირველი მათგანი არის მისამართის აღების ოპერატორი, მეორე - ლოგიკური ”და” ოპერატორი.

ცვლადის ინიციალიზაციის ორი მარტივი გზა, მინიჭების შეტყობინება და რეზერვირებული სიტყვა `const`

C++ ენა საშუალებას გვაძლევს, რომ განაცხადის გაკეთების მომენტში ყოველ ცვლადს (ტიპის მიუხედავად) მივცეთ საწყისი მნიშვნელობა ან გულისხმობის პრინციპით (იგივე უპარამეტრო კონსტრუქტორი), ან მინიჭების შეტყობინებით (ასლის გაკეთების კონსტრუქტორი). ვნახოთ რამდენიმე მარტივი მაგალითი:

```
int answer(55);
int position(( 1 + 2 ) * 4);
double x(12.563);
double height(x);    //როდესაც x ცვლადი უკვე არის ინიციალიზებული
```

და ანალოგიურად სხვა ტიპის ცვლადებისთვის. თუ ცვლადი მუდმივი არაა, მისთვის შესაძლებელია მნიშვნელობის შეცვლა პროგრამის განხორციელების პროცესში, მინიჭების შეტყობინებით. მაგალითად, განვიხილოთ პროგრამის ფრაგმენტი:

```
int size(3*5);          // size მონაწილეობს მომდევნო ორ გამოსახულებაში
int size_2;             // ცვლადი გაორმაგებული size-ისთვის
int size_3;             // ცვლადი გასამმაგებული size-ისთვის
  ...
size _2 = 2 * size;
size _3 = 3 * size;
  ...
```

ცხადია, შეგვეძლო ცვლადებისთვის დაგვერქმია სხვა სახელები და გაგვეკეთებინა იგივე.

ერთი ზოგადი გზა იმისათვის რომ C++ -ში ჩაშენებული რომელიმე ტიპის ცვლადის ინიციალიზება მოვახდინოთ განაცხადის გაკეთების შემდეგ, ან უბრალოდ შევცვალოთ მისი მიმდინარე მნიშვნელობა სხვა მნიშვნელობით, არის კლავიატურიდან (ან ფაილიდან) შემოტანილი ახალი მნიშვნელობის მინიჭება მისთვის..

მონაცემების შეტანა კლავიატურიდან C++ -ში სრულდება `cin` -ის საშუალებით (`cin` – `console input`). მაგალითად,

```
int number;

cin >> number;
```

ფრაგმენტის პირველი სტრიქონი წარმოადგენს მთელ number ცვლადზე განაცხადს, ხოლო მეორე სტრიქონი არის შეტანის შეტყობინება. მასში გამოიყენება cin და შეტანის ოპერატორი >>. შეიძლება ითქვას, რომ ჩვენ შემთხვევაში კლავიატურის ნაკადიდან ამოიღება მთელი რიცხვი და მიენიჭება >> ოპერატორის მარჯვნივ მდგომ number -ს.

```
string name;

cin >> name;
```

ფრაგმენტის შესრულების დროს კი კლავიატურაზე უნდა ავკრიფოთ სახელი (მაგალითად, Giorgi) და დავაჭიროთ Enter –ს. ამის შემდეგ სტრიქონის ტიპის name ცვლადს მიენიჭება მნიშვნელობა Giorgi (name ცვლადის შესაბამის მეხსიერებაში ჩაიწერება სტრიქონი Giorgi).

ზოგჯერ საჭირო ხდება ისეთი ცვლადების გამოყენება, რომლის მნიშვნელობები არ უნდა შეიცვალოს პროგრამის განხორციელების პროცესში. იმისათვის რომ თავი დავიზღვიოთ უნებლიე შეცდომისგან, ვიქცევით შემდეგნაირად. ვთქვათ, ხშირად გვჭირდება პროგრამაში π მუდმივის გამოყენება, რომლის მნიშვნელობაც არის დაახლოებით 3.14. შემოვიტანოთ ცვლადი და გავაკეთოთ განაცხადი და მოვახდინოთ ინიციალიზაცია:

```
const double PI(3.14);
```

ამის შემდეგ `PI` ცვლადს ჩვეულებრივ ვიყენებთ პროგრამაში და მისი მნიშვნელობა ყოველთვის არის 3.14. თუ ჩვენ შეგვეშალა და სადმე დავწერეთ

```
PI = 55.17;
```

კომპილერი მიგვითითებს შეცდომაზე. ამგვარად, რეზერვირებული სიტყვა **`const`** არის თავის დაზღვევის საშუალება როდესაც ვმუშაობთ მუდმივ სიდიდეებთან. მისი გამოყენება აგრეთვე შეიძლება სხვა საბაზო ტიპებისთვის, რომლებსაც ენა გვთავაზობს გამზადებული სახით.

თუ მივაქციეთ ყურადღება, მუდმივი (კონსტანტური) ცვლადის სახელი ჩვენ მთავრული (დიდი) ასოებით ავკრიფეთ. ესაც პროგრამირების კარგი სტილის რეკომენდაციაა: პროგრამისტი, რომელიც კითხულობს სხვის დაწერილ პროგრამას მისი გადაკეთების მიზნით, მთავრული ასოებით აკრეფილი ცვლადის სახელის დანახვისას ხვდება, რომ ეს მუდმივი ცვლადია და მისი მნიშვნელობა ერთხელ და სამუდამოდ არის განსაზღრულილი. თუ მისი

მნიშვნელობის შეცვლაა საჭირო, უნდა მოიძებნოს ინიციალიზაციის სტრიქონი და იქ გაკეთდეს ცვლილება.

მინიჭების შეტყობინების ზოგადი სახეა:

ცვლადი = *გამოსახულება;*

ოპერატორი " = " გამოიყენება მინიჭებისთვის. ეს შეტყობინება ამბობს: გამოთვალე გამოსახულება და მიანიჭე ცვლადს ამ გამოსახულების მნიშვნელობა.

ვიდრე ვისაუბრებთ გამოსახულებების შესახებ, რაც აუცილებელია მინიჭების შეტყობინების ზოგადი სახის გამო, კიდევ ერთხელ გავამახვილოთ ყურადღება " = " ოპერატორის ერთ სახასიათო თავისებურებაზე. შემდეგ ფრაგმენტში

```
int counter;          //  მთვლელი
int value = 44;       //  სიდიდე მთვლელის საწყისი მნიშვნელობისთვის
  ...
counter = value;
  ...
```

გვაქვს ორი მინიჭება. პირველი კეთდება განაცხადის გაკეთების მომენტში, ანუ სხვა სიტყვებით ცვლადი `value` ინიციალიზდება მნიშვნელობით 44. მეორე მინიჭება ცვლადს `counter` ანიჭებს იგივე მნიშვნელობას, რაც ჰქონდა ცვლადს `value`, თანაც ამ უკანასკნელის მნიშვნელობა არ იცვლება. სხვა სიტყვებით, ამ შემთხვევაში " = " ოპერატორი აკეთებს მარჯვენა მხარეში მდგომი ცვლადის ასლს მარცხენა მხარეში მდგომ ცვლადში. ეს ანალოგია ასლის გადაღებასთან დაგვეხმარება რომ გავიგოთ ე.წ. მინიჭების კონსტრუქტორის შინაარსი ისეთი ტიპის ცვლადებისთვის, რომლებსაც არა აქვთ რიცხვითი მნიშვნელობები (მაგალითად, სხვადასხვა კლასის ობიექტები).

ყურადღება უნდა მივაქციოთ განსხვავებას "=" და "==" ოპერატორებს შორის. პირველი მათგანი არის მინიჭების ოპერატორი, მეორე - შედარების ოპერატორი.

მარტივი გამოსახულებები

გამოსახულება ჩვენს კურსში იგივეს ნიშნავს რაც მათემატიკის სასკოლო კურსში: გამოსახულება შედგება (სხვადასხვა ტიპის) რიცხვების, ცვლადების, ფრჩხილების და არითმეტიკული მოქმედებისგან. შემდეგ ცხრილში ჩამოთვლილია C++ ენაში გამოყენებული 5 მარტივი ოპერატორი:

მარტივი ოპერატორების ცხრილი:

ოპერატორი	მოქმედება
*	გამრავლება
/	გაყოფა
+	შეკრება
-	გამოკლება
%	ნაშთი (მთელი გაყოფისას მიღებული)

შესრულების თვალსაზრისით, *, / და % ოპერატორებს აქვს მეტი პრიორიტეტი (უფრო მაღალი რიგი და სრულდება უფრო ადრე) + და – ოპერატორებთან შედარებით. მრგვალი ფრჩხილები გამოიყენება როგორც წევრების დასაჯგუფებლად, ასევე შესრულების რიგის შესაცვლელად. მაგალითად: (1+2)*3 = 9, მაშინ, როცა 1+2*3 = 7, ხოლო (4+5)%2 = 1.

შევადგინოთ პროგრამა, რომელიც გამოთვლის (1+2)*3 გამოსახულების მნიშვნელობას.

```
int main()
{
    ( 1 + 2 ) * 3;
    return (0);
}
```

ამ პროგრამის მიხედვით, კომპიუტერი გამოთვლის გამოსახულების მნიშვნელობას, მაგრამ ჩვენ ვერ ვნახავთ შედეგს ეკრანზე. ესაა ე.წ. "ნულოვანი ეფექტის" მაგალითი, როდესაც გვაქვს კორექტული პროგრამა, რომელიც სრულიად უსარგებლოა რადგან ვერ ვხედავთ მის შედეგს.

bool ტიპი

ზოგადად, ტიპი ნიშნავს გარკვეულ სიმრავლეს, ამ სიმრავლეზე განსაზღვრულ ოპერაციებთან ერთად. როდესაც ცვლადზე ვაკეთებთ განაცხადს, აუცილებლად მივუთითებთ ცვლადის ტიპს. ეს საკმარისია, რომ კომპილერმა გამოყოს ადგილი ცვლადისთვის და მერე, რა ვითარებაშიც არ უნდა შეხვდეს ეს ცვლადი, კომპილერმა იცის რომელი ოპერაცია როგორ უნდა შეასრულოს, რომ შედეგი კორექტულად და სწრაფად მიიღოს.

ბულის ტიპის ცვლადმა ან მუდმივმა შესაძლოა მიიღოს მხოლოდ ორი მნიშვნელობა: 0 (იგივე **false**) და 1 (იგივე **true**). ამ რიცხვებისთვის გადატვირთულია არითმეტიკული ოპერატორები ისე, რომ შედეგი არ გავიდეს ბულის სიმრავლიდან. ბულის რიცხვების შეკრება სრულდება ისევე როგორც ჩვეულებრივი რიცხვებისთვის,

გამოსახულება	შედეგი
1 + 1	0
1 + 0	1
0 + 1	1
0 + 0	0

ხოლო სხვა ფუნქციები, რაც ვახსენეთ ზემოთ (**sizeof**(), &) აგრეთვე გადატვირთულია ამ ტიპზე.

თითოეული ტიპი უნიკალურია, ამიტომ თითოეულ ტიპთან დაკავშირებულია სპეციფიკური ფუნქციები (ოპერატორები). განვიხილოთ ბულის ტიპის სპეციფიკა.

C++ ენაში ნებისმიერი გამოსახულება (შედარების, ლოგიკური, არითმეტიკული) ინტერპრეტირდება როგორც **bool** ტიპის მნიშვნელობა. ნებისმიერი ჭეშმარიტი შედარების გამოსახულება (მაგალითად 8 <= 124) C++ –ში გაიგივებულია 1-იანთან, ხოლო მცდარი (მაგალითად 453 < -3) გაიგივებულია ნულთან. იგივე ეხება ლოგიკურ გამოსახულებასაც. მაგალითად, 5 > 3 && 123 <= 3736 გამოსახულების მნიშვნელობაა 1 (**true**), ხოლო 5 > 3 && 123 > 3736 გამოსახულების შედეგია 0 (**false**). არითმეტიკული გამოსახულების გამოთვლის არანულოვანი შედეგი (–100, 2.456, 9) გაიგივდება **bool** ტიპის მნიშვნელობასთან 1 (**true**), ხოლო ნულოვანი შედეგი – მნიშვნელობასთან 0 (**false**).

ეს ძალიან მნიშვნელოვანია პროგრამის შემადგენელი ინსტრუქციების მიმდევრობის მართვისთვის. ისეთი ფუნდამენტური კონსტრუქციები, როგორიცაა **if-else** და განმეორების შეტყობინებები (ციკლები), არგუმენტებად ღებულობენ ბულის ტიპის გამოსახულებებს. ამჟამად განვიხილოთ პირველი მათგანი, რომლისთვისაც ბულის ტიპის ცვლადების ცოდნა საკმარისია.

if შეტყობინების ზოგადი ფორმა ასეთია:

if (*პირობა*)

შეტყობინება;

თუ პირობა ჭეშმარიტია (ნულისაგან განსხვავებულია), მაშინ *შეტყობინება* სრულდება, ხოლო თუ მცდარია (0-ის ტოლია) - არ სრულდება. მაგალითად

№	პროგრამის ფრაგმენტი	შედეგი

1	`cout << "AAA\n";`
	`if(5 > 3)  cout << "BBB\n";`
	`cout << "EEE\n";`
		AAA
		BBB
		EEE

2	`cout << "AAA\n";`
	`if(15 <= 3)  cout << "BBB\n";`
	`cout << "EEE\n";`
		AAA
		EEE

აქ ">" და "<=" წარმოადგენენ შედარების ოპერატორებს. შედარების ოპერატორების სრული სია მოყვანილია შემდეგ ცხრილში.

ოპერატორი	მნიშვნელობა
<=	ნაკლებია ან ტოლი
<	ნაკლებია
>	მეტია
>=	მეტია ან ტოლი
==	უდრის
! =	არ უდრის

ყურადღება უნდა მივაქციოთ, რომ ოპერატორი == შედგება ორი = ნიშნისგან და არ უნდა ავურიოთ მინიჭების შეტყობინებაში.

რამდენიმე შეტყობინების დაჯგუფება შეიძლება მათი ფიგურულ ფრჩხილებში ჩასმით. მაგალითად,

```
cout << "AAA\n";
if(5 > 3)
{
    cout << "BBB\n";
    cout << "DDD\n";
}
cout << "EEE\n";
```

ფრაგმენტის შესრულების შემდეგ დაიბეჭდება

```
AAA
BBB
DDD
EEE
```

ყველა ის შეტყობინება, რომლის შესრულება დამოკიდებულია **if**-ის პირობის შესრულებაზე, წარმოადგენს ერთ ბლოკს და შეწეულია ერთნაირად.

if შეტყობინების სრული ფორმა ასეთია:

if (*პირობა*)
    შეტყობინება1;
else
    შეტყობინება2;

თუ პირობა ჭეშმარიტია (ნულისაგან განსხვავებულია), მაშინ სრულდება *შეტყობინება1*, ხოლო თუ მცდარია (0-ის ტოლია), მაშინ- *შეტყობინება2*. სხვა სიტყვებით, *შეტყობინება1* და *შეტყობინება2* ერთმანეთის ალტერნატივებია და მხოლოდ ერთი მათგანი სრულდება. მაგალითად:

№	პროგრამის ფრაგმენტი	შედეგი
1	`cout << "1111\n";` `if(15 <= 3)  cout << "2222\n";`	1111 3333

	else cout << "3333\n"; cout << "4444\n";	4444
2	**int** a =11; cout << "1111\n"; **if**(a >= 5) cout << "2222\n"; **else** cout << "3333\n"; cout << "4444\n";	1111 2222 4444
3	**if**(1) cout << "yes"; **else** cout << "no";	yes
4	**int** k =7; **if**(k > 0) cout << k << " > 0\n"; **else** cout << k <<" <= 0\n";	7 > 0
5	**int** x =8; **if**(x % 2 == 0) cout << x <<" is even\n"; **else** cout << x << " is odd\n";	8 is even

პირობა შესაძლოა შედარებით რთული სახისაც იყოს. მაგალითად:

№	პროგრამის ფრაგმენტი	შედეგი
1	**if**(5 > 3 && 123 <= 3736) cout << "1111\n"; **else** cout << "2222\n";	1111
2	**int** x =8; **if**(x <= 3736 || 5 > 333) cout << "1111\n"; **else** cout << "2222\n";	1111
3	**if**(5 > 333 || 123 > 3736) cout << "1111\n"; **else** cout << "2222\n";	2222
4	**if**(14) cout << "1111\n"; **else** cout << "2222\n";	1111
5	**if**(2*5 - 1.2) cout << "1111\n"; **else** cout << "2222\n";	1111
6	**if**(2*5 - 10) cout << "1111\n"; **else** cout << "2222\n";	2222

საინტერესოა თუ როგორ ხდება კომპილერის მიერ && და || ოპერატორების გამოყენებით შედგენილი პირობის შემოწმება. მოკლედ რომ ვთქვათ, თუ აუცილებელი არაა, C++ არ განიხილავს ამ ოპერატორების ორივე ოპერანდს. მაგალითად, ბოლო ცხრილის მეორე მაგალითში ||-ის მარჯვნივ მდგარი პირობა აღარ შემოწმდება, რადგან მის მარცხნივ მდგარი პირობა 1-ის ტოლია და ამიტომ ლოგიკური შეკრებაც ერთის ტოლია. იგივე ცხრილის მესამე მაგალითში ორივე მხარის პირობების შემოწმება გახდა აუცილებელი რადგან პირველი პირობა არ შესრულდა (თუმცა ლოგიკური შეკრების შედეგი მაინც ნულის ტოლი დარჩა). აქვე, პირველ მაგალითში, აგრეთვე ორივე მხარის ჭეშმარიტობა შესრულდა. პირველი პირობა რომ ყოფილიყო 3>5 სახის, მაშინ მარჯვნივ აღარ გადავიდოდით, რადგან 0 && x არის 0 ნებისმიერი x -ისთვის.

საკმაოდ ხშირად გვხვდება ჩალაგებული **if**-**else**-ები. ამ დროს მთავარია გვახსოვდეს, რომ **else** ეკუთვნის მის წინ მდგომ პირველივე **if**-ს. მაგალითად, ვთქვათ პროგრამაში უკვე შეყვანილია ორი არანულოვანი ნამდვილი რიცხვი (x და y) და ჩვენს ამოცანას წარმოადგენს გავარკვიოთ და დავბეჭდოთ თუ სიბრტყის რომელ საკოორდინატო მეოთხედში არის მოთავსებული (x,y) წერტილი. ამ მიზნით შეგვიძლია გამოვიყენოთ შემდეგი კოდი:

```
if (x > 0)
{
    if ( y > 0)
       cout << "პირველი მეოთხედი\n";
```

```
        else
           cout << "მეოთხე მეოთხედი\n";
    }
    else
    {
         if ( y > 0)
             cout << "მეორე მეოთხედი\n";
        else
            cout << "მესამე მეოთხედი\n";
    }
```

თუმცა, რადგან **else** ეკუთვნის მის წინ მდგომ პირველივე **if**-ს, ამიტომ იგივე იქნებოდა თუ დავწერდით:

```
    if (x > 0)
         if ( y > 0)
            cout << "პირველი მეოთხედი\n";
        else
           cout << "მეოთხე მეოთხედი\n";
    else
         if ( y > 0)
             cout << "მეორე მეოთხედი\n";
        else
            cout << "მესამე მეოთხედი\n";
```

ზოგადად, ფრჩხილების გამოყენება ყოველთვის უფრო უსაფრთხოა.

თავი 4. მთელი რიცხვები

- ტიპი **char**
- მთელი რიცხვების ტიპები
- **switch** შეტყობინება

ტიპი **char**

ტიპი **char** აგრეთვე განეკუთვნება მთელ ტიპებს. ამ ტიპის რიცხვი იკავებს მეხსიერებაში ერთ ბაიტს და მოთავსებულია დიაპაზონში (-128)-იდან 127-მდე, ბოლოების ჩათვლით. გულისხმობის პრინციპით **char** ნიშნიანი (**signed**) ტიპია. **unsigned char** ფარავს დიაპაზონს 0–დან 255–მდე.

char ხშირად მოიხსენიება, როგორც სიმბოლური ტიპი. აქ წინააღმდეგობა არ არის, რადგან ყოველ სიმბოლოს კომპიუტერის მეხსიერებაში შეესაბამება მოკლე მთელი რიცხვი – მისი კოდი. სიმბოლური წარმოდგენა გვჭირდება, როდესაც სიმბოლური ინფორმაცია შეგვაქვს კლავიატურიდან ან გამოგვაქვს ეკრანზე (ან პრინტერზე), ხოლო კომპიუტერში სიმბოლოები შენახულია მათი კოდების საშუალებით. სიმბოლოს გარდაქმნა კოდში და პირიქით ხდება ავტომატურად. თითოეული სიმბოლოს კოდი მოყვანილია `ASCII (American Standard Code for Information Interchange)` ცხრილში, რომელიც კოდების სტანდარტული ცხრილია და მოქმედებს მთელს მსოფლიოში. მაგალითად, სიმბოლო-ციფრი `'0'` წარმოდგენილია კომპიუტერში ორობითი რიცხვით 00110000, რაც შეესაბამება ათობით მნიშვნელობას 48, ე.ი. სიმბოლო `'0'`-ის კოდია `48 (ASCII` ცხრილის მიხედვით).

ამრიგად, **char** ტიპი პროგრამაში გამოიყენება ორ შემთხვევაში: თუ ვმუშაობთ ძალიან მოკლე მთელ რიცხვებთან ან თუ ვმუშაობთ სიმბოლოებთან. ჩვენ შეგვიძლია დავბეჭდოთ ამ ტიპის ცვლადი ორივენაირად. მაგალითად, შემდეგი ფრაგმენტი

```
char ch('F');
    cout << int(ch) << endl;
    cout << ch << endl;
```

დაბეჭდავს ჯერ კოდს (70), ხოლო შემდეგ თვითონ `F` სიმბოლოს. იგივე შედეგს მივიღებთ, თუ განაცხადს გავაკეთებთ

```
    char ch(70);
```

სახით.

C++-ს სიმბოლოებისთვის გააჩნია შეტანა-გამოტანის სპეციალური სტანდარტული ფუნქციებიც: `getchar` *და* `putchar`*, ორივე ფუნქცია მოითხოვს* `#include <cstdio>` *დირექტივას.*
`getchar` *–ს აქვს სახე:*

```
    getchar()
```

ფუნქცია "კითხულობს" თითო სიმბოლოს კლავიატურიდან და აბრუნებს მის კოდს. მაგალითად,

```
    char ch;
    ch = getchar();
```

ფრაგმენტის შესრულების შემდეგ `ch` *ცვლადის მნიშვნელობა იქნება* `getchar`*-ით კლავიატურიდან წაკითხული სიმბოლო (მისი კოდი).*

`putchar` *ფუნქცია გეჭდავს სიმბოლოს ეკრანზე. მისი სახეა:*

```
    putchar(სიმბოლო)
```

მაგალითად,

```
    char ch ='Z';
    putchar(ch);
```

ფრაგმენტი დაბეჭდავს ეკრანზე `Z` *სიმბოლოს.*

getchar ფუნქციის გამოყენებისას გვმართებს სიფრთხილე, რადგან ფუნქცია არ ახდენს ჰარების იგნორირებას. საკმარისია 'A'–ს ნაცვლად ავკრიბოთ ჰარი და 'A', რომ მივიღოთ შემდეგი შედეგი: სიმბოლოდ ჩაითვლება ჰარი , ხოლო ასო 'A' დარჩება ბუფერში.

ანალოგიური პრობლემები ჩნდება, როდესაც getchar –ით შესაყვანი გვაქვს სიმბოლო, ხოლო პროგრამამ ამ მომენტისთვის უკვე მოახდინა რაიმე სხვა მონაცემის შეყვანა. მაშინ, ბოლო შეყვანის მანიშნებელი '\n' სიმბოლო, რომელიც ბუფერში არის დარჩენილი, ავტომატურად წავა შესატანი სიმბოლოს ნაცვლად.

C++ –ში საერთოდ ძნელია შეცდომების აღმოჩენა და გასწორება, მათ შორის ამ ტიპის შეცდომებისაც. ამ შემთხვევაში, გამოსავალს წარმოადგენს სიმბოლოს შემტანი getchar –ის წინ კიდევ ერთი getchar ფუნქციის გამოყენება, რომელიც "მოაშორებს" ზედმეტ სიმბოლოს. ამ ტიპის მაგალითები განხილულია ჩვენი კურსის პრაქტიკულ და ლაბორატორიულ სავარჯიშოებში.

getchar და putchar ფუნქციები ძირითადად უნდა გამოვიყენოთ მაშინ, როდესაც პროგრამა ამუშავებს მხოლოდ სიმბოლურ ინფორმაციას.

getchar ფუნქციის მსგავსია getch. მისი პროტოტიპი მოცემულია conio.h ფაილში. მას, განსხვავებით getchar-ისგან, კლავიატურის ღილაკზე თითის დაჭერისთანავე შეჰყავს სიმბოლო. რომელიმე ღილაკზე ხელის დაჭერამდე getch -ის მოქმედება ანალოგიურია system("PAUSE") ფუნქციის მოქმედებისა, ამიტომ ხშირად იგი ამ მიზნით გამოიყენება. ეს ფუნქციები საერთოა C და C++ –ისთვის. C –ში ჩნდება ეკრანის გაჩერების პრობლემა, როდესაც მონაცემები შეგვაქვს ფაილიდან. ასეთ შემთხვევებში system("PAUSE") –ის ნაცვლად ვიყენებთ getch ფუნქციას, როგორც ნაჩვენებია ქვემოთ.

```
#include <iostream>
#include <cstdio>
#include <conio.h>
using namespace std;
int main()
{
    char p;
    cout<<"Enter a symbol : ";
    p = getchar();                    // სიმბოლოს წაკითხვა
    cout<<"The symbol is : ";
    putchar(p);                       // სიმბოლოს ბეჭდვა
    getch();                          // ეკრანის გაჩერება
    cout<<endl;
    return 0;
}
```

შესრულების შედეგი შეიძლება იყოს:

```
Enter a symbol : B
The symbol is : B
Press any key to continue . .
```

მთელი რიცხვების ტიპები

C++ ფართო არჩევანს სთავაზობს პროგრამისტს მთელი რიცხვების გამოყენებისათვის. ყველაზე ხშირად გამოიყენება **int** ტიპი. მისი ზომის გარკვევა შეიძლება **sizeof** ფუნქციის საშუალებით. შეგვიძლია გამოვიყენოთ აგრეთვე **long int** და **short int** ტიპები, მაგრამ ამის აუცილებლობა ნაკლებად იგრძნობა ხოლმე. ერთერთი ფაქტი, რაც უნდა გვახსოვდეს, ისაა რომ შეგვიძლია **long int**–ის ნაცვლად ვწეროთ **long**, ხოლო **short int**–ის ნაცვლად ვწეროთ **short.** ნებისმიერ შემთხვევაში:

თუ რაიმე ამოცანაში გვჭორდება მხოლოდ არაუარყოფითი რიცხვები, მაშინ შეგვიძლია განაცხადის გაკეთების მომენტში გამოვიყენოთ რეზერვირებული სიტყვა **unsigned**, რაც საშუალებას გვაძლევს გავაორმაგოთ დადებითი მთელი რიცხვების დიაპაზონი უარყოფითი რიცხვების გამორიცხვის ხარჯზე.

შემდეგი საილუსტრაციო მაგალითი გვიჩვენებს რამდენ ბაიტს იკავებს და რა უდიდეს და უმცირეს მნიშვნელობებს ღებულობს ზოგიერთი მთელი ტიპი:

```
#include<iostream>
using namespace std;

int main()
{
   int x = INT_MIN, y = INT_MAX;
   unsigned int ux = UINT_MAX;
   long int lx = LONG_MIN, ly = LONG_MAX;
   unsigned long int lu = ULONG_MAX;
   short int sx = SHRT_MIN, sy = SHRT_MAX;
   unsigned short int su = USHRT_MAX;
   cout<<"int ikavebs "<<sizeof(int)<<" baits"<<endl;
   cout<<"misi diapazonia  " << x <<"\t:  " << y <<endl;
   cout<<"unsigned int-is diapazonia   0 : "<< ux << endl;
   cout<<"long int ikavebs " <<sizeof(long) <<" baits"<< endl;
   cout<<"misi diapazonia  "<< lx <<"\t:  " << ly <<endl;
   cout<<"unsigned long-is diapazonia   0 : " << lu << endl;
   cout<<"short int ikavebs " <<sizeof(short)<<" baits"<< endl;
   cout<<"misi diapazonia:  " << sx <<" :  "<< sy << endl;
   cout<<"unsigned short int-is diapazonia    0 : "<< su << endl;
   system("pause");
   return 0;
}
```

პროგრამის შესრულების შედეგია:

```
int ikavebs 4 baits
misi diapazonia  -2147483648    :  2147483647
unsigned int-is diapazonia   0 : 4294967295
long int ikavebs 4 baits
misi diapazonia  -2147483648    :  2147483647
unsigned long-is diapazonia   0 : 4294967295
short int ikavebs 2 baits
misi diapazonia:  -32768 :  32767
unsigned short int-is diapazonia    0 : 65535
Press any key to continue . . .
```

switch შეტყობინება

switch შეტყობინება ხშირად გამოიყენება მთელი რიგი **if-else** შეტყობინებების სანაცვლოდ. იგი მჭიდროდაა დაკავშირებული მთელ რიცხვებთან, რადგან გამოსახულება, რომელიც წარმოადგენს **switch** -ის არგუმენტს (ანუ რომლის საფუძველზეც უნდა გადაწყვიტოს ამ შეტყოობინებამ თუ რა მიმდევრობით გაგრძელდეს პროგრამის შესრულება) აუცილებლად არის მთელი (ზოგჯერ ამბობენ, რომ არის რომელიმე ჩამოთვლადი ტიპის). მისი ზოგადი სახე ასეთია:

```
switch ( გამოსახულება )
{
      case კონსტანტა1:
            შეტყობინება;
            . . .
            break;
      case კონსტანტა2:
            შეტყობინება;
            . . .
```

```
        // სრულდება მომდევნო შეტყობინებაც
    default:
        შეტყობინება;
        . . .
        break;
    case კონსტანტა3:
        შეტყობინება;
        . . .
        break;
}
```

switch შეტყობინება ითვლის გამოსახულებას, რომელიც შეიძლება იყოს მთელი, სიმბოლური ან ჩამონათვალის ტიპის და ირჩევს შესასრულებლად იმ შტოს, რომლის კონსტანტაც გამოსახულების მნიშვნელობის ტოლია. თუ გამოსახულების მნიშვნელობა არ დაემთხვევა არცერთ კონსტანტა-ს, მაშინ შესრულდება **default** შტო. თუ ეს შტო არ არის განსაზრვრული **switch** შეტყობინებაში, მაშინ ის არაფერს არ გააკეთებს.

კონსტანტების განმეორება დაუშვებელია. შტოების თანმიმდევრობა შეიძლება სხვადასხვა გვარი იყოს. ხშირად, თუ განსაკუთრებული შემთხვევა არაა, **default** შტო ბოლო შემთხვევას წარმოადგენს. რეკომენდირებულია რომ იგი ყოველთვის ჩასვათ **switch** შეტყობინებაში.

შემდეგი მაგალითი შეიცავს **if-else** შეტყობინებების მთელ რიგს:

```
if (operat == '+' ){
   result += value;
}
else
   if (operat == '-' ){
      result -= value;
   }
   else
      if (operat == '*' ){
         result *= value;
      }
      else
        if (operat == '/' ){
           if ( value == 0 ){
              cout <<" Error: Divide by zero \n " ;
              cout <<" operation ignored \n ";
            }
            else
              result /= value;
      }
      else
         cout <<" Unknown operator" << operat << endl;
```

სადაც `operat` სიმბოლური ტიპის ცვლადია და აღნიშნავს ოპერაციის ნიშანს, `value` და `result` – მთელი ტიპის ცვლადებია და აღნიშნავენ შესაბამისად რიცხვს, რომელზეც სრულდება ოპერაცია და შედეგს.

კოდის ეს ნაწილი შეიძლება გადავწეროთ **switch** შეტყობინების საშუალებით, რომელშიც სხვადასხვა **case** შეტყობინებას ვიყენებთ სათანადო ოპერაციის შესასრულებლად. **default** შტო ზრუნავს ყველა იმ შემთხვევაზე, რომლებიც გათვალისწინებული არ გვაქვს.

აღვნიშნოთ, რომ **switch** შეტყობინების გამოყენებით ჩვენი პროგრამა, რომელიც უმარტივეს კალკულატორს ქმნის, არ გამარტივებულა, ის გახდა უფრო გასაგები და აქვს შემდეგი სახე:

```
#include<iostream>
using namespace std;
int main()
{
   int result = 0;  // გამოთვლის შედეგი
   char operat;     // ოპერაციის ნიშანი
   int value;       // ოპერაციაში მონაწილე რიცხვი
   while (1) {
      cout<<"Result: "<<result<<endl;
      cout<<"Enter operator and number: ";
      cin >> operat >> value;
      if((operat == 'q') || (operat =='Q'))
         break;
      switch (operat){
        case '+':
           result += value;
           break;
        case '-':
           result -= value;
           break;
        case '*':
           result *= value;
           break;
        case '/':
           if ( value ==0 ){
              cout<<" Error: Divide by zero \n ";
              cout<<" operation ignored \n ";
           }
           else
              result /= value;
           break;
        default:
           cout<<" Unknown operator "<<operat<<endl;
           break;
     } // of switch
  } // of while
  system("pause");
  return 0;
}
```

switch –ის შიგნით გამოყენებული **break** შეტყობინება წყვეტს (ანუ ამთავრებს) **switch** შეტყობინებას. თუ არ წერია **break**, მაშინ შესრულდება რიგის მიხედვით შემდეგი შეტყობინებები. მაგალითად,

```
int control=0;
/* ასეთ დაპროგრამებას კარგი არ ეთქმის */
   switch (control) {
     case 0:
        cout<<"Morning\n";
     case 1:
        cout<<"Noon\n";
        break;
     case 2:
        cout<<"Evening\n";
   }
```

იმ შემთხვევაში, თუ `control == 0`, პროგრამა დაბეჭდავს:

```
Morning
Noon
```

case 0: შემთხვევა არ დამთავრდა **break** შეტყობინებით, ამიტომ morning-ის დაბეჭდვის შემდეგ შესრულდა შემდეგი შეტყობინებაც (**case** 1:), შედეგად დაიბეჭდა noon.
პრობლემა სინტაქსში მდგომარეობს. როცა პროგრამისტს ავიწყდება **break** შეტყობინებით ერთ-ერთი **case** –ის დამთავრება, მაშინ შესრულდება მომდევნო შეტყობინებები, ამ შემთხვევაში **case** 1. როცა პროგრამისტი გამიზნულად გამოტოვებს **break** შეტყობინებას ორი ან მეტი **case**–ის გაერთიანების მიზნით, მაშინ აუცილებლად უნდა გააკეთოს შესაბამისი კომენტარი /* სრულდება მომდევნო შეტყობინებაც */. ანუ, განხილულ მაგალითში **case** 0: მიიღებს შემდეგ სახეს:

```
case 0:
   cout<<"Morning\n";
   /* სრულდება მომდევნო შეტყობინებაც */
```

ვინაიდან **case** 2: ბოლო შტოა, ამიტომ აქ უკვე შეიძლება **break** შეტყობინების გამოტოვება.
და ბოლოს, ისმის კითხვა, რა ხდება, როცა control = 5? ვინაიდან **switch** -ს არ აქვს შტო **default**, ამიტომ ის უბრალოდ გაატარებს ამ შემთხვევას და არაფერი არ შესრულდება.
პროგრამისტმა ყოველთვის უნდა ჩართოს **default** შტო, ვინაიდან ცვლადმა control, გარდა მნიშვნელობებისა 0,1,2, შესაძლოა სხვა მნიშვნელობებიც მიიღოს. ყოველივე ზემოთქმულიდან გამომდინარე, მაგალითი მიიღებს შემდეგ სახეს:

```
int control=5;
  /* უკეთესი ვერსია */
  switch (control) {
    case 0:
       cout<<"Morning\n";
          /* სრულდება მომდევნო შეტყობინებაც */
    case 1:
       cout<<"Noon\n";
       break;
    case 2:
       cout<<"Evening\n";
        break;
    default:
          cout<<"Internal error, control value "
               <<control<<" impossible\n";
       break;
  }
```

მაშინაც კი, როცა ნამდვილად ვიცით, რომ **default** არ არის აუცილებელი, მაინც უნდა ჩავრთოთ **switch** -ში, თუნდაც ასე:

```
default:
/*საერთოდ  არაფერი სრულდება */
break;
```

თავი 5: ნამდვილი რიცხვები და განმეორების შეტყობინება

- ნამდვილი რიცხვები
- შემოკლებული ოპერატორები
- ჩრდილოვანი ეფექტები
- `++x` თუ `x++`
- განმეორების(`looping`) შეტყობინება
- **`while`** შეტყობინება
- **`continue`** შეტყობინება
- **`break`** შეტყობინება
- გვერდითი ეფექტები

ნამდვილი რიცხვები

ნამდვილმა რიცხვმა შეიძლება მთელი მნიშვნელობაც მიიღოს. მაგრამ თუ ერთ-ერთი ნამდვილი ტიპის (მაგ. **`float, double`**) ცვლადი მიიღებს მთელ მნიშვნელობას, კომპიუტერში მისი წარმოდგენა მაინც განსხვავებულია იგივე რიცხვითი სიდიდის მქონე მთელი ტიპის ცვლადის წარმოდგენისგან. ნამდვილი რიცხვების წარმოდგენის სპეციფიკის გამო, პროგრამირებაში ისინი უფრო ხშირად მოიხსენიება როგორც **მცოცავწერტილიანი** რიცხვები. მაგალითად, 5.5, 8.3, -12.6 არის მცოცავწერტილიანი რიცხვები. მცოცავწერტილიანი რიცხვების და მთელი რიცხვების გასარჩევად C++ იყენებს ათობით წერტილს. ასე, 5.0 არის მცოცავწერტილიანი რიცხვი, მაშინ როცა 5 არის მთელი. მცოცავწერტილიანი რიცხვი აუცილებლად შეიცავს ათობით წერტილს. მაგალითად: 3.14, 0.7, 5.47 და ა.შ.

თუმცა დაშვებულია ციფრი 0-ის გამოტოვება წერტილის წინ ან შემდეგ, მაინც რეკომენდებულია მათი სრული სახით ჩაწერა. მაგალითად, .51-ის ნაცვლად უმჯობესია ვწეროთ 0.51.

C++-ში რიცხვი შეიძლება შეიცავდეს ექსპონენტას. მაგალითად, `1.2e34` (ანუ `1.2 x` 10^{34}).

ჩვენ ძირითადად გამოვიყენებთ ორმაგი სიზუსტის მცოცავწერტილიანი ტიპის რიცხვებს. ეს ყველაზე გავრცელებული ტიპია და ასე აღიწერება:

`double` *ცვლადი;* *// კომენტარი*

მცოცავწერტილიანი რიცხვების დიაპაზონი დამოკიდებულია კომპიუტერზე.

`double` ტიპზე გადატვირთულია შეტანა გამოტანის ოპერატორები (<<, >>), არითმეტიკული ოპერატორების ნაწილი (ნაშთის აღების ოპერატორი % აღარ გვაქვს), მისამართის აღების, ზომის განსაზღვრის და ბევრი სხვა ოპერატორი. ეს ძალიან ამარტივებს კოდს, მაგრამ უნდა გვახსოვდეს, რომ ერთი და იგივე ოპერატორები სხვადასხვა ტიპის მონაცემებზე სხვადასხვანაირად მოქმედებენ (ანუ სხვადასხვა ალგორითმის საფუძველზე არიან აგებული). მაგალითად განვიხილოთ **რიცხვების გაყოფა.**

მთელი რიცხვების გაყოფა განსხვავდება მცოცავწერტილიანი რიცხვების გაყოფისგან. მთელი რიცხვების გაყოფისას მიიღება მთელი რიცხვი, წილადი ნაწილი კი იგნორირდება. ასე, რომ 19/10 იგივეა რაც 1.

როცა გასაყოფი და გამყოფი, ან ერთ-ერთი მათგანი მცოცავწერტილიანია, მაშინ შედეგიც მცოცავწერტილიანი რიცხვია. მაგალითად, 19.0/10.0 არის 1.9 (ისევე როგორც 19/10.0 და 19.0/10). რამდენიმე მაგალითი მოტანილია შემდეგ ცხრილში:

გამოსახულება	შედეგი	შედეგის ტიპი
1 + 2	3	მთელი
1.0 + 2	3.0	მცოცავწერტილიანი
19 / 10	1	მთელი
19 / 10.0	1.9	მცოცავწერტილიანი

მე –2 და მე –4 მაგალითში C++ ავტომატურად გარდაქმნის მთელს მცოცავწერტილიან მნიშვნელობად. იგი ანალოგიურ გარდაქმნას ასრულებს, როდესაც ხდება მთელი ტიპის ცვლადზე მცოცავწერტილიანი მნიშვნელობის მინიჭება.

მაგალითად, განვიხილოთ პროგრამის ფრაგმენტი:

```
int main()
{
    int i;                    // მთელი ტიპის ცვლადი სახელით i
    float x;                  // მცოცავწერტილიანი ცვლადი სახელით x
    x = 1.0 / 2.0;            // მიანიჭებს ცვლად x - ს მნიშვნელობას 0.5
    i= 1 / 3;                 // მიანიჭებს ცვლად i - ს მნიშვნელობას 0
    x = (1 / 2) + (1 / 2);    // მიანიჭებს ცვლად x -ს მნიშვნელობას 0.0
    x = 3.0 / 2.0;            // მიანიჭებს ცვლად x-ს მნიშვნელობას 1.5
    i = x;                    // მიანიჭებს ცვლად i-ს მნიშვნელობას 1
    return (0);
}
```

შევნიშნოთ, რომ 1/2 არის მთელი ტიპის გამოსახულება და მისი მნიშვნელობაა 0.

შემოკლებული ოპერატორები

C++–ში არის არაერთი ოპერატორი, რომლის შესრულება ნიშნავს ორი სხვა ოპერატორის ან ინსტრუქციის შესრულებას. როგორც წესი, ამ ოპერატორების უმრავლესობა აგრეთვე გადატვირთულია სხვადასხვა საბაზო ტიპზე. უფრო მეტიც, მომხმარებლის მიერ განსაზღვრულ სხვადასხვა ტიპისთვის (კლასისთვის) ხშირად ძალიან მოსახერხებელია შემოკლებული და სხვა ოპერატორების (ნაწილის მაინც) გადატვირთვა.

მაგალითად, ძალიან ხშირად არის საჭირო რომელიმე ცვლადის, მთელის ან ნამდვილის, (ვთქვათ, counter) მნიშვნელობის ერთით გაზრდა. თუ ამას მინიჭების შეტყობინებით გავაკეთებთ, გვექნება:

```
counter = counter +1;
```

იგივე შედეგს მოგვცემს ++ (მომატების ოპერატორი ანუ ინკრიმენტი) ოპერატორის გამოყენება:

```
++counter;
```

ანალოგიური მოქმედებისაა -- (მოკლების ოპერატორი ანუ დენკრიმენტი) ოპერატორი, რომელიც ერთით ამცირებს ცვლადის მნიშვნელობას. თუ გვინდა არა ერთით, არამედ 3–ით გავზარდოთ ცვლადის მნიშვნელობა, უნდა გამოვიყენოთ += ოპერატორი, ანუ შემდეგი ორი შეტყობინება აკეთებს ერთი და იგივე რამეს:

```
counter = counter + 3;
counter += 3;
```

შემდეგ ცხრილში მოყვანილია რამდენიმე მაგალითი:

ოპერატორები	შეტყობინება	ეკვივალენტური შეტყობინება
+=	x += 2;	x = x + 2;
-=	x -= 2;	x = x - 2;
*=	x *= 2;	x = x * 2;
/=	x /= 2;	x = x / 2;
%=	x %= 2;	x = x % 2;

ჩრდილოვანი ეფექტები

სამწუხაროდ, C++ არ კრძალავს **ჩრდილოვანი ეფექტების** გამოყენებას. ჩრდილოვანი ეფექტი ნიშნავს ოპერაციას, რომელიც სრულდება შეტყობინების ძირითადი ოპერაციასთან ერთად, დამატებით. მაგალითად, შემდეგი კოდი სრულიად კორექტულია:

```
size = 5;
result = ++size;
```

პირველი შეტყობინება მიანიჭებს `size`–ს მნიშვნელობას 5, მეორე მიანიჭებს `result` ცვლადს `size` –ის მნიშვნელობას (ესაა ძირითადი ოპერაცია), მაგრამ დამატებით კიდევ, გაზრდის `size` –ს (ჩრდილოვანი ეფექტი). აქ საკმაოდ ფაქიზი საკითხია იმის გარკვევა, თუ რა მიმდევრობით შესრულდება ეს ოპერაციები, მაგრამ ჩვენ ამის გარკვევას არ დავიწყებთ და არც ჩრდილოვან ეფექტებზე გავაგრძელებთ მსჯელობას, რადგან პროგრამირების კარგი სტილი გულისხმობს, რომ მაქსიმალურად უნდა ავარიდოთ თავი ჩრდილოვანი ეფექტების გამოყენებას.

`++x` თუ `x++`

მომატების ოპერატორის ეს ორი ფორმა ერთნაირად ეფექტურია C++ –ში, და თუ ჩრდილოვან ეფექტებს თავს ავარიდებთ, შედეგიც ერთნაირი აქვთ. მაგრამ პირველი, ე.წ. პრეფიქსული ფორმა `++x` შედარებით ეფექტურია და ამიტომ სჯობს თავიდანვე მას მივანიჭოთ უპირატესობა. ზოგადად, მათი მოქმედების მექანიზმი ასეთია: თუ გამოსახულებაში მონაწილეობს `x++`, მაშინ ჯერ გამოსახულების მნიშვნელობა გამოითვლება და მერე `x` –ის მნიშვნელობას მოემატება ერთი. ამიტომ, შემდეგი ფრაგმენტის შედეგად

```
n = 5;
x = n++;
```

`x = 5`. ამისგან განსხვავებით, შემდეგი ფრაგმენტის შედეგად:

```
n = 5;
x = ++n;
```

გვექნება `x = 6`.

განმეორების (looping) შეტყობინება

განმეორების შეტყობინება საშუალებას იძლევა გავიმეოროთ პროგრამული კოდის ნაწილი რამდენჯერმე ან მანამდე, ვიდრე სრულდება რაიმე პირობა. მაგალითად, განმეორების შეტყობინებები გამოიყენება, როცა ვითვლით დოკუმენტში სიტყვების რაოდენობას, ან სტუდენტთა შორის ფრიადოსნების რაოდენობას.

განმეორების შეტყობინებებია: **`while`**, **`for`** და **`do-while`**.

`while` შეტყობინება

`while` შეტყობინების ზოგადი ფორმა ასეთია:

`while` (*პირობა*)

შეტყობინება;

პროგრამა შეასრულებს **`while`** - ში არსებულ შეტყობინებას რამდენჯერმე, მანამ სანამ *პირობა* გახდება მცდარი (0). თუ პირობა თავიდანვე მცდარია, მაშინ ეს შეტყობინება საერთოდ არ შესრულდება. შესაძლოა, **`while`** –ის პირობის ჭეშმარიტების შემთხვევაში საჭირო გახდეს არა ერთის, არამედ რამოდენიმე შეტყობინების შესრულება. მაშინ შესასრულებელი შეტყობინებები ჩაისმება { } ფრჩხილებში, ანუ ბლოკში. **`while`** –ის ამ ნაწილს ეწოდება ტანი.

განვიხილოთ მაგალითი. ვთქვათ სტუდენტთა ჯგუფმა ჩააბარა გამოცდა. ჯგუფში 6 სტუდენტია. მათი გვარები და შეფასებები შეგვყავს კლავიატურიდან. ვიპოვოთ იმ სტუდენტთა რაოდენობა, რომლებმაც ჩააბარეს გამოცდა (ანუ მიიღეს 51 ქულა ან უფრო მეტი).

შევადგინოთ ალგორითმის ფსევდოკოდი:

1. სტუდენტის გვარის და სტუდენტის ქულის შესანახად გვჭირდება ორი ცვლადი, მაგალითად `name` და `grade`. ქულის შემოწმების შემდეგ ამ ცვლადებს გამოვიყენებთ სხვა სტუდენტისთვის.
2. **`while`** - ის პირობისთვის გვჭირდება მთვლელი (`stud_count`). ვიდრე ეს ცვლადი (0-იდან დაწყებული) 6 -ზე ნაკლებია, **`while`** შეტყობინება რიგრიგობით შეიყვანს სტუდენტების მონაცემებს `name` და `grade` ცვლადებში და შეამოწმებს ქულას.
3. თუ ქულა 51 -ზე მეტია, პროგრამამ ერთით უნდა გაზარდოს კიდევ ერთი ცვლადის (`passed_count`) მნიშვნელობა, რომელიც წარმოადგენს გამოცდაჩაბარებული სტუდენტების მთვლელს და რომლის მნიშვნელობაც თავიდან უნდა იყოს ნულის ტოლი.

ცხადია, ბოლოს პასუხი უნდა დავბეჭდოთ.

C++ -ის შესაბამის პროგრამას აქვს სახე:

```
//////////////////////////////////////////////////////
// პროგრამა grade.cpp
// რომელიც 6 სტუდენტის მონაცემებს მიიღებს კლავიატურიდან
// და დაბეჭდავს გამოცდაჩაბარებულების რაოდენობას
//////////////////////////////////////////////////////
#include <iostream>
#include <string>
using namespace std;

int main()
{
    int stud_count = 0;     // სტუდენტების მთვლელი
    string name;            // ცვლადი გვარისთვის
    int grade;              // ცვლადი ქულისთვის
    int passed_count = 0;   // გამოცდაჩაბარებული  სტუდენტების მთვლელი

    // ვიდრე განხილული სტუდენტების რაოდენობა 6-ს გადააჭარბებს
    while (stud_count < 6)
    {
       // მორიგი სტუდენტის მონაცემი
       cout <<"Enter name"<<endl;
       cin >> name ;

       cout <<"Enter grade"<<endl;
       cin >>grade;

       if(grade > 50)
           ++passed_count;   // გავზარდოთ გამოცდაჩაბარებულების მთვლელი
       stud_count++;         // გავზარდოთ სტუდენტების მთვლელი
    }
    // დავბეჭდოთ გასულების რაოდენობა
    cout <<"number of passed students is " << passed_count <<endl;
    system("PAUSE");
    return 0;
}
```

მაგალითად:

```
Enter name
Arevadze
Enter grade
67
Enter name
Buadze
Enter grade
40
Enter name
Jalali
Enter grade
75
Enter name
Iashvili
Enter grade
100
Enter name
Lejava
Enter grade
50
Enter name
Nikoladze
Enter grade
51
number of passed students is 4
Press any key to continue . . .
```

აქვე განვიხილოთ ამ პროგრამის ერთი განზოგადება, რომელიც საშუალებას გვაძლევს ამოვხსნათ იგივე ამოცანა იმ შემთხვევაშიც, თუ ჯგუფებში სტუდენტთა რაოდენობა ფიქსირებული არაა (სხვადასხვა ჯგუფისთვის სხვადასხვა შეიძლება იყოს). განმეორების შეტყობინება შეიტანს მონაცემებს მანამდე, ვიდრე ჩვენ არ ავკრეფთ კლავიატურიდან კლავიშთა **Ctrl+z** მიმდევრობას, რომელსაც **ფაილის დასასრული** ეწოდება და რომელიც ეკრანზე გამოჩნდება როგორც **^Z.**

იმისათვის, რომ კარნახი ყოველი შესაყვანი სტრიქონის წინ გამოვიდეს, ჩვენ მისი ორჯერ დაწერა მოგვიწევს (სცადეთ დამოუკიდებლად, რომ გააკეთოთ როგორმე სხვანაირად და მიხვდებით რატომ ვწერთ ასე).

```
//////////////////////////////////////////////////////
// პროგრამა grade1.cpp
// რომელიც სტუდენტების მონაცემებს მიიღებს კლავიატურიდან
// და დაბეჭდავს  გამოცდაჩაბარებულების რაოდენობას
//////////////////////////////////////////////////////
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string name;            // ცვლადი გვარისთვის
    int grade;              // ცვლადი ქულისთვის
    int passed_count = 0;   // გამოცდაჩაბარებული  სტუდენტების მთვლელი
    cout <<"Enter name and grade, separated by space"<<endl;
    while (cin >> name >>grade)
    {
       if(grade > 50)
           ++passed_count;
       cout <<"Enter name and grade, separated by space"<<endl;
    }
```

```
      // დავბეჭდოთ გასულების რაოდენობა
    cout <<"number of passed students is " << passed_count <<endl;
    system("PAUSE");
    return 0;
}
```

`continue` შეტყობინება

`continue` შეტყობინება წყვეტს განმეორების შეტყობინების **მიმდინარე ბიჯს**, გამოტოვებს **`continue`**-ს მომდევნო შეტყობინებებს და თავიდან იწყებს განმეორების შეტყობინების **მომდევნო ბიჯის** შესრულებას.

ვთქვათ, ისევ იგივე ამოცანას ვხსნით, მხოლოდ ახლა ცალ-ცალკე დავბეჭდოთ გამოცდაჩაბარებული, ჩაჭრილი და განმეორებით გამსვლელი სტუდენტების გვარები.

შესაბამის C++-პროგრამას აქვს სახე:

```
//////////////////////////////////////////////////////
// continue შეტყობინების გააზრება
// პროგრამა ითვლის და ბეჭდავს გამოცდაჩაბარებული, ჩაჭრილი
// და განმეორებით გამსვლელი სტუდენტების რაოდენობებს
//////////////////////////////////////////////////////
#include <iostream>
#include <string>
using namespace std;

int main()
{
    int stud_count(0);
    int passed_count(0);
    int failed_count(0);
    string name;
    int grade;

    cout <<"Enter name and grade, separated by space"<<endl;
    while (cin >> name >>grade)
    {
       stud_count++;
       if(grade < 41)
       {
          ++failed_count;
          continue;
       }
       if(grade > 50)
          ++passed_count;

       cout <<"Enter name and grade, separated by space"<<endl;
    }
    cout <<"number of passed students is " << passed_count <<endl;
    cout <<"of failed students is " << failed_count <<endl;
    cout <<"meored gasulTa raodenobaa "
         << stud_count - passed_count - failed_count <<endl;
    system("PAUSE");
    return 0;
}
```

პროგრამის მუშაობის შედეგს ჩვენს მონაცემბზე აქვს სახე:

```
number of passed students is 4
of failed students is 1
meored gasulTa raodenobaa 1
```

შევნიშნოთ, რომ ჩვენ შევამცირეთ კომენტარების რაოდენობა, რადგან პროგრამა საკმაოდ მარტივია.

შემდეგი პროგრამა ბეჭდავს სტუდენტების გვარებს და მათ მიერ მიღებულ შეფასებებს, ქულებიდან გამომდინარე.

```
//////////////////////////////////////////////
// continue შეტყობინების გააზრება
// პროგრამა ბეჭდავს სტუდენტების გვარებს და მათ მიერ
// მიღებულ შეფასებებს, ქულებიდან გამომდინარე.
//////////////////////////////////////////////
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string name;
    int grade;
    cout <<"Enter name and grade, separated by space,or Ctrl+z"<<endl;
    while (cin >> name >>grade)
    {
       if(90 < grade ) {
            cout << name << '\t' << 'A'<<endl;
            continue;
       }
       if(80 < grade  && grade < 91 ) {
            cout << name << '\t' << 'B'<<endl;
            continue;
       }
       if(70 < grade  && grade < 81 ) {
            cout << name << '\t' << 'C'<<endl;
            continue;
       }
       if(60 < grade  && grade < 71 ) {
            cout << name << '\t' << 'D'<<endl;
            continue;
       }
       if(50 < grade  && grade < 61 )  {
            cout << name << '\t' << 'E'<<endl;
            continue;
       }
       if(40 < grade  && grade < 51 )  {
            cout << name << '\t' <<"Try again"<<endl;
            continue;
       }
       if(grade < 41 )  {
            cout << name << '\t'<< "failed"<<endl;
            continue;
      }
    }
    system("PAUSE");
    return 0;
}
```

შედეგი ნაჩვენებია შემდეგ სურათზე:

```
Enter name and grade, separated by space,or Ctrl+z
Arevadze 67
Arevadze        D
Buadze 40
Buadze  failed
Jalali 76
Jalali  C
Iashvili 100
Iashvili        A
Lejava 50
Lejava  Try again
Nikoladze 51
Nikoladze       E
^Z
Press any key to continue . . .
```

break შეტყობინება

while-ში გაერთიანებული შეტყობინებების განმეორება მთავრდება, როცა მისი პირობა ხდება მცდარი (0). თუმცა განმეორების შეტყობინება შეგვიძლია შევწყვიტოთ ნებისმიერ ბიჯზე **break** (შეწყვეტა) შეტყობინების გამოყენებით.

შემდეგი მარტივი პროგრამა წარმოადგენს ამ შეტყობინების გამოყენების ნიმუშს.

```
//////////////////////////////////////////////////
// პროგრამა ითვლის [0,99] შუალედში მოთავსებული
// შემთხვევითი რიცხვების ჯამს. შეკრება წყდება,
// როგორც კი შემთხვევითი რიცხვი არის 13-ის ჯერადი.
//////////////////////////////////////////////////
#include <iostream>
using namespace std;
int main()
{
    int s = 0;
    int i;

    while ( true )
    {
        i = rand()%100;
        if (i%13 == 0) break;
         s += i ;
    }
    cout<<"s = " << s <<endl;
    system("PAUSE");
    return 0;
}
```

შემდეგი ამოცანა აგრძელებს "info.txt" ფაილში ჩაწერილი სტუდენტების მონაცემების დამუშავების თემას. ამჯერად ჩვენ გვინდა დავბეჭდოთ ერთ-ერთი ისეთი სტუდენტის მონაცემები, რომელსაც მიღებული აქვს უმაღლესი შეფასება.

```
//////////////////////////////////////////////
// break შეტყობინების გააზრება
// პროგრამა ბეჭდავს სტუდენტის მონაცემებს, რომელსაც
// მიღებული აქვს უმაღლესი შეფასება.
//////////////////////////////////////////////
```

```cpp
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main()
{
    string name;            // სტუდენტის გვარი
    int grade;              // ქულა
    int max_grade = 0;      // მაქსიმალური ქულა
    string max_student;     // მაქსიმალური ქულის მქონის გვარი

    ifstream fin("info.txt"); // ფაილის გახსნა მონაცემების წასაკითხად

    // ვიდრე ფაილში არის წასაკითხი სტრიქონები
    while ( fin >> name >>grade )
    {
        // 100 ქულაზე მეტი არ იქნება, ამიტომ აქ განმეორება შეწყდება
       if( 100 == grade )
       {
          cout<<"students "<< name
              <<" aqvs umaglesi Sefaseba 100 qula " << endl;
          break;
       }
       // თუ 100 ქულა არაა, ჩვეულებრივად ვეძებთ მაქსიმუმს
       if(grade > max_grade)
       {
          max_grade = grade;
          max_student = name;
       }
    }

     fin.close();   // ფაილის დახურვა

    /* თუ 100 ქულიანი არ შეგვხვდა, grade < 100  იქნება,
       ამიტომ მაქსიმალური ქულის მქონის მონაცემებს აქ ვბეჭდავთ */
    if( grade < 100 )
       cout << "students " << max_student << " aqvs umaglesi Sefaseba "
            << max_grade <<" qula " << endl;

    system("pause");
    return 0;
}
```

შედეგი:

info.txt ფაილი	გამოტანის ეკრანი
Arevadze 67 Buadze 40 Jalali 76 Iashvili 100 Lejava 50 Nikoladze 51	students Iashvili aqvs umaglesi Sefaseba 100 qula

გვერდითი ეფექტები

C++ საშუალებას გვაძლევს გამოვიყენოთ მინიჭების შეტყობინება თითქმის ყველგან. მაგალითად, მინიჭების შეტყობინება შეგიძლიათ ჩასვათ სხვა მინიჭების შეტყობინების შიგნით:

```
/* არ დააპროგრამოთ ასე  */
        average = total_value / (number_of_entries = last - first);
```

ეს ექვივალენტურია შემდეგი ორი შეტყობინებებისა:

```
/* დააპროგრამეთ ასე */
        number_of_entries = last - first;
        average = total_value / number_of_entries;
```

პირველი ვერსია “მალავს” `number_of_entries`-ის მინიჭებას გამოსახულების შიგნით. პროგრამა კი ნათელი და მარტივი უნდა იყოს.

C++ იმის საშუალებასაც აძლევს პროგრამისტს, რომ მან მინიჭების შეტყობინება ჩასვას **`while`**–ის პირობაში. მაგალითად:

```
/* არ დააპროგრამოთ ასე */
      while ((current_number = last_number + old_number) < 100)
          cout << "Term " << current_number << endl;
```

ერიდეთ ასეთნაირად დაპროგრამებას. მიაქციეთ ყურადღება, რამდენად კარგად ჩანს ლოგიკა ქვემოთ მოყვანილ ვერსიაში:

```
/* დააპროგრამეთ ასე */
      while (1) {
          current_number = last_number + old_number;
          if (current_number >= 100)
             break;
          cout << "Term " << current_number << endl;
      }
```

თავი 6: ფუნქციები, სტრიქონები (`string`) და ვექტორები (`vector`)

- ფუნქციები
- ობიექტები და მეთოდები
- სტრიქონები (`string`) და ვექტორები (`vector`) - ზოგადი მიმოხილვა
- სტრიქონის (`string`) ობიექტის კონსტრუირება
- მოქმედება სტრიქონებზე

ფუნქციები

ფუნქცია არის პროგრამული კოდის ბლოკი (ანუ ფიგურულ ფრჩხილებს შორის მოთავსებული ფრაგმენტი), რომელსაც მინიჭებული აქვს სახელი. სახელის საშუალებით ხდება ფუნქციის გამოძახება იმდენჯერ, რამდენჯერაც საჭიროა.

პროგრამული კოდის ნებისმიერ ფრაგმენტს ვერ ვაქცევთ ფუნქციად, რადგან აუცილებელია სინტაქსის გარკვეული წესების დაცვა.

მეორე მხრივ, პროგრამული კოდის ნებისმიერი ფრაგმენტი არც უნდა ვაქციოთ ფუნქციად, რადგან ფუნქციის შესაქმნელად აუცილებელია მოტივის არსებობა. აზრი არა აქვს ისეთი ფუნქციის შექმნას, რომელსაც არასოდეს, ან მხოლოდ ერთხელ გამოვიყენებთ. პროგრამული უზრუნველყოფის ინჟინერიაში (კურიკულუმით ეს საგანი გათვალისწინებულია მეშვიდე სემესტრში) არსებობს რამდენიმე პრინციპი, რომელთა გათვალისწინება ხდება ფუნქციის შექმნის დროს. ერთი ძირითადი პრინციპი არის **მრავალჯერ გამოყენებადობა**, ანუ **reusability.** იდეალურ შემთხვევაში, ახალი ფუნქცია თავსდება რომელიმე ბიბლიოთეკაში, რათა მისაწვდომი იყოს მომხმარებლებისთვის. პროგრამული კოდის კარგი ბიბლიოთეკა ბევრია, ზოგი მათგანი არის ღია ლიცენზიით, ზოგი კომერციულით.

სხვა გარემოებები, რაც უნდა გავითვალისწინოთ ფუნქციის შექმნისას, არის: კარგი სტილი, კოდის გარჩევის და გადაკეთების სიადვილე, თვითონ ფუნქციის ზომა (მიზანშეწონილად არ ითვლება ძალიან გრძელი ფუნქციების შექმნა) და სხვა.

ფუნქცია ინდენად ფუნდამენტური ცნებაა პროგრამირებაში, რომ მუდმივად ხდება მისი განვითარება პროგრამირების ენებში, შესაბამისად, ნებისმიერი პროგრამისტი ერთი მხრივ მუდმივად იძენს გამოცდილებას, მეორე მხრივ კი მუდმივად ითვისებს ახალ-ახალ საშუალებებს, რასაც ახალი კომპილერები სთავაზობს. ამიტომ დავიწყოთ უმარტივესით, რაც უკვე ვიცით ოდნავ სხვა ტერმინებში.

ჩვენ უკვე ვისაუბრეთ აღნიშვნებზე. მაგალითად, მათემატიკური $h : R \to Z$ ჩანაწერი C++ ენაში იღებს (კომპილერისთვის) განაცხადის სახეს:

```
int h(double);
```

თუმცა, რეალურ პროგრამირებაში არავინ იყენებს ერთსიმბოლოიან სახელს ფუნქციისთვის, რადგან ეს ეწინააღმდეგება კარგ სტილს. როგორც ვხედავთ ფუნქციის აღწერა გადადის განაცხადში (პროგრამირებაშიც ვიყენებთ ტერმინს აღწერა, იგივე დანიშნულებით რაც მათემატიკაში). ახლა ვნახოთ როგორ სახეს მიიღებს ფუნქციის განსაზღვრა. მაგალითად, თუ $f : R \to R$ წარმოადგენს კვადრატულ სამწევრს $f(x) = 5x^2 + 11x - 2$, ამ ფუნქციაზე განაცხადი და მისი განსაზღვრა ერთად C++ ენაში იღებს სახეს:

```
double f(double x)
{
    return 5*x*x+11*x-2;
}
```

და მას ეწოდება ფუნქციის განხორციელება ანუ **იმპლემენტაცია.**

მაგალითი ძალიან მარტივია, მაგრამ საკმაოდ ზოგადიც: თუ ფუნქცია ითვლის რაიმე გამოსახულების მნიშვნელობას, შეგვიძლია მექანიკურად დავიმახსოვროთ, რომ ტოლობის ნიშნის როლს ასრულებს რეზერვირებული სიტყვა **return**. რეკომენდებულია, რომ განაცხადი ფუნქციაზე, ანუ ფუნქციის **პროტოტიპი** მოთავსებული იყოს **main()** ფუნქციის წინ, ხოლო განაცხადი და განსაზღვრა ერთად - **main()**-ის შემდეგ. ან, რაც უფრო პროფესიონალურია, პროტოტიპები მოთავსებული იყოს ცალკე header ფაილში, ხოლო იმპლემენტაციები - იგივე სახელის მქონე .cpp ფაილში.

განხილული მაგალითი ძალიან მარტივი და არაბუნებრივი იყო, რადგან კონკრეტულად ამ სახის ფუნქციაზე მოთხოვნილება ალბათ არავის გაუჩნდება სხვა დროს. განვიხილოთ სხვა, შედარებით რეალისტური ამოცანა. შევქმნათ ფუნქცია, რომელიც ყოველი ნამდვილი რიცხვისთვის გამოთვლის მის ნიშანს. თუ ფუნქციას დავარქმევთ sign სახელს, მათემატიკურად მისი აღწერა ასეთი იქნება:

$$sign : R \to Z, \qquad sign(x) = \begin{cases} 0, & თუ \quad x = 0, \\ 1. & თუ \quad x > 0, \\ -1, & თუ \quad x < 0. \end{cases}$$

ფუნქციის პროტოტიპი არის:

```
int sign(double);
```

ხოლო იმპლემენტაციაა:

```
int sign (double x)
{
    if(x > 0)
        return 1;
    if(x < 0)
        return -1;
    return 0;
}
```

ყურადღება მივაქციოთ რამდენიმე გარემოებას:

- მათემატიკურ აღწერაში სულ ერთია რომელი შემთხვევა იქნება პირველი, რომელი მეორე და რომელი მესამე. მაგრამ განხორციელებაში (იმპლემენტაციაში) პირველ რიგში უნდა გავითვალისწინოთ ყველაზე ალბათური შემთხვევა, მერე ნაკლებ ალბათური და ა. შ.
- ფუნქციების შიგნით **if-else** კონსტრუქცია ძალიან სპეციფიკურად მუშაობს, რადგან ფუნქციის შიგნით პირველივე **return** ამთავრებს ფუნქციას. ამიტომ, თუ შესრულდა ისეთი **if** -ის პირობა, რასაც მოყვება **return**-ოპერატორი, მაშინ **else** -ის გამოყენება უკვე ზედმეტია, რადგან ფუნქცია დაასრულებს მუშაობას. ამიტომ აქვს ჩვენ მაგალითს ასეთი სახე.

ახლა განვიხილოთ მარტივი არამათემატიკური ფუნქციის მაგალითი, რომელსაც არგუმენტი არ სჭირდება. ვთქვათ, გვინდა გვქონდეს ფუნქცია, რომელიც საჭიროების შემთხვევაში დაბეჭდავს გზავნილს შეცდომის შესახებ. მის იმპლემენტაციას შესაძლოა ჰქონდეს ასეთი სახე:

```
void errorMessage (void)
{
    std::cout << "Error!" << std::endl;
}
```

მოკლედ შევეხოთ ამ ფუნქციების გამოყენების საკითხს. ვთქვათ, ამ სამივე ფუნქციის პროტოტიპი მოყვანილია **main()** ფუნქციის წინ, ხოლო იმპლემენტაციები მის შემდეგ. მაშინ:

```
    std::cout << f(1.5) << std::endl;
```

დაბეჭდავს კვადრატული სამწევრის მნიშვნელობას როცა არგუმენტის მნიშვნელობაა `1.5`. იგივეს იზამს შემდეგი ფრაგმენტი:

```
double x = 1.5;
std::cout << f(x) << std::endl;
```

შემდეგი ფრაგმენტი:

```
double x = -71.5;
std::cout << sign(x) <<s td::endl;
```

დაბეჭდავს `-71.5`-ის ნიშანს, ანუ `-1` -ს.

ბოლოს, შეტყობინება

```
errorMessage ();
```

დაბეჭდავს მარტივ გზავნილს: `Error!`

დავიმახსოვროთ რამდენიმე მნიშვნელოვანი ფაქტი:

1. ფუნქციის შიგნით სხვა ფუნქციის განსაზღვრა აკრძალულია, თუმცა ფუნქციის შიგნით შეგვიძლია გამოვიძახოთ უკვე არსებული სხვა ფუნქციები;
2. ერთადერთი ფუნქცია, რომელსაც არ იძახებს რომელიმე სხვა ფუნქცია, არის **`main()`.** მისი სახელიც მიგვითითებს, რომ მისგან ხდება სხვა ფუნქციების გაშვება (რომლებმაც შეიძლება კიდევ სხვა ფუნქციები გაუშვან და ა.შ.);
3. თუ **`main()`** ფუნქციის წინ მოვათავსებთ რამდენიმე ფუნქციის იმპლემენტაციას, მაშინ მოგვიწევს ანგარიში გავუწიოთ, რომ რომელიმე ფუნქციის იმპლემენტაციაში არ იყოს ისეთი ფუნქციის გამოძახება, რომელიც მის წინ არაა იმპლემენტირებული.

ფუნქციების შესწავლა გრძელდება მთელი სიცოცხლის განმავლობაში, უფრო კონკრეტულად, ყველა მომდევნო მეცადინეობაზე მეტ-ნაკლები ინტრენსივობით შევეხებით ამ თემას.

ობიექტები და მეთოდები

ობიექტის განმარტება ძალიან გავს ცვლადისას. ობიექტი არის ადგილი კომპიუტერის მეხსიერებაში, რომელიც განკუთვნილია გარკვეული ტიპის მონაცემების შესანახად. მეხსიერება გამოიყოფა განაცხადის გაკეთების მომენტში და ობიექტი იგივდება იმ სახელთან, რომელიც განაცხადშია მითითებული. განსხვავება ცვლადთან იმაში მდგომარეობს, რომ ერთი და იგივე ტიპის ობიექტებს აქვთ შესაძლებლობა, რომ გამოიძახონ სპეციფიკური ფუნქციები, რომლებიც მხოლოდ ამ ტიპის ობიექტებზეა განსაზღვრული, ან ამ ტიპის სპეციფიკის გათვალისწინებით არის იმპლემენტირებული.

განსხვავდება გამოძახების ფორმატიც. მაგალითად, ფუნქცია **`sizeof()`** განსაზღვრულია ნებისმიერი ტიპის `x` ცვლადისა და ობიექტისთვის, და მისი გამოძახება ხდება **`sizeof`**`(x)` სახით. ამისგან განსხვავებით, რამდენიმე ტიპის ობიექტისთვის არსებობს ფუნქცია **`size()`**, რომელიც ზომავს ამ ტიპის `x` ობიექტს რაღაც აზრით (მაგალითად, რამდენი რიცხვი წერია ვექტორში). მაგრამ, რადგან როგორც `x` ობიექტი, ასევე **`size()`** ფუნქცია ეკუთვნის ერთი და იგივე ტიპს, ამიტომ `x` ობიექტისთვის **`size()`** ფუნქციის გამოძახება მოხდება **`x.size()`** სახით.

C++ ენაში ჩაშენებული ტიპების გარდა გათვალისწინებულია ახალი ტიპების შექმნა, რაც კეთდება **კლასის** საშუალებით. ამ საკითხს დიდ დროს დავუთმობთ შემდეგში. ამჯერად საკმარისია დავიმახსოვროთ, რომ ტიპის კონკრეტულ ეგზემპლარს, რომლისთვისაც გამოიყოფა მეხსიერებაში ეგზემპლარის, სახელთან გაიგივებული ადგილი, ეწოდება **ობიექტი**, ხოლო სპეციალურად ამ კონკრეტული ტიპისთვის შექმნილ ფუნქციას ეწოდება **მეთოდი**.

ობიექტის ცნება ძალიან მნიშვნელოვანია ობიექტზე ორიენტირებულ პროგრამირებაში და მას თანდათან უფრო მეტი მნიშვნელობა ენიჭება. მაგალითად, C++ ენის იმ გაფართოებულ დიალექტში, რომელსაც იყენებს `Visual Studio`, და რომელსაც ჰქვია `CLR/C++`, ყველაფერი - მუდმივები, მარტივი და შედგენილი ტიპის ცვლადები და ა. შ. წარმოადგენენ ობიექტებს,

რადგან ყველა ტიპს გააჩნია სტანდარტული ფუნქციების გარკვეული მარაგი. ამ ფუნქციების ნაწილს აქვს ერთი და იგივე სახელი, მაგრამ მორგებული არის ამ კონკრეტულ ტიპზე თავისი იმპლემენტაციით. მაგალითად, ამ ენაში თუ გვინდა რიცხვი `5.5` გარდავქმნათ სტრიქონად, უნდა გამოვიძახოთ შესაბამისი მეთოდი და დავწეროთ: `(5.5).ToString()`. C++ ენის სტანდარტული ვერსია იცავს ბალანს, მარტივი ტიპებისთვის მეთოდები განსაზღვრული (ყოველ შემთხვევაში ცხადი სახით) არ არის.

სტრიქონი (`string`) და ვექტორი (`vector`) – ზოგადი მიმოხილვა.

სტრიქონი (`string`) წარმოადგენს მონაცემთა ერთ-ერთ მნიშვნელოვან ტიპს C++ ენაში, რომელიც შექმნილია როგორც კლასი. ამიტომ ხშირად უპირატესობას მივანიჭებთ დაკონკრეტებას და ვიტყვით "`string` კლასი", "`string` ტიპი"-ს ნაცვლად.

`string` კლასი შეიცავს ბევრ ფუნქციას (მეთოდს), რაც საშუალებას გვაძლევს ვიმუშაოთ სტრიქონებთან იმის მსგავსად, როგორც ვმუშაობთ ენის ჩაშენებულ (**`int`, `double`, `char`**) ტიპებთან. მაგალითად, შეგვიძლია გამოვიყენოთ სტრიქონებთან ზოგიერთი ოპერატორი: `=` (მინიჭების), `==` (ტოლობაზე შედარების) , `>` (მეტობის), `+` (შერწყმის ან კონკატენაციის), `+=` (კონკატენაციისა და მინიჭების) და სხვა.

`string` ტიპის მონაცემი `string` კლასის ობიექტს წარმოადგენს. ამიტომ პროგრამაში შექმნილი ყოველი სტრიქონისთვის მისაწვდომია კლასში განსაზღვრული ფუნქციები. მაგალითად,

```
string S ("This is a test string");
```

განაცხადის შემდეგ პროგრამაში შეგვიძლია გავარკვიოთ `S` სტრიქონის სიგრძე `size()`ფუნქციის გამოძახებით: `S.size()`. იგივე მოქმედების არის ფუნქცია `length()`. პროგრამის ორივე შეტყობინება

```
cout << S.size() << endl;
cout << S.length() << endl;
```

დაბეჭდავს S სტრიქონის მიმდინარე სიგრძეს (21 –ს).

სტრიქონისთვის მეხსიერების განაწილება ხდება დინამიკურად მისი შექმნის მომენტში. მაგალითად, `Visual C++` გარემოში ცარიელ სტრიქონს თავიდანვე გამოეყოფა 15 ბაიტი, ხოლო თუ სტრიქონის სიგრძეს გავზრდით (მაგალითად სხვა სტრიქონის დამატების - კონკატენაციის ხარჯზე), მეხსიერება ორმაგდება იმდენჯერ, რამდენჯერაც საჭიროა.

სრტიქონის სიგრძე შეიძლება შეიცვალოს პროგრამის განმავლობაში. მაგალითად, ფრაგმენტი

```
string K;
K = "First string";
cout << K.size() << endl;
K += " and Second string";
cout << K.size() << endl;
K = "Short";
cout << K.size() << endl;
```

დაბეჭდავს

```
12
30
5
Press any key to continue . . .
```

სტრიქონის თითოეულ სიმბოლოზე მიმართვა (წვდომა) ხდება ე.წ. ინდექსის მეშვეობით, რომელიც არის კვადრატულ ფრჩხილებში ჩაწერილი არაუარყოფითი მთელი რიცხვი. ინდექსის მნიშვნელობის ათვლა იწყება ნულიდან. მაგალითად, K სტრიქონისთვის `K[0]` არის სიმბოლო `'S'`, `K[1]` – სიმბოლო `'h'`, `K[2]` –`'o'`, `K[3]` –`'r'` და `K[4]` – სიმბოლო `'t'`. ე.ი. K

სტრიქონის ბოლო სიმბოლოს ინდექსი არის `K.size()-1` –ის ტოლი, ხოლო თვითონ ბოლო სიმბოლო იქნება `K[K.size()-1]`.

ინგლისურენოვან ლიტერატურაში, ინდექსის ნაცვლად ხშირად იყენებენ ტერმინს "`offset`"-გადაწევა. თუ `offset` ნულია, ეს ნიშნავს რომ გვაინტერესებს პირველივე სიმბოლო, რითიც სტრიქონი იწყება. თუ `offset` ერთია, ე.ი. ერთის შემდეგზეა ლაპარაკი და ა.შ.

ინდექსების სწორად შერჩევა პროგრამისტის პასუხისმგებლობაა. საზღვრიდან გასულმა ინდექსმა შეიძლება გამოიწვიოს გაუთვალისწინებელი შედეგი. მაგალითად, წინა ფრაგმენტში რომ დავამატოთ

```
K [10] = 'A';
```

შეტყობინება, პროგრამის შესრულება ავარიულად შეწყდება, ხოლო `Debug` რეჯიმში შეცდომა ასე აიხსნება – `vector subscript out of range`.

სიმბოლოებზე წვდომისათვის `string` კლასი ასევე უზრუნველყოფს ფუნქციას `at` შემდეგი ფორმატით: `at( <ინდექსი> )`. მაგალითად, იგივე K სტრიქონისთვის

```
cout << K.at(4) << endl;
```

შეტყობინება დაბეჭდავს სიმბოლოს `t`, ხოლო

```
K.at(1) = 'p';
cout << K << endl;
```

ფრაგმენტი დაბეჭდავს გარდაქმნილ K სტრიქონს – `Sport`.

უნდა გვახსოვდეს, რომ პროგრამაში, რომელიც იყენებს `string` კლასს აუცილებელია `#include <string>` ბრძანების ჩართვა.

C++ –ის სტანდარტული ბიბლიოთეკის კიდევ ერთი მნიშვნელოვანი კლასია `vector`.

ვექტორი (`vector`) არის მეხსიერებაში თანმიმდევრობით განთავსებული ერთი და იგივე ტიპის მონაცემების (ერთი და იგივე სიგრძის მონაკვეთების) სიმრავლე. პროგრამაში შეგვიძლია შევქმნათ მთელი რიცხვების ვექტორი, ნამდვილი რიცხვების ვექტორი, სტრიქონების ვექტორი და ა.შ. ვექტორში განთავსებულ მონაცემებს მისი ელემენტები ეწოდება.

```
vector<int> V;
```

განაცხადის საფუძველზე იქმნება ცარიელი ვექტორი, რომლის ზომას გავიგებთ `size()` ფუნქციის გამოყენებით. მაგალითად,

```
cout << V.size() << endl;
```

ჩვენ შემთხვევაში დაბეჭდავს 0 –ს.

ვექტორის შევსება ხდება დინამიკურად, მონაცემის დამატებით მის ბოლოში. ელემენტის დამატება სრულდება `push_back( <ელემენტი> )` ფუნქციის მეშვეობით:

```
V.push_back(100);
```

შეტყობინება ჩასვავს ვექტორში მთელ რიცხვს 100, და ვექტორის ზომა გახდება 1 –ის ტოლი.

```
vector<double> Vec(2);
```

განაცხადი შექმნის ნამდვილ რიცხვთა 2 ელემენტიან ვექტორს და ჩასვავს გამოყოფილ მეხსიერებაში ნულებს (გაანულებს ელემენტებს).

მსგავსად სტრიქონისა, ვექტორის ელემენტებზე მიმართვისთვის ვიყენებთ ინდექსებს ან `at` ფუნქციას. მაგალითად, შემდეგი პროგრამის

```
#include <iostream>
#include <vector>
```

```
using namespace std;
int main(){
  vector<double> Vec(2);
  cout<<"size ="<<Vec.size()<<endl;
  cout<<"first element is: "<<Vec.at(0)<<endl;
  cout<<"second element is: "<<Vec.at(1)<<endl;
  Vec[0] = 0.025;
  Vec[1] = -87.65;
  cout<<"\nAfter assignment\nsize ="<<Vec.size()<<endl;
  cout<<"first element is: "<<Vec[0] <<endl;
  cout<<"second element is: "<<Vec[1]<<endl;
  return 0;
}
```

შესრულების შედეგია

```
size =2
first element is: 0
second element is: 0

After assignment
size =2
first element is: 0.025
second element is: -87.65
Press any key to continue . . .
```

მივაქციოთ ყურადღება `#include<vector>` ბრძანებას. პროგრამაში, რომელიც იყენებს `vector`–ს მისი ჩართვა აუცილებელია.

განვიხილოთ კიდევ ერთი მაგალითი, სადაც შევქმნით სტრიქონების ვექტორს, მასში ჩავწერთ კლავიატურიდან შემოტანილ რამოდენიმე სიტყვას, შემდეგ დავალაგებთ სიტყვებს ზრდადობით და ისე დავბეჭდავთ:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <string>
using namespace std;
int main(){
  vector<string> Sentence;
  string word;
  while(cin>>word)
     Sentence.push_back(word);
  cout << "Number of words = " << Sentence.size() << endl;
  sort(Sentence.begin(), Sentence.end());
  int i =0;
  while(i < Sentence.size()){
     cout<<Sentence[i]<<" ";
     ++i;
  }
  cout << endl;
  return 0;
}
```

პროგრამის შედეგია

```
Paris London Washington Berlin
^Z
Number of words = 4
Berlin London Paris Washington
Press any key to continue . . .
```

სიტყვების დალაგებისთვის პროგრამაში ვისარგებლეთ `sort` ალგორითმით, ამისთვის დაგვჭირდა `#include<algorithm>` ბრძანების ჩართვა.

სტრიქონის (string) ობიექტის კონსტრუირება

ჩვენ ვნახეთ როგორ ხდებოდა რამდენიმე მარტივი ტიპის ცვლადის ინიციალიზება მინიჭების ოპერატორით ან კონსტრუქტორით. მარტივ ტიპებთან შედარებით, კლასს აქვს მრავალფეროვანი საშუალებები იმისთვის, რომ თავისი ახლადშექმნილი ობიექტების მნიშვნელობები განსაზღვროს. ეს კეთდება სპეციალური ფუნქციების, კონსტრუქტორების საშუალებით, რომლების გამოძახებაც ხდება ავტომატურად, განაცხადის გაკეთების მომენტში: თუ ახალშექმნილი ობიექტის სახელს მივუწერთ ფრჩხილებს (და არგუმენტსაც), ავტომატურად ამუშავდება ფუნქცია კონსტრუქტორი, რომელიც არგუმენტში მითითებულ მნიშვნელობას ჩაწერს ახალ ობიექტში.

კონკრეტულ მაგალითებზე, განვიხილოთ string კლასის რამდენიმე კონსტრუქტორი. ზოგიერთ მათგანს, რომლებიც შედარებით მოძველებულია და განკუთვნილია C ენის საშუალებებთან თავსებადობისთვის, არ განვიხილავთ.

```
#include <iostream>
#include <string>
using namespace std;

int main ()
{
    // კონსტრუქტორი არგუმენტების გარეშე:
    string s1;
    s1 = " short line for testing";
    cout << "s1  is: " << s1 << endl;
    // ასლის კონსტრუქტორი
    string s2 (s1);
    cout << "s2  is: " << s2 << endl;
    // პირველი არგუმენტი სტრიქონია,
    // მეორე არგუმენტი წარმოადგენს სიმბოლოების რაოდენობას
    string s3 (s1,7);
    cout << "s3  is: " << s3 << endl;
    // კონსტრუქტორის არგუმენტები:
    // 1 - სტრიქონი
    // 2 - საწყისი პოზიცია
    // 3 - სიმბოლოების რაოდენობა
    string s4 (s1,7,4);
    cout << "s4  is: " << s4 << endl;
    // კონსტრუქტორის არგუმენტები:
    // 1 - სიმბოლოების რაოდენობა
    // 2 - თვითონ ჩასასმელი სიმბოლო
    string s5 (15,'*');
    cout << "s5  is: " << s5 << endl;
    // კონსტრუქტორის არგუმენტები:
    // 1 – begin საწყისი იტერატორი
    // 2 - end იტერატორი
    string s6 (s1.begin(),s1.end()-11);
    cout << "s6  is: " << s6 << endl;
    system("PAUSE");
    return 0;
}
```

OUTPUT:

```
s1  is:  short line for testing
s2  is:  short line for testing
s3  is: line for testing
s4  is: line
s5  is: ***************
s6  is:  short line
Press any key to continue . . .
```

თანამედროვე ტენდენციით, უპარამეტრო კონსტრუქტორების გამოყენება მხოლოდ იმ შემთხვევაშია მიზანშეწონილი, თუ იგი რაიმე მნიშვნელობას ჩაწერს ობიექტში გულისხმობის პრინციპით. ამიტომ,

```
string s1;
s1 = " short line for testing";
```

ფრაგმენტის გამოყენების ნაცვლად უმჯობესია თუ ინიციალიზაციას გავაკეთებთ მინიჭების შეტყობინებით:

```
string s1 = " short line for testing";
```

ან პარამეტრიანი კონსტრუქტორის გამოყენებით:

```
string s1 (" short line for testing");
```

უშუალოდ იტერატორებს ჩვენს კურსში არ განვიხილავთ, მაგრამ განვიხილავთ მასთან ახლოს მდგომ ცნებას - პოინტერს, რაც გარკვეულ წარმოდგენას მოგვცემს იტერატორის მოქმედების მექანიზმზე. გარკვეულ წარმოდგენას ამ საკითხზე ჩვენ შევიქმნით არაერთი მაგალითით, რაც განხილული იქნება ჩვენს კურსში.

მოქმედება სტრიქონზე

განვიხილოთ სტრიქონზე მოქმედების რამდენიმე საილუსტრაციო მაგალითი. უფრო სრულად, ეს მასალა შეგიძლიათ იხილოთ ელექტრონული სწავლების სისტემაში, ნაკვეთში „დამატებითი ინფორმაცია“.

1. სტრიქონების შეტანა და ბეჭდვა. **=, +, +=** ოპერატორების გამოყენება

```
#include <iostream>
#include <string>
using namespace std;
int main(){
   string Name, lastName;
   cout << "\nEnter your name and lastname\n";
   cin >> Name >> lastName;
   string a = "My name";  // სტრიქონის ინიციალიზება
   string b(" is ");      // სტრიქონის ინიციალიზება
   string fullName;
   fullName = a +  b;     // სტრიქონების შერწყმა და მინიჭება
   // სტრიქონთან += ოპერარის გამოყენება
   fullName += Name + " " + lastName;
   cout << fullName + "!" << endl;
   system("pause");
   return 0;
}
```

პროგრამა დაბეჭდავს

```
Enter your name and lastname
Luka Toreli
My name is Luka Toreli!
Press any key to continue . . .
```

2. როდის გამოვიყენოთ სტრიქონის კლავიატურიდან შეტანის ფუნქცია `getline` ?

```
#include <iostream>
#include <string>
using namespace std;
int main(){
   cout << "Enter text\n";
   string words;
```

```
    cin >> words; // კლავიატურიდან შეიტანეთ ტექსტი : I'm successful student
    cout << "\nEntered text is\n";
    cout << words << endl;
    system("pause");
    return 0;
}
```

პროგრამის შესრულების შედეგია:

```
Enter text
I'm successful student

Entered text is
I'm
Press any key to continue . . .
```

შეტანის შეტყობინება `cin >> words;` შეცვალეთ ფუნქციით `getline(cin, words);`

```
#include <iostream>
#include <string>
using namespace std;
int main(){
    cout << "Enter text\n";
    string words;
    getline(cin, words);// შეიტანეთ ტექსტი : I'm successful student
    cout << "\nEntered text is\n";
    cout << words << endl;
    system("pause");
    return 0;
}
```

ახლა პროგრამის შესრულების შედეგია:

```
Enter text
I'm successful student

Entered text is
I'm successful student
Press any key to continue . . .
```

გააკეთეთ სწორი დასკვნები.

3. `append` ფუნქციის გამოყენება

```
#include <iostream>
#include <string>
using namespace std;
int main (){
     string str = "Nobody is perfect";
     string s   = ""; // ცარიელი სტრიქონი
     // s სტრიქონის ბოლოში str სტრიქონის 6 სიმბოლოს
     // დამატება, დაწყებული 0 პოზიციიდან
     s.append(str,0,6);
     cout << "s is : " << s << endl;
     // str-ის ქვესტრიქონის დამატება: მე -6 პოზიციიდან მის ბოლომდე
      // (s -ს დაემატება ' is perfect' )
     s.append(str.begin()+6, str.end());
     cout << "s is : " << s << endl;
     // სამი '!' დამატება
     s.append(3,'!');
     cout << "s is : " << s << endl;
     return 0;
}
```

გამოტანის ეკრანი:

```
s is : Nobody
s is : Nobody is perfect
s is : Nobody is perfect!!!
Press any key to continue . . .
```

4. ფუნქციები empty და erase

ა)

```
#include <iostream>
#include <string>
using namespace std;
int main (){
    string str = "*******";
    // სანამ str არ გახდება ცარიელი
    while ( ! str.empty() )
    {
      cout << str << endl;
    // წავშალოთ str -ის ბოლო სიმბოლო
      str.erase(str.end()-1);
    }
    cout << endl;
    return 0;
}
```

გამოტანის ეკრანი:

```
*******
******
*****
****
***
**
*
Press any key to continue . . .
```

ბ)

```
#include <iostream>
#include <string>
using namespace std;
int main (){
    string str, s;
    char ch = 'A';
    while (ch <= 'Z'){
       str.append(1,ch);
     ch++;
    }
    s = str;
    cout << "str is: " << str << endl;
    cout << "s   is: " << str << endl;

    // წავშალოთ str სტრიქონის პირველი 13 სიმბოლო
    str.erase(0,13);
    cout << "Erased range fron str : " << str << endl;

    // წავშალოთ str სტრიქონის ბოლო 13 სიმბოლო
    //( დაწყებული მე-14 სიმბოლოდან )
    str = s.erase(13,13);
    cout << "Erased range from s   : " << str << endl;
    // წავშალოთ s -ის ერთი სიმბოლო (ინდექსით 1)
    cout << endl << "Erase one, second character from s\n";
    s.erase(s.begin()+1);
    cout << "s     is: " << s << endl;
    // წავშალოთ s -იდან სიმბოლოების ჯგუფი
    // (პირველი ოთხი სიმბოლო)
    s.erase(s.begin(),s.begin()+4);
    cout << "s     is: " << s << endl;
    return 0;
}
```

გამოტანის ეკრანი:

```
str is: ABCDEFGHIJKLMNOPQRSTUVWXYZ
s   is: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Erased range fron str : NOPQRSTUVWXYZ
Erased range from s   : ABCDEFGHIJKLM

Erase one, second character from s
s     is: ACDEFGHIJKLM
s     is: FGHIJKLM
Press any key to continue . . .
```

6. ფუნქცია `find`.

```
#include <iostream>
#include <string>
using namespace std;
int main (){
    string  str("C++ is best language");
    int pos1, pos2;
    // str სტრიქონში ვეძებთ სიტყვას "best"
    pos1 = str.find ("best");
    cout << "Word best is found on position " << pos1+1
         << endl;

    pos2 = str.find ("best",pos1+1);
    cout << "Word best is found on position " << pos2+1
         << endl;

    // ვეძებთ სიმბოლო 'g' -ს პირველ შესვლას
    pos1 = str.find('g');
    cout << "First character 'g' found on position "
         << pos1 << endl;
    return 0;
}
```

გამოტანის ეკრანი:

```
Word best is found on position 8
Word best is found on position 0
First character 'g' found on position 15
Press any key to continue . . .
```

7. ალგორითმი `reverse`.

```
#include <iostream>
#include <string>
using namespace std;
int main (){
    string str = "Programming Language";
    reverse(str.begin(), str.end());
    cout << str << endl;
    system("pause");
    return 0;
}
```

გამოტანის ეკრანი:

```
egaugnaL gnimmargorP
Press any key to continue . . .
```

თავი 7: განმეორების **for** შეტყობინება. ფუნქციები **rand**(), **time**() და **srand**(). მოქმედება ვექტორზე

- **for** შეტყობინება
- **for, break** და **continue**
- **rand, time** და **srand** ფუნქციები
- მოქმედება ვექტორზე
- **do – while** შეტყობინება

for შეტყობინება

for წარმოადგენს განმეორების ერთ-ერთ შეტყობინებას. განმეორებათა რიცხვს თვითონ შეტყობინების სახე განსაზღვრავს. **for** შეტყობინებების ზოგადი სახე ასეთია:

```
for ( ინიციალიზება; პირობა; კორექცია )                    (1)
    შეტყობინება;
```

ინიციალიზების ნაწილში ხდება გამოყენებული ცვლადისთვის (ცვლადებისთვის) საწყისი მნიშვნელობის მინიჭება. ეს ნაწილი სრულდება მხოლოდ ერთხელ **for** –ის შესრულების დაწყებისთანავე. *პირობა* განსაზღვრავს, შესრულდეს თუ არა შეტყობინება (**for** –ის ტანი). ტანის შესრულების შემდეგ სრულდება *კორექციის* ნაწილი, სადაც ხდება ცვლადებისთვის მნიშვნელობების შეცვლა.

(1) ექვივალენტურია შემდეგის:

```
ინიციალიზება;
while (პირობის შემოწმება ) {
    შეტყობინება;
    კორექცია;
}
```

მაგალითად, შემდეგი ფრაგმენტი [15, 25] შუალედის რიცხვების შესაკრებად იყენებს განმეორების **while** შეტყობინებას.

```
int total = 0;        // ყველა რიცხვის ჯამი
int number = 15;      // რიცხვის მიმდინარე მნიშვნელობა
while(number <= 25 ){
     total += number;
     ++number;
}
cout << "The grand total is "<< total << endl;
```

გადავწეროთ ეს ფრაგმენტი განმეორების **for** შეტყობინების გამოყენებით

```
int total = 0;  // ყველა რიცხვის ჯამი
int number;     // რიცხვის მიმდინარე მნიშვნელობა
for(number = 15; number <= 25; ++number)
    total += number;
cout << "The grand total is "<< total << endl;
```

ან უფრო მოკლედაც

```
int total = 0;  // ყველა რიცხვის ჯამი
for(int number = 15; number <= 25; ++number)
    total += number;
cout << "The grand total is "<< total << endl;
```

for არ შესრულდება არცერთხელ, თუ *პირობა* თავიდანვე მცდარია, ხოლო თუ *პირობა* ყოველთვის ჭეშმარიტია, განმეორება არ დასრულდება. მაგალითად

```
for( int n=15; n>9 && 10-2*5; n++ )
   puts("ar shesruldeba");
```

```
for( int n=10; 0 || -100; n++ )
   puts("ar dasruldeba");
```

for, break და continue

for შეტყობინების განმეორება მთავრდება, როდესაც მისი პირობა ხდება მცდარი (0). თუმცა შეგვიძლია შევწყვიტოთ **for** –ის შესრულება ნებისმიერ ბიჯზე **break** შეტყობინების გამოყენებით.

continue შეტყობინება წყვეტს **for** –ის **მიმდინარე ბიჯს**, გამოტოვებს **continue**-ს მომდევნო შეტყობინებებს და თავიდან იწყებს **for** –ის **მომდევნო ბიჯის** შესრულებას.

break და **continue** შეტყობინებების გამოყენება **for** –თან ნაჩვენებია ქვემოთ მოცემულ მაგალითებში:

პროგრამის ფრაგმენტი	შედეგი

`int s = 30;`
`for(int n =7; true; --n){`
`  s -= n;`
`  if(n == 4) break;`
`}`
`cout<<"s = "<<s<<endl;`
	`/* s -ის მნიშვნელობა მცირდება 7 -ით, 6-ით, 5-ით და ბოლოს 4 -ით */`
	`s = 8`

`int n =5, p =1;`
`for( ; n<=11; n++){`
`  if( n%2 == 0) continue;`
`  p *=n;`
`}`
`cout<<"p = "<<p<<endl;`
	`/* p -ს მიენიჭება 5, 7, 9 და 11 (კენტი) რიცხვების ნამრავლი */`
	`p = 3465`

შემდეგი პროგრამა შეკრებს შემთხვევით რიცხვებს, ვიდრე ჯამი არ გახდება 10 –ის ჯერადი, ოღონდ ჯამში არ გაითვალისწინებს 13-ზე დამთავრებულ რიცხვებს:

```
#include <iostream>
using namespace std;

int main (){
    int a;       // შემთხვევითი რიცხვი
    int s =0;    // რიცხვების ჯამი

    for (  ; ;  )  // უსასრულო განმეორება, იგივეა რაც  while(true)
    {
       a = rand();
       if(a%100 == 13)
          continue;
       s += a;
       if(s%10 == 0)
          break;
    }
    cout<< "s = " << s <<'\n';
    return 0;
}
```

პროგრამის შესრულების შედეგია:

```
s = 554810
Press any key to continue . . .
```

ფუნქციები `rand()`,`time()` და `srand()`

C++ პროგრამაში შემთხვევითობის ელემენტის შეტანა შეიძლება `rand()` ფუნქციის გამოყენებით. `rand()` ფუნქციაზე განაცხადს შეიცავს `stdlib.h` თავსართი ფაილი. ფუნქცია აგენერირებს (ქმნის) მთელ ფსევდოშემთხვევით რიცხვს `[0, RAND_MAX]` შუალედიდან. მუდმივი `RAND_MAX` ასევე განსაზღვრულია `stdlib.h` ფაილში როგორც

```
#define RAND_MAX 0x7fff
```

სადაც 16-ობითი მნიშვნელობა `0x7fff` ათობითი `32 767`-ის ტოლია.

ფრაგმენტი

```
int i, k;
for(i = 1; i <= 100; i++) {
   k = rand();
   cout << k << '\t';
   if ( i%10 == 0) cout << endl;
}
```

10 სვეტად დაგვიბეჭდავს 100 შემთხვევით მთელ რიცხვს.

(როგორ უნდა შეიცვალოს `for(i = 0; i < 100; i++)` შეტყობინების ტანი, რომ მივიღოთ იგივე შედეგი?)

აქ `k = rand();` შეტყობინების შესრულება მიანიჭებს `k` ცვლადს რომელიღაცა მთელ რიცხვს `[0, RAND_MAX]` დიაპაზონიდან..

შეგვიძლია მოვახდინოთ შემთხვევითი `k` რიცხვის გენერირება `[0,m]` შეალედიდან, ამისათვის უნდა გამოვიყენოთ `rand()` ფუნქცია შემდეგი სახით: `k = rand()% (m+1);`

თუ კი შემთხვევითი მთელი `k` რიცხვის მიღება გვინდა `[n,m]` დიაპაზონიდან, უნდა დავწეროთ `k = rand()%(m+1-n) + n;`
მაგალითად,

```
#include <iostream>
using namespace std;
int main(){
    int i, k, n, m;
    puts("ShemoitaneT diapazonis sazgvrebi : ");
    cin >> n >> m;
    for(i =1; i <= 10; i++) {
        k = rand()%(m+1-n) + n;
        cout << k << ' ';
    }
    cout << endl;
    return 0;
}
```

პროგრამის შესრულების შესაძლო შედეგია

```
ShemoitaneT diapazonis sazgvrebi :
-5 30
0 30 29 -1 12 23 25 13 29 15
Press any key to continue . . .
```

პროგრამის რამდენჯერმე გაშვება გვიჩვენებს, რომ ყოველთვის ვღებულობთ ზუსტად იგივე რიცხვთა მიმდევრობას. საქმე ისაა, რომ `rand()` ფუნქცია ახდენს ე.წ. ფსევდოშემთხვევითი რიცხვების გენერირებას, მაგრამ `rand()` ფუნქციის ალგორითმი არის დეტერმინირებული იმ აზრით, რომ ამ ფუნქციის ბევრჯერ გამოყენება გვაძლევს ფსევდოშემთხვევითი რიცხვების ერთი და იგივე სერიას:

41 18467 6334 26500 19169 15724 11478 29358 26962 24464 . . . (2)

`rand()` ფუნქციის ამ თვისებით სარგებლობენ პროგრამის გამართვის პროცესში: იმის დასადგენად თუ როგორ აისახება შედეგებზე პროგრამაში შეტანილი ცვლილებები აუცილებელია პროგრამა შესრულდეს ერთი და იმავე მონაცემებზე.

გამართვის პროცესის დასრულების შემდეგ უნდა მოხდეს პროგრამის მოდიფიცირება, რომ სხვადასხვა დროს მისი გაშვებისას მიიღებოდეს შემთხვევით რიცხვთა სხვადასხვა მიმდევრობა. ამას *რანდომიზაციას* უწოდებენ. რანდომიზაციას უზრუნველყოფს სტანდარტული ბიბლიოთეკის `srand()` ფუნქცია შემდეგი ფორმატით:

`srand(` *პარამეტრი* `),`

სადაც *პარამეტრი* **`unsigned`** ტიპის ცვლადი ან მუდმივია. *პარამეტრის* სხვადასხვა მნიშვნელობისათვის იგივე პროგრამაში გამოყენებული `rand()` მოახდენს განსხვავებული ფსევდოშემთხვევითი რიცხვების მიმდევრობების გენერირებას.

`srand` აღწერილია `stdlib.h` ფაილში.

პარამეტრის სახით, ჩვეულებრივ, იყენებენ `time(NULL)`ფუნქციის შედეგს:

```
srand( time(NULL) );
```

`time(NULL)`-ის შედეგი წარმოადგენს მიმდინარე კალენდარულ დროს, გაზომილს წამებში. დროის ათვლა მიმდინარეობს 1970 წლის 1 იანვრიდან.

საინტერესოა ფუნქციის სათაურიც. გავრცელებული ვერსიით, `srand` ნიშნავს შემთხვევითობის დათესვას (`seed rand`), თუმცა, იგი შეიძლება ნიშნავდეს წანაცვლებულ შემთხვევითობას (`shifted rand`) , რადგან რა არგუმენტითაც არ უნდა გავუშვათ ფუნქცია `srand()`, `rand()` ფუნქციის მომდევნო შედეგები მაინც მოგვცემს (2) მიმდევრობას, დაწყებულს რაღაც მომენტიდან (მაგალითად, მე-100-იდან დაწყებული, ან 2 345-ედან და ა.შ.).

`time` ფუნქციით სარგებლობისათვის საჭიროა `#include <ctime>` ბრძანების ჩართვა პროგრამაში.

შემდეგი პროგრამა ქმნის 15 შემთხვევით რიცხვს [-7, 15] შუალედიდან და ბეჭდავს 5 სვეტად.

```
#include <iostream>
#include <ctime>
using namespace std;
int main(){
   int i, k;
   srand( time(NULL) );
   for( i=1; i<=10; i++){
      k = rand()% 23 - 7;
      cout << k << '\t';
      if(i % 5 == 0) cout << endl;
   }
   system("PAUSE");
   return 0;
}
```

პროგრამის შესრულების შედეგები:

ტესტი # 1					ტესტი # 2				
-7	5	1	15	14	10	-6	-3	2	13
13	5	-3	2	5	8	8	-3	-4	9
14	-2	2	9	11	9	-2	2	4	-2
Press any key to continue . . .					Press any key to continue . . .				

შემთხვევითი რიცხვების გენერირება ხშირად გამოიყენება ვექტორებთან მუშაობის დროს – ვექტორის შემთხვევითი მნიშვნელობებით შესავსებად. ეს ხერხი დიდი ზომის ვექტორებთან მომუშავე პროგრამების გამართვის კარგი საშუალებაა.

მოქმედება ვექტორზე

ვექტორში განსათავსებელი საწყისი მონაცემები პროგრამას შეიძლება მიეწოდებოდეს კლავიატურიდან ან ფაილიდან. ვექტორის ხიბლი ის არის, რომ არაა აუცილებელი წინასწარ ვიცოდეთ მასში ჩასაწერი მონაცემების რაოდენობა.

მაგალითად, შემდეგი პროგრამა შეიტანს კლავიატურიდან სტუდენტების გვარებს და ჩაწერს სათანადო ტიპის ვექტორში. შემდეგ დაალაგებს გვარებს ანბანის მიხედვით და ისე დაბეჭდავს. გვარების შეტანა უნდა დავასრულოთ `Ctrl+z` კლავიშების კომბინაციას აკრეფით:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <string>
using namespace std;
int main(){
    vector<string> Surnames;
    string name;
    cout << "Enter Students' Surnames :\n";
    while( cin >> name )
       Surnames.push_back(name);
    sort(Surnames.begin(), Surnames.end());
    cout << "In alphabetical order:\n";
    for(int i=0; i < Surnames.size(); ++i)
        cout << Surnames[i] << endl;
    system("PAUSE");
    return 0;
}
```

პროგრამის შესრულების შედეგია

```
Enter Students' Surnames :
Kiladze  Dondua  Iashvili  Aroshidze  Shengelia
^Z
In alphabetical order:
Aroshidze
Dondua
Iashvili
Kiladze
Shengelia
Press any key to continue . . .
```

პროგრამაში გამოვიყენეთ ალგორითმი `sort(Surnames.begin(),Surnames.end());` რომელმაც დაალაგა `Surnames` ვექტორის ელემენტები ზრდადობით. `sort` ალგორითმის გამოყენებისტვის პროგრამაში უნდა ჩავრთოთ `#include <algorithm>` ბრძანება.

შემდეგ პროგრამას შეჰყავს კლავიატურიდან ნამდვილი რიცხვები (მანამ, სანამ არ შევიტანთ სიმბოლოს ან `Ctrl+z` კლავიშების კომბინაციას) და ათავსებს სათანადო ტიპის ვექტორში. შემდეგ პოულობს და ბეჭდავს ვექტორის უდიდესი და უმცირესი ელემენტების ჯამს.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int main(){
```

```
    vector<double> V;
    double real;
    cout << "Enter real numbers, "
            "at the end enter symbol" << endl;
    while( cin >> real )
       V.push_back( real );
    sort( V.rbegin(), V.rend() );
    double sum;
    sum = V[0] + V[ V.size()- 1 ];
    cout<<"Sum of the maximum and minimum is "
        << sum <<endl;
    system("PAUSE");
    return 0;
}
```

პროგრამის შესრულების შედეგია

```
Enter real numbers, at the end enter symbol
1.2 0.3 12.25 -11.25 3.0 20.5 -10 -5.5 0 2.125
Sum of the maximum and minimum is 9.25
Press any key to continue . . .
```

აქ `sort` ალგორითმი გამოყენებული გვაქვს `sort(V.rbegin(),V.rend());` სახით, რაც უზრუნველყოფს V ვექტორის ელემენტების დალაგებას კლებადობით. ამაში შეგვიძლია დავრწმუნდეთ, თუ დავბეჭდავთ ვექტორს დალაგების შემდეგ:

```
for(int i=0; i < V.size(); i++)
    cout << V[i] << ' ';
```

`vector` კლასში განსაზღვრულია ბევრი სასარგებლო ფუნქცია და ასევე შექმნილია ბევრი სტანდარტული ალგორითმი, რომელიც მუშაობს ვექტორთან. განვიხილოთ რამდენიმე მათგანი:

1. ალგორითმი `count`.

პროგრამა ითვლის, რამდენჯერ გვხვდება ვექტორში რიცხვი 23. ვექტორში 20 შემთხვევითი რიცხვია [10, 30] შუალედიდან.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int main(){
   int number;
   vector<int> V;
   const int N = 20;
   for(int counter = 0;counter < 20;counter++)
      V.push_back( rand()%21 + 10 );
   for(int i=0; i<V.size(); ++i)
       cout<<V[i]<<' ';
   cout<<endl;
   int quantity;
   quantity = count( V.begin(), V.end(), 23);
   cout<<quantity<<endl;
   system("PAUSE");
   return 0;}
```

პროგრამის შესრულების შედეგია

```
30 18 23 29 27 26 22 10 29 30 24 15 23 16 17 18 23 24 28 28
3
Press any key to continue . . .
```

2. ალგორითმი count_if.

პროგრამა ითვლის რამდენი უარყოფითი რიცხვია number.dat ფაილში.

```
#include <iostream>
#include <fstream>
#include <vector>
#include <algorithm>
using namespace std;
bool isNegative(double x)  // ფუნქცია–პრედიკატზე განაცხადი
{                          // და მისი განსაზღვრა
   return x < 0;
}
int main(){
   double number;
   vector<double> Vec;
   ifstream fin("number.dat");
   while( fin >> number )
      Vec.push_back( number );
   int quantity;
   // აქ გამოიყენება ფუნქცია isNegative
   quantity = count_if( Vec.begin(), Vec.end(), isNegative);
   cout<<"In file is "<<quantity
       <<" negative numbers"<<endl;
   system("PAUSE");
   return 0;
}
```

პროგრამის შესრულების შედეგია

numbers.dat ფაილი	გამოტანის ეკრანი
10 -2 3 -34 -91 67 8 19 -100 511 0 -13	In file is 5 negative numbers Press any key to continue . . .

3. ალგორითმი insert.

data.txt ფაილი შეიცავს სიტყვებს. ჩავამატოთ მის დასაწყისში სიტყვა "header"

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
using namespace std;
int main(){
   vector<string> Words;
   string w;
   ifstream fin("data.txt");
   while( fin >> w )
      Words.push_back( w );
   Words.insert(Words.begin(),"header");
   fin.close();
   ofstream fout("data.txt");
   for( int i=0; i < Words.size(); i++ )
       fout << Words[i] <<' ';
   return 0;
}
```

პროგრამის შესრულების შედეგია

data.txt ფაილი	data.txt ფაილი პროგრამის შესრულების შემდეგ
do for while int double	header do for while int double

4. ფუნქცია pop_back().

ვექტორში ნამდვილი რიცხვებია. წაშალეთ ვექტორის ბოლო ელემენტი. რიცხვები ვექტორში შეიტანეთ reals.info ფაილიდან.

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;
void printVector(vector<double> );
int main(){
   vector<double> D;
   double number;
   ifstream ifs("reals.info");
   while( ifs >> number )
      D.push_back( number );
   D.pop_back();
   printVector(D);
   system("PAUSE");
   return 0;
}
void printVector(vector<double> x){
   for(int i=0; i<x.size(); i++)
       cout<<x[i]<<' ';
   cout<<endl;
}
```

პროგრამის შესრულების შედეგია

reals.info ფაილი	გამოტანის ეკრანი
-0.01 5.25 43.75 1.5 -0.823	-0.01 5.25 43.75 1.5 Press any key to continue . . .

5. ალგორითმი erase.

reals.dat ფაილი შეიცავს ნამდვილ რიცხვებს. ჩაწერეთ რიცხვები სათანადო სპეციფიკაციის ვექტორში და შემდეგ წაშალეთ კლავიატურიდან შემოტანილი ნამდვილი რიცხვის ტოლი ვექტორის ელემენტი. თუ ასეთი ელემენტი ვექტორში არ არის, დაბეჭდეთ შესაბამისი გზავნილი.

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;
void printVector(vector<double> );
int main(){
   vector<double> v;
   double number;
   ifstream fin("reals.dat");
   while(fin >> number)
      v.push_back(number);
   printVector(v);
```

```
    double del;
    cout<<"Enter number to delete: ";
    cin>>del;
    int i;
    for(i=0; i<v.size(); i++)
       if(v[i] == del) break;
    if(i == v.size())
       cout<<"vector not contains this number\n";
    else{
       v.erase( v.begin()+ i );
       printVector(v);
    }
    system("PAUSE");
    return 0;
}
void printVector(vector<double> x){
    for(int i=0; i<x.size(); i++)
        cout<<x[i]<<' ';
    cout<<endl;
}
```

პროგრამის შესრულების შედეგია

reals.dat ფაილი	გამოტანის ეკრანი
1.2 -0.03 2.5 0.056 12.45	1.2 -0.03 2.5 0.056 12.45 Enter number to delete: 2.5 1.2 -0.03 0.056 12.45 Press any key to continue . . .

დამატებითი ინფორმაცია `string` და `vector` კლასების ფუნქციების და ალგორითმების შესახებ შეგიძლიათ იხილოთ ინტერნეტ–მისამართზე http://cplusplus.com/

განმეორების შეტყობინება **do – while**

do – while განმეორების შეტყობინების ზოგადი ფორმაა:

```
do
{
   შეტყობინება;
}
while (პირობა);
```

do – while შეტყობინება განსხვავებით **while** შეტყობინებისგან იწყებს განმეორებას უპირობოთ, შეასრულებს განმეორების ერთ ბიჯს მაინც, ხოლო დანარჩენი ბიჯების შესრულება უკვე დამოკიდებულია პირობაზე: თუ პირობა ჭეშმარიტია, განმეორება გრძელდება, თუ კი მცდარია – წყდება. **do – while** -თან **break;** და **continue;** შეტყობინებების გამოყენების წესი იგივეა, რაც **while** –თან და **for**–თან.

როდის სჯობს **do – while** შეტყობინების გამოყენება? ამას ამოცანის პირობა გვიკარნახებს – თუ დარწმუნებული ვართ, რომ განმეორება ერთხელ მაინც უნდა შესრულდეს, შეგვიძლია გავაფორმოთ იგი **do – while** სახით.

თავი 8: ცვლადების ხილვადობის არე და ფუნქციები. ფუნქციის პარამეტრები.

- ხილვადობის არე და მეხსიერების კლასი, სტეკი
- **void** ტიპის ფუნქცია, უპარამეტრო ფუნქცია
- ფუნქციისთვის არგუმენტის გადაცემა მნიშვნელობით (`Pass-by-value`)
- არგუმენტის გადაცემა მისამართით (`Pass-by-reference` , `Pass-by-const reference`)

ხილვადობის არე და მეხსიერების კლასი, სტეკი

თითოეულ ცვლადს ორი ატრიბუტი აქვს: ხილვადობის არე და მეხსიერების კლასი. ცვლადის ხილვადობის არე არის პროგრამის ის ნაწილი, სადაც ცვლადის სახელი (იდენტიფიკატორი) შეიძლება გამოვიყენოთ ცვლადის მნიშვნელობაზე წვდომისათვის. თუ ცვლადის მნიშვნელობა მისაწვდომია პროგრამის ნებისმიერ ნაწილში, მაშინ მას **გლობალური** ეწოდება. მისგან განსხვავებით, **ლოკალური** ცვლადის ხილვადობის არე არის ის ბლოკი, რომელშიც კეთდება მასზე განაცხადი და ამ ბლოკის გარეთ ეს ცვლადი არაა ხელმისაწვდომი. ბლოკი არის კოდის ნაწილი, რომელიც შემოსაზღვრულია ფიგურული ფრჩხილებით (**{}**). შემდეგ პროგრამაში ნაჩვენებია განსხვავება ლოკალურსა და გლობალურ ცვლადებს შორის.

```
#include <iostream>
using namespace std;
int global;      // გლობალური ცვლადი
int main(){
   int local;    // ლოკალური ცვლადი
   global = 10; // აქ გლობალურ ცვლადზე წვდომა გვაქვს
   local = 19;  // ასევე ლოკალურ ცვლადზეც

   {// გაიხსნა ბლოკი
         // განაცხადი ბლოკის ლოკალურ ცვლადზე
         int block_local = 100;
         // აქ წვდომა გვაქვს სამივე ცვლადზე
         cout<< global + local + block_local<<endl;
   }// ბლოკი დაიხურა

   // აქ შეგვიძლია მივმართოთ global და local ცვლადებს
   cout<< global + local<<endl;
   cout<< block_local<<endl; // შეცდომა! აქ ბლოკში აღწერილ ცვლადზე
                             // წვდომა აღარა გვაქვს
   return 0;
}
```

ჩვენ შეგვიძლია რომელიმე შიდა ბლოკში შემოვიღოთ იგივე სახელის მქონე ლოკალური ცვლადი. ამ ბლოკის შიგნით ამ სახელით მხოლოდ ლოკალური ცვლადი იქნება ხილვადი. ამ ბლოკის გარეთ კი იგივე სახელით კვლავ გლობალური ცვლადი ხდება ხილვადი.

გაუგებრობის თავიდან ასაცილებლად რეკომენდებულია, რომ სხვადასხვა ხილვადობის ცვლადებს ერთიდაიგივე სახელი არ დავარქვათ.

მეხსიერების კლასის მიხედვით ცვლადები შეიძლება იყოს ორიდან ერთი: მუდმივად არსებული ან დროებითი. გლობალური ცვლადები ყოველთვის მუდმივად არსებული ცვლადებია. ისინი იქმნება და ინიციალდება პროგრამის შესრულების დაწყებამდე. დროებითი ცვლადები მეხსიერების იმ ნაწილში თავსდება, რომელსაც სტეკი (`stack`) ეწოდება. მრავალი დროებითი ცვლადის გამოყენებამ შეიძლება "სტეკის გადავსების" შეცდომა (`stack overflow`) გამოიწვიოს.

სტეკში დროებითი ცვლადის მიერ დაკავებული ადგილი თავისუფლდება შესაბამისი ბლოკის დასრულების შემდეგ. ბლოკში ყოველი შესვლისას დროებით ცვლადს სტეკში ხელახლა გამოეყოფა ადგილი.

სტეკის გადავსება სხვადასხვა მიზეზით შეიძლება მოხდეს. ერთ შემთხვევაში მას იწვევს მრავალი დროებითი ცვლადი ან დიდი დროებითი ვექტორი, სხვა შემთხვევაში შესაძლოა რომელიმე ფუნქცია დაუსრულებლად იძახებდეს თავის თავს (მაგალითად, ასე ხდება ცუდად ორგანიზებული რეკურსიის შემთხვევაში).

დროებითი ცვლადები გვხვდება როგორც ბლოკებში, ასევე ფუნქციებში. განსაკუთრებით საინტერესო სიტუაციები იქმნება, როდესაც გვხვდება ერთმანეთში ჩაწყობილი ბლოკები, ან, რაც უფრო გავრცელებული შემთხვევაა, ერთი ფუნქცია იძახებს მეორეს, ის სხვას და ა. შ. ამ შემთხვევაში სტეკში ხდება ახალ-ახალი ინფორმაციის ჩატვირთვა (push). როცა რომელიმე ბლოკი დაიხურება ან რომელიმე ფუნქცია დაასრულებს მუშაობას, სტეკიდან "გაქრება" (pop) შესაბამისი ცვლადები და აგრეთვე ზოგიერთი სხვა ინფორმაცია, რასაც მთლიანობაში უწოდებენ სტეკის კადრს.

თავისი მუშაობის პრინციპის გამო, სტეკს ხშირად ადარებენ ერთმანეთზე დადგმული ერთი ზომის თეფშების წყებას, რომელშიც ახალი თეფშის დამატება შესაძლებელია ზემოდან, და რომელიმე თეფშის აღებაც მხოლოდ იმ შემთხვევაში შეიძლება, თუ ის ზემოდან დევს. სტეკშიც ბოლოს დამატებული ინფორმაცია თავსდება სტეკის ბოლოში და წაშლაც ხდება ბოლოდან. მუშაობის ასეთ პრინციპს ეწოდება LIFO(last-in,first-out) - ანუ ბოლოს მოსული პირველი მიდის.

სტეკის სიგრძე დამოკიდებულია ოპერაციულ სისტემაზე და კომპილერზე, რომელსაც თქვენ იყენებთ.

ლოკალური ცვლადი არის დროებითი გარდა იმ შემთხვევისა, როცა მის განაცხადში მითითებულია **static**.

თუ **static** გამოიყენება გლობალურ ცვლადზე განაცხადში, მაშინ მას აქვს სრულიად სხვა დანიშნულება. ის მიუთითებს, რომ ცვლადი არსებობს მხოლოდ მიმდინარე ფაილში.

შემდეგი მაგალითი გვიჩვენებს განსხვავებას მუდმივად არსებულ და დროებით ლოკალურ ცვლადებს შორის. სიმარტივისთვის, temporary არის დროებითი ცვლადი, ხოლო permanent არის მუდმივად არსებული. ორივე ცვლადზე განაცხადი კეთდება **for** შეტყობინების ბლოკის დასაწყისში. **for** შეტყობინების ტანის ყოველი განმეორებისას ხდება temporary ცვლადის ხელახალი შექმნა და ინიციალიზება, ხოლო სტატიკური ცვლადი permanent იქმნება და მისი ინიციალიზება ხდება მხოლოდ ერთხელ **for** -ის პირველი ბიჯის დროს.

```
#include <iostream>
using namespace std;
int main(){
   int counter;      // განმეორების მთვლელი
   for( counter=0; counter<3; ++counter ){
     int temporary = 1;          // დროებითი ცვლადი
     static int permanent = 1;  // მუდმივი ცვლადი
     cout<<"Temprary "<< temporary
         <<" Permanent "<< permanent<<endl;
     ++temporary;
     ++permanent;
   }
   return 0;
}
```

პროგრამის შედეგია:

```
Temprary 1 Permanent 1
Temprary 1 Permanent 2
Temprary 1 Permanent 3
Press any key to continue . . .
```

დროებით ცვლადებს ზოგჯერ ავტომატურ ცვლადებს უწოდებენ, რადგან მათთვის მეხსიერების გამოყოფა ხდება ავტომატურად. სპეციფიკატორი **auto** შეიძლება გამოყენებული იყოს იმის მაჩვენებლად, რომ ცვლადი დროებითია. პრაქტიკაში ეს სპეციფიკატორი არ გამოიყენება.

შემდეგი ცხრილით მოცემულია განაცხადის გაკეთების სხვადასხვა შემთხვევები

განაცხადი გაკეთებულია	მოქმედების არე	კლასი	ინიციალიზება
ყველა ბლოკის გარეთ	გლობალური	მუდმივი	ერთხელ
static, ყველა ბლოკის გარეთ	გლობალური	მუდმივი	ერთხელ
ბლოკის შიგნით	ლოკალური	დროებითი	ბლოკში ყოველი შესვლისას
static, ბლოკის შიგნით	ლოკალური	მუდმივი	ერთხელ

ისევ ფუნქციის შესახებ

აქ თავი მოვუყაროთ ყველა იმ ცოდნას ფუნქციის შესახებ, რაც უკვე გაგვაჩნია.

ფუნქციის იმპლემენტაციის ფორმატი შემდეგია:

<*დასაბრუნებელი ტიპი*> ფუნქციის სახელი (<*პარამეტრების სია*>)
{
განაცხადი ცვლადებზე
შეტყობინებები
}

<დასაბრუნებელი ტიპი> განმარტავს ფუნქციით შექმნილი მნიშვნელობის ტიპს. რადგანაც ფუნქცია განკუთვნილია კონკრეტული ამოცანის ამოსახსნელად, მისი შესრულების შედეგი შეიძლება იყოს რიცხვი, სიმბოლო, სტრიქონი და ა.შ. სწორედ ეს არის ფუნქციის დასაბრუნებელი მნიშვნელობა. როდესაც საქმე მათემატიკურ ფუნქციებს ეხება, დასაბრუნებელი ტიპი იგივეა რაც ფუნქციის მნიშვნელობათა სიმრავლე.

ფუნქციის სახელის საშუალებით კეთდება ფუნქციაზე განაცხადი (მისი პროტოტიპის მოცემა), ფუნქციის განსაზღვრა (შესასრულებელი მოქმედებების აღწერა) და ფუნქციაზე მიმართვა (მისი გამოძახება). კომპილერისთვის ნებისმიერ ობიექტზე განაცხადი ნიშნავს, რომ ეს ობიექტი გაიგივდება მეხსიერების გარკვეულ მონაკვეთთან. ფუნქციების შემთხვევაში, ფუნქციის სახელი გაიგივდება ამ მონაკვეთის დასაწყისთან, რაც ძალიან მოსახერხებელია თუ დავაპირებთ ფუნქციის არგუმენტად სხვა ფუნქციის გამოყენებას. ეს მიდგომა ძალიან განვითარებულია ბევრ თანამედროვე პროგრამირების ენაში. მაგალითად, შემდეგი მარტივი ფუნქციისთვის შეგვიძლია ვნახოთ, რომ მისი მისამართი მართლაც იგივეა რაც მისი სახელი:

```
int f(int x){
    return x*x;
}
int main(){
   cout<<int(&f)<<endl;
   cout<<int(f)<<endl;
   return 0;
}
```

<პარამეტრების სია> წარმოადგენს ფუნქციის პარამეტრების ჩამონათვალს მათი ტიპების მითითებით. ჩამონათვალში პარამეტრები გამოიყოფა მძიმით, ამასთან ტიპი უნდა მიეთითებოდეს ყოველ პარამეტრს. პარამეტრების საშუალებით ფუნქციას გადაეცემა მისთვის საჭირო საწყისი მონაცემები (`main`-იდან ან სხვა ფუნქციიდან). დასაშვებია, რომ პარამეტრების სია იყოს ცარიელი. მაგალითად, მათემატიკური ფუნქციებისთვის, პარამეტრი იგივეა რაც არგუმენტი და თანამედროვე ტენდენციით პროგრამირებაშიც სულ უფრო ხშირად ხდება ტერმინ არგუმენტის გამოყენება პარამეტრის ნაცვლად.

ფიგურულ ფრჩხილებში მოთავსებულია ამოსახსნელი ამოცანის რეალიზებისთვის საჭირო ცვლადებზე განაცხადები და შესრულებადი შეტყობინებები - ფუნქციის ტანი.

ფუნქციის ტანში აღწერილი ცვლადები და ფუნქციის პარამეტრები წარმოადგენენ მის *ლოკალურ* ცვლადებს, რომლებიც ხილვადი (ცნობილი, წვდომადი) არიან მხოლოდ ფუნქციაში.

ვთქვათ, პროგრამაში გვჭირდება ფუნქცია, რომელიც ითვლის ორი მთელი რიცხვის საშუალო არითმეტიკულს. ასეთ ფუნქციაზე განაცხადს ექნება სახე

```
double Average(int x, int y);
```

სადაც

`double` - ფუნქციის ტიპია, ანუ მისი დასაბრუნებელი მნიშვნელობის ტიპი;

`Average` - ფუნქციის სახელია;

`x` და `y` - ფუნქციის პარამეტრებია.

ფუნქციაზე განაცხადს ასევე პროტოტიპი ეწოდება. პროტოტიპის შემდეგ უნდა დავსვათ წერტილ-მძიმე (`;`), წინააღმდეგ შემთხვევაში მივიღებთ კომპილაციის შეცდომას.

პროტოტიპში პარამეტრების სახელების მითითება აუცილებელი არ არის - კომპილერი ამ სახელებს უგულვებელყოფს. მაგალითად,

```
double Average(int , int );
```

პროტოტიპის სწორი ჩანაწერია.

ფუნქციის გააქტიურება, ანუ მასში დაპროგრამებული კოდის შესრულების დაწყება, ხდება მისი გამოძახების გზით. მაგალითად, `Average` ფუნქციის გამოძახების ფრაგმენტი შესაძლოა ასე გამოიყურებოდეს:

```
int a = 287, b = -143;
double answer = Average(a, b);
```

სადაც

`a` და `b` -ს ფუნქციის *არგუმენტებია.*

პროტოტიპის საშუალებით კომპილერი ამოწმებს, რამდენად სწორია ფუნქციაზე მიმართვა. ფუნქციის გამოძახების დროს მისი არგუმენტები (`a` და `b`) უნდა შეესაბამებოდეს პროტოტიპში აღწერილ პარამეტრებს (`x` და `y`): არგუმენტების რაოდენობა, ტიპები და მათი რიგი უნდა ემთხვეოდეს პარამეტრების რიცხვს, თითოეულის ტიპსა და რიგს. წინააღმდეგ შემთხვევაში ფიქსირდება კომპილაციის შეცდომა.

ასევე უნდა ემთხვეოდეს პროტოტიპში მითითებული ფუნქციის ტიპი და `answer` ცვლადის ტიპი. შეუსაბამობის შემთხვევაში კომპილერი ავტომატურად გარდაქმნის ფუნქციის დასაბრუნებელი მნიშვნელობის ტიპს `answer` ცვლადის ტიპზე (რამაც შეიძლება გამოიწვიოს ინფორმაციის დაკარგვა) ან მივიღებთ კომპილაციის შეცდომას: `there is no acceptable conversion`.

სინტაქსური შეცდომა ფიქსირდება მაშინაც, როდესაც შეუსაბამობაა ფუნქციის პროტოტიპსა და ფუნქციის იმპლემენტაციის სათაურს შორის. ორივეში უნდა ემთხვევოდეს როგორც დასაბრუნებელი ტიპი, ისე პარამეტრების რიცხვი და მათი ტიპები.

`Average` ფუნქციის იმპლემენტაციის შესაძლო სახეა:

```
double Average(int x, int y){
   double t; //  x, y და t - ფუნქციის ლოკალური ცვლადებია
   t = (x + y)/ 2. ;
   return t;
}
```

`Average` ფუნქციის ტანის ბოლო ინსტრუქცია

```
return t;
```

წარმოადგენს `return` შეტყობინებას.

თუ ფუნქციამ უნდა დააბრუნოს შედეგი, მაშინ მის ტანში სრულდება შეტყობინება

```
return ( გამოსახულება );
```

რომელშიც ჯერ გამოითვლება გამოსახულების მნიშვნელობა და შემდეგ `return` დააბრუნებს მას ფუნქციაზე მიმართვის წერტილში.

ფუნქცია შეიძლება შეიცავდეს ერთზე მეტ `return` –ს. მაგალითად, როგორც ადრე განხილულ ფუნქციაში `sign`, რომელიც ყოველ ნამდვილ რიცხვს შეუსაბამებდა მის ნიშანს:

```
int sign (double x){
   if( x > 0 )
       return 1;
   if( x < 0 )
       return -1;
   return 0;
}
```

ფუნქციაში სამი `return` შეტყობინებაა, რომლიდანაც სრულდება ერთ–ერთი.

აღვნიშნოთ, რომ `Average` ფუნქცია შეიძლება ჩაიწეროს უფრო მოკლედაც, ლოკალური `t` ცვლადის შემოღების გარეშე:

```
double Average(int x, int y){
   return (x + y)/ 2. ;
}
```

`void` ტიპის ფუნქცია

C++ -ში, განხვავებით მათემატიკისგან, აქტიურად გამოიყენება ისეთი ფუნქციები, რომლებიც არ აბრუნებენ მნიშვნელობას (მაგალითად, ფუნქციის დანიშნულებაა გზავნილის ბეჭდვა, ხატვა, ვექტორში მონაცემების მოთავსება, ვექტორის ელემენტების ბეჭდვა და სხვა). ასეთი ფუნქციები `void` ტიპისაა. მაგალითად, შემდეგი ფუნქცია განკუთვნილია მთელ რიცხვთა ვექტორის ელემენტების დასაბეჭდად:

```
void printVector(vector<int> x)
{
   for(int i=0; i< x.size(); i++)
     cout<<x[i]<<'\t';
   cout<<endl;
}
```

შემდეგ პროგრამაში ნაჩვენებია `printVector` ფუნქციის გამოყენება (გამოძახება). გზად, ნაჩვენებია თუ როგორ ჩავამატოთ რამდენიმე ერთნაირი ელემენტი მოცემულ ადგილზე.

```
#include <iostream>
#include <vector>
using namespace std;
void printVector( vector<int> );
int main()
{
   vector<int> V;
   for(int count=1; count<=10; count++)
      V.push_back( rand() );
   printVector( V );
   V.insert(V.begin() + V.size()/2, 2, 300);
   printVector( V );
   return 0;
}
void printVector( vector<int> x )
{
    for(int i=0; i<x.size(); i++)
       cout<<x[i]<<' ';
    cout<<endl;
}
```

პროგრამის შედეგია:

```
41 18467 6334 26500 19169 15724 11478 29358 26962 24464
41 18467 6334 26500 19169 300 300 15724 11478 29358 26962 24464
Press any key to continue . . .
```

`void` ტიპის ფუნქციის ტანში `return` შეტყობინება ან არ გამოიყენება, ან (თუ საჭიროა ფუნქციის შესრულების შეწყვეტა რაიმე პირობის მიხედვით) გამოიყენება ფორმატით `return;`

ორივე შემთხვევაში ფუნქციის დასაბრუნებელი მნიშვნელობა განსაზღვრული არ არის.

უპარამეტრო ფუნქცია

განხვავებით მათემატიკისგან, უამრავ შემთხვევაში C++ -ის ფუნქცია არ საჭიროებს საწყის მონაცემებს (მას არ გადაეცემა არგუმენტები). ამ ფაქტის აღსანიშნავად ენის სინტაქსი ითვალისწინებს ორ შესაძლებლობას:

- პარამეტრების ცარიელი სია.
  მაგალითად,

  ```
  float Calculation();
  ```

  განაცხადი აღნიშნავს, რომ ფუნქცია `Calculation` დააბრუნებს ნამდვილ რიცხვს, მას არა აქვს პარამეტრები და გამოძახების დროს არ გადაეცემა არგუმენტები.

- მომსახურე `void` სიტყვის გამოყენება.
  `Calculation` ფუნქციაზე ზემოთ მოყვანილის ტოლძალოვანი განაცხადია

  ```
  float Calculation(void);
  ```

  სადაც მრგვალ ფრჩხილებში ჩაწერილი `void` ნიშნავს პარამეტრების ცარიელ სიას.

ქვემოთ მოყვანილ პროგრამაში ფუნქცია `Calculation` ითვლის [217, 426] შუალედში 13-ის ჯერადი რიცხვების საშუალო არითმეტიკულს.

```
#include <iostream>
using namespace std;
float Calculation();
int main()
{
```

```
    float k;
    k = Calculation();
    cout<<k<<endl;
    return 0;
}
float Calculation(){
    int s =0, k =0;
    for(int n = 426/13*13; n >= 217; n -= 13){
        s += n;
        k++;
    }
    return (float)s/k;
}
```

პროგრამა დაბეჭდავს:

```
318.5
Press any key to continue . . .
```

ფუნქციისთვის არგუმენტის გადაცემა მნიშვნელობით (**Pass-by-value**)

არგუმენტის გადაცემის უმარტივესი ხერხია გადავცეთ ფუნქციას არგუმენტის ასლი. ამ შემთხვევაში ფუნქციის გამოძახების დროს მის პარამეტრს ენიჭება იმ მნიშვნელობის ასლი, რომელსაც არგუმენტად მივიჩნებთ. არგუმენტის გადაცემის ამ ხერხს უწოდებენ მნიშვნელობით გადაცემას – `Pass-by-value`.

შემდეგ მაგალითში განისაზღვრება სამ რიცხვს შორის უმცირესის პოვნის ფუნქცია `min3Int`. პროგრამაში შეგვაქვს 3 მთელი რიცხვი. შემდეგ ეს რიცხვები გადაეცემა ფუნქციას `min3Int`, რომელიც დაადგენს მათ შორის უმცირესს. **`return`** შეტყობინება დაუბრუნებს `main`-ს უმცირესის მნიშვნელობას, და ის დაიბეჭდება.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int min3Int(int, int, int); // ფუნქციის პროტოტიპი
int main()
{
    int a, b, c;
    cout<<"ShemoiteneT 3 mTeli ricxvi: ";
    cin>>a>>b>>c;
    cout<<"Minimaluri udris "
        << min3Int(a,b,c) <<endl;
    return 0;
}
// minimum ფუნქციის განსაზღვრა
int min3Int(int x, int y, int z){
    int min = x;
    if (y < min) min = y;
    if (z < min) min = z;
    return min;
}
```

პროგრამის შესრულების შედეგია:

```
ShemoiteneT 3 mTeli ricxvi: 23 7 69
Minimaluri udris 7
Press any key to continue . . .
```

ფუნქციის პროტოტიპი **int** min3Int(**int**, **int**, **int**); გვიჩვენებს, რომ ფუნქცია "ელოდება" 3 მთელ არგუმენტს და დააბრუნებს მთელს.

ფუნქციის გამოძახება ჩვენ მაგალითში ხდება გამოტანის შეტყობინებაში

```
cout<<"Minimaluri udris "<< min3Int(a,b,c)<<endl;
```

და გვიჩვენებს, რომ ფუნქციაში გათვალისწინებული მოქმედებები უნდა შესრულდეს a, b და c არგუმენტებზე – მთელ რიცხვებზე. min3Int(a,b,c) -ის შესრულება შემდეგი ფრაგმენტის შესრულების ეკვივალენტურია:

```
int x = a;  int y = b;  int z = c;
int min =x;
if (y < min) min =y;
if (z < min) min =z;
return min;
```

ე.ი. x პარამეტრს მიენიჭება a არგუმენტის მნიშვნელობა, y პარამეტრს – b-ს მნიშვნელობა, z პარამეტრს – c-ს მნიშვნელობა (ანუ ფუნქციის ლოკალური ცვლადები "დაიჭერენ" მათთვის გამოძახების დროს გადაგზავნილ მნიშვნელობებს), ხოლო ფუნქციის შეტყობინებები, ფაქტობრივად, შესრულდება a, b და c მნიშვნელობებისთვის. უნდა გვახსოვდეს, რომ პარამეტრებსა და არგუმენტებს შორის შესაბამისობა დგინდება მათი რიგის მიხედვით: პირველ პარამეტრს (x -ს) შეესაბამება პირველი არგუმენტი (a), მეორე პარამეტრს (y -ს) - მეორე არგუმენტი (b), ხოლო მესამე პარამეტრს (z -ს) - მესამე არგუმენტი (c).

არგუმენტის გადაცემა მისამართით (Pass-by-reference)

ფუნქციაში არგუმენტების გადაცემის 2 ხერხი არსებობს: არგუმენტების გადაცემა მნიშვნელობით და არგუმენტების გადაცემა მისამართით.

ფუნქციის გამოძახების დროს მის პარამეტრებს და ლოკალურ ცვლადებს გამოეყოფათ ადგილი ოპერატიული მეხსიერების სპეციალურ არეში – სტეკში. როდესაც არგუმენტი გადაეცემა ფუნქციას მნიშვნელობით, ფუნქციის პარამეტრს ენიჭება ამ არგუმენტის ასლი. ამიტომ, თუ ფუნქციის ტანში მისი პარამეტრი იცვლის მნიშვნელობას, შესაბამისი არგუმენტი არ იცვლება.

ხშირად საჭიროა ფუნქციაში გადაცემული არგუმენტის შეცვლა. მაგალითად, როდესაც:

- ფუნქციიდან ერთზე მეტი მნიშვნელობის დაბრუნება გვჭირდება. მაშინ ერთ მნიშვნელობას დააბრუნებს **return** შეტყობინება, დანარჩენი მნიშვნელობების დაბრუნება კი უნდა მოხერხდეს პარამეტრების მეშვეობით.
- პროგრამაში გვჭირდება ფუნქციის მიერ არგუმენტის შეცვლილი მნიშვნელობა.
- ფუნქციას დასამუშავებლად გადაეცემა მონაცემთა დიდი ობიექტი. ასეთი არგუმენტის გადაცემა მნიშვნელობით (მისი კოპირება პარამეტრში) მოითხოვს გარკვეულ დროს, რაც ანელებს პროგრამის შესრულებას.

მსგავს შემთხვევებში ფუნქციას უნდა გადაეცეს არა არგუმენტის ასლი, არამედ მისი მისამართი. მაშინ ყველა ცვლილება, რომელსაც ფუნქციაში განიცდის პარამეტრი, სინამდვილეში ხორციელდება არგუმენტზე. ასეთ პარამეტრებს ეწოდებათ ცვლადი პარამეტრები.

არგუმენტის მისამართით გადაცემის ერთი ხერხია შესაბამისი პარამეტრი აღვწერით ე.წ. reference–ის სახით. ტერმინი reference ხშირად ითარგმნება როგორც მითითება, მეტსახელი ან ფსევდონიმით მიმართვა.

reference არის პროგრამული ობიექტის ფსევდონიმი (alias), ანუ, მარტივად რომ ვთქვათ, იმ პროგრამული ობიექტის მეტსახელი, რომელსაც იგი მიუთითებს.

მაგალითად,

```
int j = 10, k;
```

```
int &i = j;     // განაცხადი  i მითითებაზე (ფსევდონიმზე)
```

განაცხადი აღნიშნავს, რომ `i` ირიბად მიუთითებს `j`-ს, ანუ *მთელი ტიპის* `i` ცვლადის მისამართი იგივეა, რაც მთელი ტიპის `j` ცვლადისა (ე.ი. `i` და `j` - ერთი და იგივე მეხსიერების სახელებია). ცხადია, რომ მითითებაზე (ფსევდონიმზე) განაცხადისთანავე აუცილებლად უნდა მოხდეს მისი ინიციალიზება.

შემდეგი ფრაგმენტი

```
int j = 10, k;
int &i = j;     // განაცხადი  i მითითებაზე (ფსევდონიმზე)
cout<<j<<" "<< i;    //  დაიბეჭდება  10 10
k = 121;
i = k;
//  i-ს მიენიჭა k-ს მნიშვნელობა, ე.ი.  j-ც გახდება k-ს ტოლი
cout<<j<<" "<<i;     // დაიბეჭდება  121  121
i++;
cout<<j<<" "<<i;     // დაიბეჭდება  122  122
```

გვარწმუნებს, რომ მითითება `i` არის `j` ცვლადის (რომელსაც `i` მიუთითებს) მეტსახელი.

ფუნქციის `reference`-პარამეტრი წარმოადგენს შესაბამისი არგუმენტის ფსევდონიმს. მაგალითად,

```
void f(int & i);
```

ფუნქციის პროტოტიპში `i` არის `int` ტიპის `reference`-პარამეტრი (ამბობენ აგრეთვე, რომ `i` არის მითითება `int` ტიპზე).

ვთქვათ, `k` – მთელი ტიპის ცვლადია. `f` ფუნქციის

```
f(k);
```

გამოძახების დროს სრულდება მინიჭება

```
int &i = k
```

ანუ `i` ხდება `k` ცვლადის მეტსახელი. ამიტომ `i` პარამეტრზე მიმართვა ფუნქციის ტანში ფაქტობრივად ნიშნავს `k` არგუმენტზე მიმართვას, და `i` –ს შეცვლა იგივე `k` –ს შეცვლას ნიშნავს.

არგუმენტის ასეთ გადაცემას უწოდებენ `Pass-by-reference`.

შემდეგ საილუსტრაციო მაგალითში ნაჩვენებია `reference`-პარამეტრის გამოყენება

```
#include <iostream>
using namespace std;
void f(int &);
int main(){
   int value = 1;
   cout<<"value–s sawyisi mnishvneloba: "
       <<value<<'\n';
   f(value);     //value–ს გადავცემთ მითითებით
   cout<<"value–s axali mnishvneloba: "
       <<value<<'\n';
   return 0;
}
void f(int &i){
   i = 10;    // არგუმენტის შეცვლა
}
```

პროგრამის შესრულების შედეგია:

```
value–s sawyisi mnishvneloba: 1
value–s axali mnishvneloba: 10
Press any key to continue . . .
```

შემდეგ მაგალითში reference–პარამეტრს გამოვიყენებთ ვექტორის კლავიატურიდან შევსების ფუნქციაში, ხოლო ვექტორის ელემენტების ბეჭდვის ფუნქციაში – const reference–პარამეტრს:

```
#include <iostream>
#include <vector>
using namespace std;
void fillVector(vector<int>& );
void printVector(const vector<int>& );
int main(){
   vector<int> A;
   fillVector(A);
   cout<<"Vectori:\n";
   printVector(A);
   return 0;
}
void fillVector(vector<int>& x){
   int number;
   while( cin>>number )
      x.push_back(number);
}
void printVector(const vector<int>& x){
   for(int i=0; i<x.size(); i++)
       cout<<x[i]<<' ';
   cout<<endl;
}
```

პროგრამის შესრულების შედეგია:

```
3 -7 19 231 0 25 -87 ^Z
Vectori:
3 -7 19 231 0 25 -87
Press any key to continue . . .
```

```
void fillVector(vector<int>& x);
```

ფუნქციის პარამეტრი წარმოადგენს reference–პარამეტრს. ეს აუცილებელია, რადგან ფუნქციის დანიშნულებაა ჩაწეროს ვექტორში კლავიატურიდან შემოსული მთელი რიცხვები, ანუ ფუნქციის

```
fillVector(A);
```

გამოძახების შედეგად A ვექტორი უნდა შეიცვალოს.

ფუნქცია printVector ბეჭდავს ვექტორს. რადგან A ვექტორის ზომა შეიძლება იყოს დიდი, ამ ფუნქციის პარამეტრი მიზანშეწონილია გამოვიყენოთ reference სახით, მიუხედავად იმისა რომ ვექტორის შეცვლას არ ვგეგმავთ. ასეთ შემთხვევებში უნდა ვიყოთ ფრთხილად – თუ ფუნქციის ტანში შემთხვევით შეიცვლება x ვექტორის ელემენტი (ელემენტები), ეს შეგვიცვლის A ვექტორს. ენაში გვაქვს დაცვის მექანიზმი, რომელიც მსგავსი შეცდომებისგან გვიცავს – const reference პარამეტრის გამოყენება:

```
void printVector(const vector<int>& x);
```

ახლა თუ ნებით ან უნებლიეთ შევეცდებით a ვექტორის შეცვლას, კონპილერი გამოიტანს შეცდომის შესახებ გზავნილს: 'x' : you cannot assign to a variable that is const.

თავი 9: პოინტერი. ფუნქციის არგუმენტი – პოინტერი. ფუნქციის პარამეტრების ინიციალიზება (გულისხმობის პრინციპით). ფუნქციების გადატვირთვა.

- მისამართები. მოქმედებები მისამართებზე.
- პოინტერი, მუდმივი პოინტერი.
- პოინტერის გამოყენება ფუნქციის არგუმენტად (Pass-by-pointer)
- ფუნქციის არგუმენტების მნიშვნელობა გაჩუმებით
- ფუნქციების გადატვირთვა
- ფუნქციის არგუმენტი – ფუნქცია

მისამართები. მოქმედებები მისამართებზე

C++ უზრუნველყოფს ძირითადი ტიპის მონაცემებთან (მთელი, სიმბოლური, ათწილადი) მუშაობას. პროგრამისტს თვითონაც შეუძლია მონაცემთა მრავალფეროვანი ტიპების შექმნა. ტიპები ერთმანეთისგან განსხვავდება არა მხოლოდ მეხსიერებაში მათი წარმოდგენით, არამედ იმ მოქმედებების (ოპერაციები, ფუნქციები) ერთობლიობითაც, რაც ამ ტიპის მონაცემებზეა დაშვებული.

ერთ-ერთი ძირითადი ოპერაცია არის მონაცემის მისამართის განსაზღვრა. თუ x არის რაიმე ტიპის მონაცემი, მაშინ &x არის მისი მისამართი მეხსიერებაში. იმის სანახავად, თუ რას წარმოადგენს მისამართი, საკმარისია იგი ამოვბეჭდოთ როგორც ათობითი ან თექვსმეტობითი რიცხვი.

&x–ის რიცხვითი მნიშვნელობა წარმოადგენს მეხსიერებაში იმ მონაკვეთის დასაწყისს, სადაც მოთავსებულია x–ის მნიშვნელობა. აღსანიშნავია, რომ &x–ის საშუალებით შეგვიძლია ისიც გავიგოთ თუ სად მთავრდება x–ის მნიშვნელობის შემცველი მონაკვეთი.

კარგად რომ გავერკვეთ მისამართების არითმეტიკაში, მეხსიერება უნდა წარმოვიდგინოთ როგორც დიდი მონაკვეთი, რომელიც დაყოფილია ბაიტებად და წარმოადგენს რიცხვითი ღერძის დისკრეტულ ანალოგს. მართლაც, &x იძლევა ათვლის წერტილს (სადაც &x, იგივეა რაც &x+0), ანუ საწყისი ბაიტის მისამართს, ხოლო ბაიტების ის რაოდენობა, რაც საჭიროა x მონაცემის მეხსიერებაში მოსათავსებლად წარმოადგენს ერთეულს. თუ ერთი და იგივე ტიპის მონაცემები მეხსიერებაში მიმდევრობით არის მოთავსებული, მაშინ &x+1 წარმოადგენს x-ის მომდევნო მონაცემის მისამართს და ა. შ. შეგვიძლია მეხსიერებაში ნავიგაცია &x-ის ორივე მხარეს "ერთეულის" ჯერადი მისამართებით.

ვთქვათ, გვაქვს განაცხადი

```
double x = 5.009;
```
(1)

და &x არის 1245012 –ის ტოლი, ხოლო **sizeof**(x) არის 8. ეს სიდიდეები დამოკიდებულია სისტემაზე, ხოლო მისამართის მნიშვნელობა, საზოგადოდ, დამოკიდებულია იმაზეც თუ კონკრეტულ სიტუაციაში რამდენად არის დატვირთული კომპიუტერი და პროგრამის სხვადასხვა გაშვებაზე შეიძლება სხვადასხვა მნიშვნელობა მოგვცეს. განსხვავებული იქნება ათვლის წერტილი, ხოლო მასშტაბი უცვლელი.

```
cout<<int(&x + 1);
```

შეტყობინების გამოყენებით მარტივად შეგვიძლია შევამოწმოთ, რომ &x+1 არის არა 1245013, არამედ 1245020, რადგან **double** ტიპის ცვლადისთვის ერთეულს წარმოადგენს 8 ბაიტი, რაც დაემატა მისამართს. ანალოგიურად, &x+3 არის 1245036 და ა. შ. ნავიგაცია შეიძლება ორივე მიმართულებით. მაგალითად, &x-186 არის 1243524.

მისამართებზე მოქმედებები მეხსიერების სხვადასხვა მონაკვეთებზე წვდომის საშუალებას იძლევა. ცხადია &x მისამართზე მოთავსებულია x–ის მნიშვნელობა, ანუ 5.0. x-ის მნიშვნელობის მოპოვება მისამართის საშუალებით შეიძლება ასე: (&x)[0]. ანალოგიურად, იმისათვის რომ გავიგოთ თუ რა მნიშვნელობა არის მოთავსებული &x-186 მისამართზე, ვწერთ (&x)[-186].

&x-ზე მოთავსებული მნიშვნელობის განსაზღვრისთვის C++ -ში გათვალისწინებულია ირიბი მნიშვნელობის აღების, ანუ განმისამართების ოპერაცია *(&x), იგივე *&x. მაგალითად, *(&x+17) იგივეა, რაც (&x)[17].

ყოველ ცვლადს, მაგალითად (1) განაცხადით განსაზღვრულს, გააჩნია სამი მახასიათებელი: აუცილებლად სახელი და მისამართი, და მნიშვნელობა. ამისგან განსხვავებით, მის მისამართს ანუ &x–ს გააჩნია სახელი (x–ის მისამართი), მნიშვნელობა (ჩვენს მაგალითში 2293620) და ირიბი მნიშვნელობა (ჩვენს მაგალითში 5.0). როგორც ვხედავთ, ცვლადის მისამართს არ გააჩნია თავისი მისამართი, თუმცა ხშირად ჩნდება მისამართების დამახსოვრების აუცილებლობა მასზე განთავსებული მნიშვნელობის აღდგენის მიზნით. ჩვენთვის აქამდე ცნობილი მონაცემთა ტიპების საფუძველზე ამის გაკეთება შეუძლებელია.
მაგალითად, ავიღოთ

```
int  p;
```

და დავიმახსოვროთ

```
p = &x;        // შეცდომა
```

ფორმალურად p ტოლია x–ის მისამართის, მაგრამ მისი საშუალებით ჩვენ ვეღარ აღვადგენთ &x -ის ირიბ მნიშვნელობას ანუ x–ს, რადგან მთელი ტიპის ცვლადზე ირიბი მნიშვნელობის აღების ოპერაცია (*p) განსაზღვრული არაა.

მიმთითებელი ანუ პოინტერი

მისამართებთან სამუშაოდ C++ ენაში შემოტანილია მისამართის ტიპი. მისამართის ტიპის ცვლადს პოინტერი (მიმთითებელი) ეწოდება. პოინტერის დანიშნულებაა ცვლადის მისამართის შენახვა. მასზე განაცხადს აქვს სახე:

*ტიპი *პოინტერი;*

მაგალითად, შეტყობინება

```
float *f_ptr;                                    (2)
```

ნიშნავს, რომ f_ptr –ის მნიშვნელობა შესაძლებელია იყოს მხოლოდ **float** ტიპის ცვლადის მისამართი, ხოლო მისი ირიბი მნიშვნელობა იქნება **float** ტიპისა. ზოგადად, პოინტერი არის ცვლადი, რომელსაც გააჩნია ოთხი მახასიათებელი: სახელი, მისამართი (ესენი აუცილებლად), აგრეთვე მნიშვნელობა და ირიბი მნიშვნელობა. (2) შეტყობინებით შემოტანილ პოინტერს ბოლო ორი ჯერ-ჯერობით არ გააჩნია.

შემდეგი კოდის ფრაგმენტი:

```
double *f_ptr, x = 13.45;
f_ptr = &x;
```

ნიშნავს შემდეგს: f_ptr –ის ირიბი მნიშვნელობა არის **double** ტიპის, x -ის მნიშვნელობაც არის **double** ტიპის, f_ptr –ის მნიშვნელობა x -ის მისამართია, ხოლო ირიბი მნიშვნელობა არის 13.45, ანუ *f_ptr ტოლია 13.45-ის.
ან, მაგალითად,

```
int a = 9;
int *a_ptr = &a;
```

ნიშნავს, რომ a –ს მნიშვნელობა და a_ptr –ის ირიბი მნიშვნელობა *a_ptr, ორივე უდრის 9-ს. უფრო მეტიც, ერთის შეცვლა იწვევს იგივე მნიშვნელობით მეორის შეცვლას. მაგალითად, *a_ptr = 34; შეტყობინების შემდეგ a –ს მნიშვნელობა იქნება 34.

პოინტერების ერთ-ერთ თავისებურებას ის წარმოადგენს, რომ ყველა ტიპის პოინტერი მეხსიერებაში, ჩვეულებრივ, 4 ბაიტს იკავებს. ეს იძლევა საშუალებას, რომ განვსაზღვროთ **void** ტიპის პოინტერიც, რომელსაც შემდეგ შეგვიძლია მივანიჭოთ სხვა ტიპის პოინტერების მნიშვნელობები.

არარსებული მისამართის აღსანიშნავად გამოიყენება მუდმივი NULL. მაგალითად, შეტყობინება

```
f_ptr = NULL;
```

ნიშნავს, რომ ამ პოინტერს არა აქვს კონკრეტული მნიშვნელობა. თუ გამოვბეჭდავთ, არარსებული მისამართი დაიბეჭდება როგორც 0. NULL მუდმივი განსაკუთრებით სასარგებლოა დინამიკურ მონაცემთა სტრუქტურებთან მუშაობისას.

მუდმივი პოინტერი

მუდმივ პოინტერზე განაცხადის გაკეთება, სხვა ტიპებისგან განსხვავებით, მეტ დაკვირვებას მოითხოვს. თუ ჩვენ, მაგალითად, გავაკეთებთ ასეთ განაცხადს:

```
const int number = 5;
```

იგი აცნობებს C++ –ს, რომ number ცვლადი არის მუდმივი, ასე რომ

```
number = 10;    // შეცდომა
```

არის შეცდომა. ფრაგმენტი

```
char symbol = 'M';
const char *ch_ptr = &symbol;
```

C++ –ისთვის არ ნიშნავს, რომ ცვლადი ch_ptr არის მუდმივი, იგი აცნობებს C++ –ს რომ ამ ცვლადის ირიბი მნიშვნელობა არის მუდმივი, ანუ ირიბ მნიშვნელობას, რომელზეც მიუთითებს პოინტერი, ვერ შევცვლით, მაგალითად,

```
*ch_ptr = 'X';
```

შეცდომაა. მაგრამ შეგვიძლია შევცვალოთ პოინტერის მნიშვნელობა.

იმისათვის რომ C++ –ისთვის თვითონ პოინტერი იყოს მუდმივი, უნდა გავაკეთოთ, მაგალითად, ასეთი განაცხადები:

```
char c = 'M';
char * const c_ptr = &c;
```

ან

```
double y = 0.00357;
double * const d_ptr = &y;
```

რაც ნიშნავს, რომ პოინტერი მუდმივია, მისი ირიბი მნიშვნელობა კი არა. მაშასადამე სწორია

```
*c_ptr = 'B';      // სწორია, რადგან *c_ptr არის სიმბოლო
*d_ptr = 6.1;      // სწორია, რადგან *d_ptr არის რიცხვი
```

მაგრამ არასწორია

```
c_ptr += 3;        // არასწორია, რადგან c_ptr მუდმივია
d_ptr = &y - 1;    // არასწორია, რადგან d_ptr მუდმივია
```

დასასრულ, C++ საშუალებას გვაძლევს ერთდროულად გამოვაცხადოთ მუდმივებად როგორც პოინტერი, ასევე მისი ირიბი მნიშვნელობა. მაგალითად:

```
int K = 719;
const int *const ptr = &K;
```

პოინტერის გამოყენება ფუნქციის არგუმენტად (**Pass-by-pointer**)

ჩვენ განვიხილეთ ფუნქციისთვის არგუმენტების გადაცემის ორი გზა: არგუმენტების ასლების გადაცემა (Pass-by-value), და არგუმენტების გადაცემა მისამართებთან ერთად (Pass-by-reference).

პირველი მიდგომა ყველაზე ადრინდელია. დაკავშირებულია დროისა და მეხსიერების ხარჯთან, ამ დროს ფუნქციას შეუძლია რომ მხოლოდ ერთადერთი მნიშვნელობა დააბრუნოს **return** შეტყობინების გამოყენებით. ფუნქციის გამოძახებისას არგუმენტების ასლების გადაცემა ჰგავს "ცალმხრივ მოძრაობას" - პარამეტრების შეცვლილი მნიშვნელობები უკან აღარ ბრუნდება, სამაგიეროდ, ესაა საკმაოდ კარგი გზა საიმედო კოდის შექმნისთვის.

მეორე მიდგომა ძალიან თანამედროვეა, იგი აქტიურად გამოიყენება პროგრამირების სხვა თანამედროვე ენებშიც და მას დიდ დროს დავუთმობთ შემდეგში.

C++ -ში არსებობს საშუალება, რომ ფუნქციას გადავცეთ არა არგუმენტის ასლი, არამედ მისი მისამართი. რა ცვლადის მისამართსაც გადავაწვდით, იმ ცვლადის დაბრუნება აღარაა საჭირო, რადგან რეალურ მეხსიერებაში ფუნქციის მოქმედების შედეგად ჩაიწერება საჭირო მნიშვნელობა. ამ მიზნის მისაღწევად (ე.ი. მისამართების გადასაცემად) უნდა ვისარგებლოთ პოინტერით.

მაგალითად, გვაქვს ფუნქცია, რომელიც ზრდის count ცვლადს ერთით, ხოლო პროგრამაში ეს ფუნქცია რამდენჯერმე გამოიძახება. გავიგოთ, რამდენჯერ მოხდა ფუნქციის გამოძახება.

```
#include <iostream>
using namespace std;
void inc_count(int *count_ptr)
{
    (*count_ptr)++;
}
int main()
{
    int count(0);                   // განმეორებათა რაოდენობა
    while (count < 10)
        inc_count(&count);
    cout<<"count =  "<<count<<endl;
    system("PAUSE");
    return (0);
}
```

როგორც ფუნქციის სათაური გვიჩვენებს, ფუნქციას გადაეცემა მთელი ტიპის რიცხვის მისამართი count_ptr, რომლის ირიბი მნიშვნელობა არის მთელი. მართლაც, main –ში ამ ფუნქციის ყოველი გამოძახება ხდება სახით: inc_count(&count); შედეგად ფუნქციის ლოკალური ცვლადი "დაიჭერს" მისამართს ანუ count_ptr = &count; და არგუმენტის ამ მნიშვნელობისთვის ფუნქციის ტანი (*count_ptr)++; განხორციელდება სახით:

(*&count)++; ანუ count++;

როგორც ვხედავთ, ყოველ გამოძახების შედეგად იზრდება იმ ცვლადის მნიშვნელობა, რომლის მისამართსაც ფუნქცია მიიღებს არგუნეტად.

ვთქვათ, ჩვენი ამოცანის პირობაა: დაწერეთ პროგრამა, რომელიც გამოითვლის კლავიატურიდან შემოტანილი რადიუსის მქონე წრის ფართობსა და წრეწირის სიგრძეს. ორივე სიდიდის მისაღებად დაწერეთ და პროგრამაში გამოიყენეთ ერთი ფუნქცია.

```
#include <iostream>
using namespace std;

const double PI = 3.14159;

double Circle(double r, double * sAddress);

int main()
{
   double radius, area, length;
   cout<<"Enter circle's radius  ";
   cin >> radius;
   length = Circle( radius, &area );
   cout<<"Area of Circle = "<< area
      <<"\nLength of Circle = "<< length << endl;
   return 0;
}

double Circle(double r, double * sAddress)
{
   * sAddress = PI * r * r;
   return 2 * PI * r;
}
```

პროგრამის შედეგი:

```
Enter circle's radius  1
Area of Circle = 3.14159
Length of Circle = 6.28318
Press any key to continue . . .
```

`Circle` ფუნქცია **`return`** –ის საშუალებით დააბრუნებს წრეწირის სიგრძეს, და ეს სიდიდე მიენიჭება `main`–ის `length` ცვლადს, ხოლო პარამეტრი –პოინტერის საშუალებით დააბრუნებს წრის ფართობს შემდეგნაირად: გამოძახების დროს გვექნება `sAddress=&area;` ამიტომ ფუნქციის ტანში პირველი შეტყობინება შესრულდება რეალური ცვლადებისთვის შემდეგი სახით:

```
*&area = PI * radius * radius;
```

რაც ნიშნავს, რომ `area` ცვლადის მისამართზე ჩაიწერება (ანუ`area` ცვლადს მიენიჭება) წრის ფართობის მნიშვნელობა.

ფუნქციის არგუმენტების მნიშვნელობა გულისხმობის პრინციპით

C++ -ის სინტაქსის მიხედვით ფუნქციის პარამეტრებს შეიძლება მივანიჭოთ საწყისი მნიშვნელობები, ანუ მოვახდინოთ პარამეტრების ინიციალიზება ფუნქციაზე განაცხადის დროს.

ფუნქციის გამოძახებისას, ჩვეულებრივ, ფუნქციის არგუმენტებისა და პარამეტრების რიცხვი უნდა ემთხვევოდეს. თუ ფუნქციის პროტოტიპში პარამეტრებს მინიჭებული აქვთ საწყისი მნიშვნელობები, ფუნქციის არგუმენტების რაოდენობა შეიძლება იყოს პარამეტრების რაოდენობაზე ნაკლები. ამ შემთხვევაში გამოტოვებული არგუმენტის ნაცვლად ფუნქცია გამოიყენებს მისი შესაბამისი პარამეტრის საწყის მნიშვნელობას. ასეთ არგუმენტებს ეწოდებათ *ნაგულისხმევი არგუმენტები.*

მაგალითად, თუ ფუნქციის პროტოტიპია

```
void f ( int x = 3, int y = 7 );
```

მაშინ მის გამოძცებას შესაძლოა ჰქონდეს შემდეგი სახე:

```
f( 10, –67);
```

ან

```
f();   // გამოიძახება f( 3, 7);
```

შესაძლებელია პარამეტრების ნაწილობრივი ინიციალიზება. ამ შემთხვევაში ინიციალიზება უნდა იწყებოდეს *მარჯვნიდან, დაწყებული ბოლო პარამეტრით*. მაგალითად,

```
void func(int a, int b=2, float c=3.75);
```

სინტაქსურად სწორი პროტოტიპია. ამ ფუნქციის გამოძახებას შესაძლოა ჰქონდეს სახე:

```
func(10);    // გამოიძახება f ( 10, 2, 3.75  );
func(5,7);   // გამოიძახება f ( 5, 7, 3.75 );
func(1,3,12.7);
```

ხოლო გამოძახება

```
func();
```

გამოიწვევს კომპილაციის შეცდომას.

ფუნქციის პროტოტიპში დასაშვებია პარამეტრების სახელების გამოტოვება. მაგალითად, სწორი იქნება პროტოტიპი

```
void func1(int, int=2, float=3.75);
```

უნდა გავითვალისწინოთ, რომ `func1` ფუნქციის განსაზღვრისას პარამეტრების სახელების მითითება *აუცილებელია*, ხოლო პარამეტრების საწყისი მნიშვნელობების გამეორება *შეცდომაა*. მაგალითად, `func1` ფუნქციის განსაზღვრას შესაძლოა ჰქონდეს სახე

```
void func1(int x, int y, float z){
    // შესრულებადი შეტყობინებები
}
```

გავითვალისწინოთ, რომ თუ პროტოტიპი და ინტერპრეტაცია განცალკევებულია, მაშინ ნაგულისხმევი არგუმენტების მითითება მხოლოდ პროტოტიპშია საჭირო.

შემდეგ საილუსტრაციო მაგალითში შევქმნით ფუნქციას `Sum`, რომელიც გამოძახების შესაბამისად გამოითვლის ორი ნამდვილი რიცხვის ან სამი ნამდვილი რიცხვის ჯამს.

```
#include <iostream>
using namespace std;
double Sum(double, double, double = 0.);
int main(){
   double a, b, c;
   cout<<"Enter 3 real numbers:  ";
   cin >> a >> b >> c;
   cout<<"Two numbers sum is "
        << Sum( a, b)<<endl
        <<"Three numbers sum is "
        <<Sum( a, b, c )<<endl;
   return 0;
}
double Sum(double x, double y, double z)
{
     return x + y + z;
}
```

პროგრამა დაბეჭდავს:

```
Enter 3 real numbers:  1.2 3.4 5.6
Two numbers sum is 4.6
Three numbers sum is 10.2
Press any key to continue . . .
```

ფუნქციების გადატვირთვა

C++ -ში შესაძლებელია ერთი და იგივე სახელი დავარქვათ რამდენივე ფუნქციას იმ პირობით, რომ ასეთი ფუნქციების პარამეტრების სია იქნება განსხვავებული: განსხვავებული უნდა იყოს ან პარამეტრების რიცხვი, ან მათი ტიპი, ან პარამეტრების რიცხვიც და ტიპიც. ფუნქციების ამ თვისებას ეწოდება *ფუნქციათა გადატვირთვა* (`function overloading`). გადატვირთული (`overloaded`) ფუნქციების გამოძახების დროს C++ -ის კომპილერი აანალიზებს არგუმენტების რაოდენობას, მათ ტიპსა და რიგითობას და ისე ადგენს შესასრულებელი ფუნქციის შესაბამის ეკზემპლარს. უნდა აღვნიშნოთ, რომ ფუნქციების დასაბრუნებელ მნიშვნელობას კომპილერი "ყურადღებას არ აქცევს". ე.ი. თუ ფუნქციების პროტოტიპები განსხვავდება მხოლოდ დასაბრუნებელი მნიშვნელობებით, ასეთი ფუნქციების გადატვირთვა არ შეიძლება.

შემდეგ მაგალითში გადატვირთულია ფუნქცია `Minimal`, რომლის სამი ვერსია განსხვავებული მოქმედებისაა: პირველი პოულობს სამი სიმბოლოდან უმცირესს, მეორე ადგენს უმცირესს მთელ რიცხვთა ვექტორში, ხოლო მესამე აბრუნებს უმცირესს ორ ნამდვილ რიცხვს შორის.

```
#include <iostream>
#include <vector>
using namespace std;
char Minimal(char, char, char );
int Minimal(vector<int> );
double Minimal(double, double );
int main(){
   char a, b, c;
   cout<<"ShemoitaneT 3 simbolo ";
   cin >> a >> b >> c;
   cout<<"ShemoitaneT 2 namdvili ricxvi ";
   double d, t;
   cin >> d >> t;
   vector <int> V;
   int n;
   cout<<"Shemoitanet ramdenime mTeli ricxvi  ";
   while( cin >> n)
        V.push_back(n);
   cout<<"3 simbolos shoris unciresi = "
        <<Minimal(a, b, c)<<endl;
   cout<<"2 namdvil ricxvs Soris umciresi = "
        <<Minimal( d, t )<<endl;
   cout<<"Veqtoris umciresi elementi = "
        <<Minimal( V )<<endl;
   return 0;
}
char Minimal(char x, char y, char u){
     char min = x;
     if( y < min ) min = y;
     if( u < min ) min = u;
     return min;
}
int Minimal(vector<int> x){
   int min = x[0];
   for(int i=0; i<x.size(); ++i)
       if( x[i] < min )
         min = x[i];
   return min;
}
double Minimal(double a, double b){
     return a < b ? a : b;
```

```
}
```
პროგრამა დაბეჭდავს:

```
ShemoitaneT 3 simbolo a A z
ShemoitaneT 2 namdvili ricxvi 23.78 10.25
Shemoitanet ramdenime mTeli ricxvi  100 -180 435 -200 793
^Z
3 simbolos Soris unciresi = A
2 namdvil ricxvs Soris umciresi = 10.25
Veqtoris umciresi elementi = -200
Press any key to continue . . .
```

C++ ენის საკმაოდ გავრცელებულ დიალექტში (C++/CLR), გულისხმობის პრინციპით არგუმენტების გადაცემა არ არის დაშვებული, რადგან ალტერნატივად განიხილება ფუნქციების გადატვირთვის მექანიზმი. მაგალითად, განვიხილოთ თუ რა იქნება

```
void F(int a, int b=2, float c=3.75)
{
    std::cout << a+b+c << std::endl;
}
```

ფუნქციის ალტერნატივა (C++/CLR)-ში. იგივე შედეგის მისაღწევად, ვაკეთებთ სამ ფუნქციას: ერთს სამი არგუმენტით,

```
void F(int a, int b, float c)
{
    std::cout << a+b+c << std::endl;
}
```

ერთს ორი არგუმენტით,

```
void F(int a, int b, float c)
{
    std::cout << a+b+3.75 << std::endl;
}
```

ერთს ერთი არგუმენტით,

```
void F(int a, int b, float c)
{
    std::cout << a+2+3.75 << std::endl;
}
```

ფუნქციის არგუმენტი – ფუნქცია

ფუნქციას შეიძლება არგუმენტად გადავცეთ სხვა ფუნქცია. შემდეგი პროგრამა ვექტორში ითვლის 3–ის ჯერად ელემენტებს და შემდეგ უარყოფით ელემენტებს. პროგრამაში შექმნილია **bool** ტიპის 2 ფუნქცია `isMultiple_3` და `isNegative`. პირველი ადგენს, არის თუ არა მისი მთელი არგუმენტი 3–ის ჯერადი რიცხვი, მეორე კი – არის თუ არა მისი მთელი არგუმენტი უარყოფითი რიცხვი. ასევე გვაქვს ფუნქცია `Count`, რომელსაც არგუმენტად გადავცემთ პირველს და შემდეგ მეორე ფუნქციას. ვექტორში 15 შემთხვევითი რიცხვია `[-20;40]` შუალედიდან.

```
#include <iostream>
#include <vector>
#include <ctime>
using namespace std;
bool isNegative(int );
bool isMultiple_3(int );
int Count(vector<int>, bool f(int) );
void fillVector(vector<int>& );
void printVector(const vector<int>& );
```

```
int main (){
  vector<int> A;
  fillVector( A );
  printVector( A );
  cout<<"3-is jeradebis raodenoba = "
       << Count( A, isMultiple_3 )<< endl;
  cout<<"UaryofiTebis raodenoba = "
       << Count( A, isNegative )<< endl;
  return 0;
}
bool isNegative(int x){
  return x < 0;
}
bool isMultiple_3(int x){
  return x%3 == 0;
}
int Count(vector<int> x, bool f(int) ){
   int counter =0;
   for(int i=0; i< x.size(); ++i)
      if( f( x[i]) ) counter++;
   return counter;
}
void fillVector(vector<int>& x){
      int number, count = 1;
      srand( time(NULL) );
      while( count <= 15 ){
         number = rand()%61 - 20;
         x.push_back( number );
         count++;
      }
}
void printVector(const vector<int>& x){
   for(int i=0; i<x.size(); ++i)
     cout<<x[i]<<' ';
   cout<<endl;
}
```

პროგრამის მუშაობის შედეგია:

```
22 -9 -2 -15 15 2 39 38 -20 -4 7 22 -20 3 10
3-is jeradebis raodenoba = 5
UaryofiTebis raodenoba = 6
Press any key to continue . . .
```

თავი 10: ორგანზომილებიანი ვექტორი

- ვექტორის ელემენტი – ვექტორი
- ტიპის განმსაზღვრელი **typedef**
- ორგანზომილებიანი ვექტორის გადაცემა ფუნქციაში
- მოქმედება ორგანზომილებიან ვექტორზე

ვექტორების ვექტორი (ორგანზომილებიანი ვექტორი)

პრაქტიკულ ამოცანებში ხშირად მოსახერხებელია საწყისი მონაცემები წარმოვადგინოთ ცხრილის სახით. C++ გვაძლევს საშუალებას ასეთი წარმოდგენისთვის შევქმნათ შესაბამისი მონაცემთა ტიპი – ვექტორების ვექტორი, ანუ ორგანზომილებიანი ვექტორი.

მაგალითად, განაცხადი

```
vector< vector <int> > intMatrix;
```

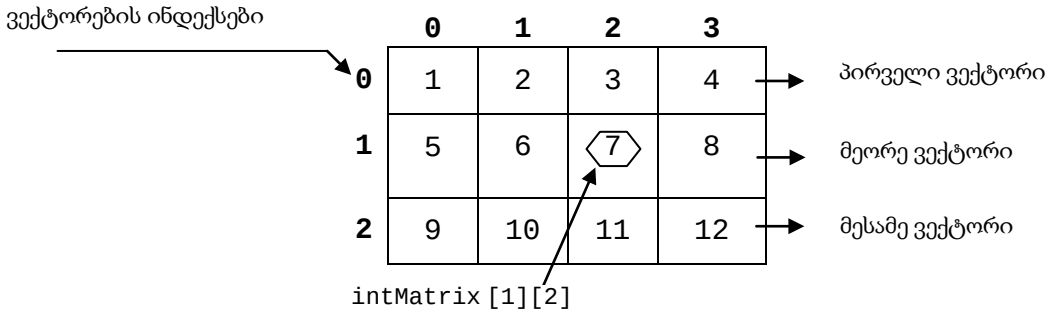
შექმნის ვექტორს `intMatrix`, რომლის ყოველი ელემენტი იქნება მთელი რიცხვების ვექტორი. ვამბობთ იქნება, რადგან ჯერჯერობით ეს ვექტორი ცარიელია. ვექტორში ელემენტების დამატება (`push_back()` მეთოდით) დიდად არ განსხვავდება ერთგანზომილებიან ვექტორში ელემენტების დამატებისგან და ამაზე ოდნავ მოგვიანებით ვილაპარაკებთ.

ვთქვათ, `intMatrix` ვექტორში უკვე ჩაწერილია გარკვეული ელემენტები. მაშინ `intMatrix` ვექტორის ელემენტებს შეიძლება მივმართოთ ინდექსის გამოყენებით: `intMatrix[0], intMatrix[1], ... , intMatrix[ intMatrix.size() – 1 ]`. ოღონდ, ეს ელემენტები არის, თავის მხრივ, მთელი რიცხვების ვექტორები და მათში ჩაწერილ რიცხვებზე წვდომა ხორციელდება ორი ინდექსის გამოყენებით: `intMatrix[i][j]`, სადაც `i` არის `intMatrix` – ში ელემენტი–ვექტორის ინდექსი, ხოლო `j` არის ამ ელემენტ–ვექტორში ელემენტის (რიცხვის) ინდექსი.

მაგალითად, `intMatrix` –ის პირველი (ინდექსით `0`) ვექტორის მესამე (ინდექსით `2`) რიცხვი არის

```
intMatrix[0][2]
```

თუ `intMatrix` ვექტორს აქვს სამი ელემენტი (თითოეული ელემენტი - ერთგანზომილებიანი ვექტორია), ხოლო მისი ყოველი ელემენტი (როგორც ერთგანზომილებიანი ვექტორი) შეიცავს ოთხ მთელ რიცხვს, მაშინ ორგანზომილებიანი ვექტორი `intMatrix` შეგვიძლია წარმოვიდგინოთ როგორც ცხრილი. `intMatrix` –ის ელემენტები ასეთი ცხრილის სტრიქონებს შეესაბამება, ხოლო რიცხვები – სტრიქონებში ელემენტებს.



თავდაპირველად ცარიელ `intMatrix` ვექტორში რიცხვები რომ ჩაიწეროს, საჭიროა შემდეგი მოქმედებების ჩატარება:

- უნდა გაუნაწილდეს (ანუ გამოეყოს) დინამიკური მეხსიერება ერთგანზომილებიან ვექტორს;
- დაემატოს იგი `intMatrix` –ს `push_back()` ფუნქციის გამოყენებით;
- შემდეგ ყოველ `intMatrix[i]` –ურში ჩავწეროთ რიცხვები.

შემდეგი პროგრამა შექმნის ორგანზომილებიან `intMatrix` ვექტორს, რომელიც შეიცავს `M (4)` ელემენტს – ვექტორს, ყოველში `N (5)` რიცხვით.

```
#include <iostream>
#include <iomanip>
#include <vector>
using namespace std;
const int M = 4, N = 5;
int main (){
   vector< vector <int> > intMatrix;
    // intMatrix -ში ვექტორების დამატება
    for(int i=0; i<M; i++)
    {
         vector<int> row;
         intMatrix.push_back(row);
    }
    // ყოველ intMatrix[i] ვექტორში რიცხვების ჩაწერა
    for(int i=0; i<M; i++)
       for (int j=0; j<N; j++)
           intMatrix[i].push_back(i+j);
    // intMatrix ვექტორის სტრიქონ-სტრიქონ ბეჭდვა
    for(int i=0; i<intMatrix.size(); i++)
    {
        for (int j=0; j<intMatrix[i].size(); j++)
        {
           cout<<setw(4)<<intMatrix[i][j];
        }
        cout<<endl;
    }
    cout<<endl;
    return 0;
}
```

პროგრამის მუშაობის შედეგია:

```
   0   1   2   3   4
   1   2   3   4   5
   2   3   4   5   6
   3   4   5   6   7
Press any key to continue . . .
```

როდესაც ორგანზომილებიანი ვექტორი შეიცავს ერთი ზომის ვექტორებს (მათი ელემენტების რაოდენობა ერთი და იგივეა) შეგვიძლია მათთვის მეხსიერება გამოვყოთ განაცხადის გაკეთებისთანავე.

გავიხსენოთ ერთგანზომილებიანი ვექტორის შექმნის ერთ–ერთი ხერხი:

```
vector<int> X(4,100);
```

განაცხადის საფუძველზე იქმნება `X` ვექტორი, რომელსაც შექმნისთანავე გამოეყოფა მეხსიერება `4` მთელი რიცხვისთვის და თითოეული რიცხვის მეხსიერებაში ჩაიწერება `100`.

```
vector<int> X(4);
```

შემთხვევაში კი X ვექტორის 4 –ივე ელემენტს გულისხმობის პრინციპით მიენიჭება 0.

თუ ვისარგებლებთ ამ ხერხით, მაშინ

```
vector <int> row(5);
vector< vector <int> > intMatrix(4, row);
```

განაცხადი უზრუნველყოფს intMatrix ორგანზომილებიანი ვექტორისთვის 4 ვექტორი–ელემენტის შექმნას, ყოველში 5 მთელი 0–ლის ტოლი რიცხვით.

ამიტომ წინა ამოცანაში შეგვიძლია დავწეროთ

```
vector <int> row(N);
vector< vector <int> > intMatrix(M, row);
```

რაც გარანტიას გვაძლევს, რომ intMatrix –სთვის ყველა საჭირო მეხსიერება იქნება განაწილებული და მეტიც, განულებული. ამიტომ push_back ფუნქციის გამოყენება ამ შემთხვევაში საჭირო აღარ არის. უფრო მეტიც, ამ შემთხვევაში ვექტორის შევსების დროს შეგვიძლია გავითვალისწინოთ, რომ intMatrix ვექტორის ზომა არის intMatrix.size(), ხოლო მისი ყოველი სტრიქონის ზომა (ელემენტების რაოდენობა) არის intMatrix[i].size(), სადაც i ინდექსი მოთავსებულია ნულსა და intMatrix.size()-1 სიდიდეებს შორის (ბოლოების ჩათვლით). ახლა, ჩვენი პროგრამა მიიღებს შემდეგ სახეს:

```
#include <iostream>
#include <iomanip>
#include <vector>
using namespace std;
const int M = 4, N = 5;
int main (){
    vector <int> row(N);
    vector< vector <int> > intMatrix(M, row);
   // ყოველ intMatrix[i] ვექტორში რიცხვების ჩაწერა
   for(int i=0; i<intMatrix.size(); i++)
      for (int j=0; j<intMatrix[i].size(); j++)
          intMatrix[i][j]= i+j;
   // intMatrix ვექტორის სტრიქონ-სტრიქონ ბეჭდვა
   for(int i=0; i<intMatrix.size(); i++)
   {
       for (int j=0; j<intMatrix[i].size(); j++)
       {
          cout<<setw(4)<<intMatrix[i][j];
       }
       cout<<endl;
   }
   cout<<endl;
   return 0;
}
```

ცხრილის სახით წარმოდგენასთან ანალოგიით ორგანზომილებიან ვექტორში შემავალ ვექტორებს მოიხსენიებენ როგორც მის სტრიქონებს, ხოლო ვექტორებში ერთი და იგივე ადგილას მდგომ ელემენტებს (ტოლი მეორე ინდექსით) – როგორც სვეტებს.

სხვათა შორის, შეგვიძლია სხვადასხვა სტრიქონს კიდევ დავუმატოთ სხვადასხვა რაოდენობის ელემენტები, რის შედეგადაც ჩვენი ვექტორი გახდება არამართკუთხა.

განხილული ორი მაგალითი სრულიად საწინააღმდეგო იყო ერთმანეთის იმ აზრით, რომ პირველში გავაკეთეთ სრულიად ცარიელი ვექტორი და მასში დინამიკურად ჩავამატეთ როგორც სტრიქონები, ასევე სვეტები. მეორე მაგალითში კი თვიდანვე დავაფიქსირეთ სვეტებისა და სტრიქობების რაოდენობები.

შესაძლებელია მესამე, კომპრომისული ვარიანტიც: თავიდან დავაფიქსიროთ მხოლოდ სტრიქონების რაოდენობა, ხოლო სტრიქონებში ელემენტები ჩავყაროთ `push_back()` მეთოდით. მაგალითად:

```
#include <iostream>
#include <iomanip>
#include <vector>
using namespace std;
const int M(4), N(5);
int main (){
    vector< vector <int> > intMatrix(M);
   // ყოველ intMatrix[i] ვექტორში რიცხვების ჩაწერა
   for(int i=0; i<intMatrix.size(); i++)
      for (int j=0; j<N; j++)
          intMatrix[i].push_back(i+j);
   // intMatrix ვექტორის სტრიქონ-სტრიქონ ბეჭდვა
   for(int i=0; i<intMatrix.size(); i++)
   {
       for (int j=0; j<intMatrix[i].size(); j++)
       {
          cout<<setw(4)<<intMatrix[i][j];
       }
       cout<<endl;
   }
   cout<<endl;
   return 0;
}
```

ტიპის განმსაზღვრელი `typedef`

C++ შესაძლებლობას აძლევს პროგრამისტს განსაზღვროს საკუთარი ტიპი **`typedef`** შეტყობინების საშუალებით. **`typedef`** შეტყობინების ზოგადი ფორმა ასეთია:

`typedef` *განაცხადი ტიპზე;*

სადაც *განაცხადი ტიპზე* იგივეა, რაც განაცხადი ცვლადებზე, იმ განსხვავებით, რომ ცვლადის სახელის ნაცვლად გამოიყენება ტიპის სახელი. მაგალითად:

```
typedef int count;
```

განსაზღვრავს ახალ ტიპს `count`, რომელიც იგივეა რაც `int`. ასე, რომ განაცხადი:

```
count flag;
```

იგივეა რაც

```
int flag;
```

`typedef` –ის გამოყენებით ჩვენ პროგრამაში შეგვიძლია განვსაზღვროთ შემდეგი ორი ტიპი:

```
typedef vector<int> row;
typedef vector< row > intMatrix;
```

მაშინ პროგრამა მიიღებს სახეს:

```
#include <iostream>
#include <iomanip>
#include <vector>
using namespace std;
typedef vector<int> row;
typedef vector< row > matrix;
```

```
const int M = 4, N = 5;
int main (){
   row R(N);
   matrix intMatrix(M, R);
   for(int i=0; i<M; i++)
      for (int j=0; j<N; j++)
          intMatrix[i][j]= i+j;
   for(int i=0; i<intMatrix.size(); i++)
   {
       for (int j=0; j<intMatrix[i].size(); j++)
       {
          cout<<setw(4)<<intMatrix[i][j];
       }
       cout<<endl;
   }
   cout<<endl;
   return 0;
}
```

ორგანზომილებიანი ვექტორის გადაცემა ფუნქციაში

განხილულ საილუსტრაციო მაგალითში ჩვენ ვქმნით ორგანზომილებიან ვექტორს და შემდეგ ვბეჭდავთ მას ცხრილის სახით ვექტორ–ვექტორ. ამოცანის კოდი მთლიანად მოთავსებულია `main` –ში, რაც C++ ენაში პროგრამირების კარგ სტილად არ ითვლება. დროა პროგრამა ავაგოთ C++ -ისთვის ჩვეულ სტილში: დავწეროთ ვექტორის შევსების ფუნქცია, ვექტორის ბეჭდვის ფუნქცია, ხოლო `main` –ში მოვახდინოთ ამ ფუნქციების გამოძახება:

ვექტორში რიცხვების ჩაწერის ფუნქცია ასე გამოიყურება:

```
void fillMatrix(vector< vector<int> > & a){
   for(int i=0; i<M; i++)
      for(int j=0; j<N; j++)
          a[i][j]= i+j;
}
```

ვექტორის ელემენტების ბეჭდვის ფუნქციას აქვს სახე

```
void printMatrix(const vector< vector<int> > &a){
   for(int i=0; i<a.size(); i++)
   {
       for (int j=0; j<a[i].size(); j++)
           cout<<setw(4)<<a[i][j];
       cout<<endl;
   }
   cout<<endl;
}
```

ხოლო ჩვენი პროგრამა ასე გადაიწერება:

```
#include <iostream>
#include <iomanip>
#include <vector>
using namespace std;
void fillMatrix(vector< vector<int> > & a);
void printMatrix(const vector< vector<int> > &a);
const int M = 4, N = 5;
int main (){
   vector <int> row(N);
   vector< vector <int> > intMatrix(M, row);
```

```
   fillMatrix( intMatrix );
   printMatrix( intMatrix );
   return 0;
}
void fillMatrix(vector< vector<int> > & a){
   for(int i=0; i<M; i++)
      for (int j=0; j<N; j++)
          a[i][j]= i+j;
}
void printMatrix(const  vector< vector<int> > &a){
   for(int i=0; i<a.size(); i++)
   {
       for (int j=0; j<a[i].size(); j++)
           cout<<setw(4)<<a[i][j];
       cout<<endl;
   }
   cout<<endl;
}
```

თუ კი ვისარგებლებთ **typedef** შეტყობინებით და როგორც ადრე შემოვიღებთ ტიპის ახალ სახელს (ახალ ტიპს) ორგანზომილებიანი ვექტორის ელემენტი–ვექტორისთვის

```
typedef vector<int> row;
```

და თვითონ ორგანზომილებიანი ვექტორისთვის

```
typedef vector< row > matrix;
```

მაშინ ჩვენი პროგრამა გამარტივებული სახით ჩაიწერება

```
#include <iostream>
#include <iomanip>
#include <vector>
using namespace std;
typedef vector<int> row;
typedef vector< row > matrix;
const int M = 4, N = 5;
void fillMatrix(matrix & a);
void printMatrix(matrix a);
int main (){
   row R(N);
   matrix intMatrix(M, R);
   fillMatrix( intMatrix );
   printMatrix( intMatrix );
   return 0;
}
void fillMatrix(matrix & a)
{
   for(int i=0; i<M; i++)
      for (int j=0; j<N; j++)
          a[i][j]= i+j;
}
void printMatrix(matrix a){
   for(int i=0; i<a.size(); i++)
   {
       for (int j=0; j<a[i].size(); j++)
           cout<<setw(4)<<a[i][j];
       cout<<endl;
```

```
    }
    cout<<endl;
}
```

მოქმედება ორგანზომილებიან ვექტორზე

აუცილებელი არაა ორგანზომილებიანი ვექტორის ელემენტები – ვექტორები იყვნენ ერთი ზომისა. შეგვიძლია შემოვიღოთ ისეთი ორგანზომილებიანი ვექტორები, რომლებიც შეესაბამება სხვადასხვა ფორმის ცხრილებს, მაგალითად სამკუთხა ან ე.წ. კბილოვან ცხრილებს, რაც ორგანზომილებოანი ვექტორის უპირატესობაა.

შემდეგ მაგალითში შევქმნით სამკუთხა ორგანზომილებიან ვექტორს `triangularMatrix`:

```
#include <iostream>
#include <vector>
using namespace std;
const int M =4;
int main()
{
    vector<vector <int> > triangularMatrix;
    for(int i = 0; i < M; i++)
    {
         vector<int> row;
         triangularMatrix.push_back(row);
    }
    for(int i=0; i<M; i++)
       for (int j=0; j<= i; j++)
           triangularMatrix[i].push_back(10*i+j);

    for(int i=0; i< triangularMatrix.size(); i++)
    {
        for (int j=0; j<triangularMatrix [i].size(); j++)
        {
           cout << triangularMatrix[i][j]<<"\t";
        }
        cout<<endl;
    }

    cout<<endl;
    return 0;
}
```

პროგრამის შესრულების შედეგია:

```
0
10      11
20      21      22
30      31      32      33
Press any key to continue . . .
```

შემდეგ მაგალითში კი იქმნება ორგანზომილებიანი ვექტორი `juggedMatrix`, რომლის ელემენტები – ვექტორები შეიცავენ სხვადასხვა რაოდენობის რიცხვებს:

```
#include <iostream>
#include <vector>
using namespace std;
const int M =4;
int main()
{
    vector<vector <int> > juggedMatrix;
    for(int i = 0; i < M; i++)
    {
```

```
        vector<int> row;
        juggedMatrix.push_back(row);
    }
    int count, number;
    for(int i=0; i<M; i++){
       cout<<"ramdeni ricxvia "<<i+1<<" veqtorSi? ";
        cin>> count;
        cout<<"SemoitaneT "<<count<<" mTeli ricxvi  ";
       for(int k=0; k<count; k++)
       {
             cin >> number;
            juggedMatrix[i].push_back(number);
        }
    }
    cout<<"\nSevqmeniT organzomilebiani veqtori \n\n";
    for(int i=0; i< juggedMatrix.size(); i++)
    {
        for (int j=0; j< juggedMatrix[i].size(); j++)
        {
           cout << juggedMatrix[i][j]<<"  ";
        }
        cout<<endl;
    }
    cout<<endl;
    return 0;
}
```

პროგრამის შესრულების შედეგია:

```
ramdeni ricxvia 1 veqtorSi? 3
SemoitaneT 3 mTeli ricxvi  110 5 -67
ramdeni ricxvia 2 veqtorSi? 5
SemoitaneT 5 mTeli ricxvi  2 7 9 0 28
ramdeni ricxvia 3 veqtorSi? 2
SemoitaneT 2 mTeli ricxvi  91 -4
ramdeni ricxvia 4 veqtorSi? 1
SemoitaneT 1 mTeli ricxvi  356

SevqmqniT organzomilebiani veqtori

110  5  -67
2  7  9  0  28
91  -4
356

Press any key to continue . . .
```

განვიხილოთ ორგანზომილებიან ვექტორზე მოქმედების რამდენიმე მაგალითი. ამჯერად საქმე გვექნება ე.წ. მართკუთხა ორგანზომილებიან ვექტორებთან, რომლებიც ერთი ზომის ვექტორებს შეიცავენ.

ამოცანის პირობაში ჩანაწერი A(M x N) ნიშნავს, რომ A ორგანზომილებიანი ვექტორი შეიცავს M ვექტორს, თითოეულს N ელემენტით.

ამოცანა 1. გაცვალეთ ორგანზომილებიანი Matrix(M x N) ვექტორის პირველი და ბოლო ვექტორი (ანუ სტრიქონი).

```
#include <iostream>
#include <iomanip>
#include <vector>
```

```
using namespace std;
const int M = 3, N = 4;
void fillMatrix(vector< vector <int> > & x);
void Transform(vector< vector <int> > & x);
void printMatrix(vector< vector <int> > x);
int main()
{
   vector <int> Row(N);
   vector< vector <int> > Matrix(M, Row);
   fillMatrix(Matrix);
   Transform(Matrix);
   cout<<"\nAfter transform matrix is:\n\n";
   printMatrix(Matrix);
   return 0;
}
void fillMatrix(vector< vector <int> > & x)
{
      for(int i=0; i<x.size(); i++)
        for(int j=0; j<x[i].size(); j++)
           cin >> x[i][j];
}
void printMatrix(vector< vector <int> > x){
   for(int i=0; i< x.size(); i++)
   {
      for(int j=0; j< x[i].size(); j++)
         cout <<setw(5)<<x[i][j];
      cout<<endl;
   }
   cout<<endl;
}
void Transform(vector< vector <int> > & x)
{
   swap( x[0], x[M-1] );
}
```

პროგრამის შედეგი:

```
1 2 3 4
5 6 7 8
9 10 11 12

After transform matrix is:

    9   10   11   12
    5    6    7    8
    1    2    3    4

Press any key to continue . . .
```

ამოცანა 2. ჩაწერეთ ორგანზომილებიან V(M x N) , M = 5, N = 3 ვექტორში შემთხვევითი რიცხვები [-12,42] შუალედიდან, შემდეგ იპოვეთ ვექტორის კენტი ელემენტების ჯამი და დაბეჭდეთ.

```
#include <iostream>
#include <iomanip>
#include <ctime>
#include <vector>
using namespace std;
const int M = 5, N = 3;
void fillMatrix(vector< vector<int> > & a);
void printMatrix(vector< vector<int> > a);
bool Odd(int x);
```

```
int sumOdds(vector< vector<int> > a);
int main()
{
   vector<int> row(N);
   vector< vector<int> > V(M, row);
   fillMatrix( V );
   printMatrix( V );
   int S = sumOdds( V );
   cout<<"Sum of odd elements = "<<S<<endl;
   return 0;
}
void fillMatrix(vector< vector<int> > & a)
{
      srand(time(NULL));
      for(int i=0; i<M; i++)
         for(int j=0; j<N; j++)
            a[i][j] = rand()%55 - 12;
}
bool Odd(int x)
{
   return x%2 == 1;
}
int sumOdds(vector< vector<int> > a)
{
   int sum =0;
   for(int i=0; i<M; i++)
      for(int j=0; j<N; j++)
         if( Odd(a[i][j]) )
            sum += a[i][j];
   return sum;
}
void printMatrix(vector< vector<int> > a)
{
   for(int i=0; i<M; i++)
   {
      for(int j=0; j<N; j++)
         cout <<setw(9)<<a[i][j];
      cout<<endl;
   }
   cout<<endl;
}
```

პროგრამა დაბეჭდავს

```
       29       30       -3
       33       17       37
       26       31        0
       32       28       28
        4       40       -6

Sum of odd elements = 147
Press any key to continue . . .
```

ამოცანა 3. A(M x N) ორგანზომილებიან ვექტორში ნამდვილი რიცხვებია. დაწერეთ ფუნქცია, რომელიც დააბრუნებს ვექტორის უდიდესი და უმცირესი ელემენტების ჯამს. რიცხვები ვექტორში ჩაწერეთ `reals.in` ფაილიდან. ამისათვის ასევე დაწერეთ ფუნქცია. პროგრამაში გამოიყენეთ ორივე ფუნქცია და დაბეჭდეთ შედეგი.

```cpp
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
using namespace std;
const int M = 3, N = 4;
typedef vector<double> row;
typedef vector< row > matrix;
void fillMatrix(matrix & x);
void printMatrix(matrix x);
double maxminSum(matrix x);
int main(){
   row R(N);
   matrix A(M, R);
   fillMatrix(A);
   printMatrix(A);
   double sum = maxminSum(A);
   cout<<"Udidesi da umciresi ricxvebis jami = "<<sum<<endl;
   return 0;
}
void fillMatrix(matrix & x)
{
    ifstream fin("reals.in");
    for(int i=0; i<x.size(); i++)
       for(int j=0; j<x[i].size(); j++)
          fin >> x[i][j];
}
void printMatrix(matrix x){
   for(int i=0; i< x.size(); i++)
   {
      for(int j=0; j< x[i].size(); j++)
         cout <<setw(10)<<x[i][j];
      cout<<endl;
   }
   cout<<endl;
}
double maxminSum(matrix x){
   double min, max;
   min = max = x[0][0];
   for(int i=0; i<x.size(); i++)
      for(int j=0; j<x[i].size(); j++)
         if(x[i][j] > max)
            max = x[i][j];
         else if(x[i][j] < min)
                  min = x[i][j];
   return max + min;
}
```

პროგრამის შესრულების შედეგია:

reals.in ფაილი	გამოტანის ეკრანი
5.8 12.25 -33.75 0.0 250.5 150.45 10.0 -150.25 324.5 91.0 0.0 -1.75	5.8 12.25 -33.75 0 250.5 150.45 10 -150.25 324.5 91 0 -1.75 Udidesi da umciresi ricxvebis jami = 174.25 Press any key to continue . . .

თავი 11: მონაცემთა ახალი ტიპის შექმნა - კლასი

- მონაცემთა ახალი ტიპების შექმნა
- განაცხადი კლასზე
- ტერმინოლოგიური შეთანხმებები
- კონსტრუქტორები და განაცხადები ობიექტებზე
- პოინტერი კლასის ობიექტზე

მონაცემთა ახალი ტიპების შექმნა

პროფესიონალი პროგრამისტები პროგრამებს წერენ რეალური სამყაროს პრობლემების გადასაჭრელად. მაგალითად, რაიმე დაწესებულებაში დასაქმებულების მონაცემების დასამუშავებლად, რაიმე მოწყობილების მოდელის შესადგენად და ფუნქციონირების სიმულირებისთვის და ა. შ..

ასეთ შემთხვევებში ძალიან მოსახერხებელია, რომ კონკრეტული ამოცანის დაპროგრამების პროცესში ყოველი ობიექტისთვის, რომლის შესწავლას და დამუშავებასაც ვაპირებთ, შევქმნათ შესაბამისი პროგრამული ეკვივალენტი (წარმოდგენა) და განვახორციელოთ საჭირო მოქმედებები მათზე.

ჩვენ უკვე განვიხილეთ მონაცემთა უმარტივესი ტიპები (**int, double, char, bool** და სხვა), აგრეთვე შედარებით რთული ტიპები (**string, vector**). ნებისმიერი მათგანისთვის, განაცხადი ამომწურავ ინფორმაციას აძლევს კომპილერს: რა მოცულობის მეხსიერება გამოყოს ამ ტიპის ყოველი წარმომადგენლისთვის (ცვლადისთვის ან ობიექტისთვის), რა მოქმედებები არის ნებადართული ამ ტიპის წარმომადგენლებისთვის და რა საწყისი ინფორმაცია იწერება მათში (ინიციალიზაციის პროცესში).

დაწყებული **string** და **vector** ტიპებიდან, როგორც წესი, აღარ იყენებენ ტერმინს ცვლადი. იგივეა სამართლიანი მომხმარებლის მიერ შექმნილი ტიპებისთვის, რაც კლასის საშუალებით განხორციელდება. მიზეზი ძალიან მარტივია: მარტივი ტიპი ნიშნავს, რომ ამ ტიპის ცვლადი არის რაღაც განუყოფელი, მარტივი, რომელიც არ არის შედგენილი სხვა, უფრო მარტივი ტიპის ცვლადებისგან. ეს რაღაც ძალიან გავს ქიმიური ელემენტების სიტუაციას, სადაც არის უმარტივესი არაორგანული ნივთიერებები და არის შედგენილი, ორგანული ნაერთები.

string და **vector** ტიპების თითოეული წარმომადგენელი, რომელსაც უკვე ობიექტს ვუწოდებთ, შედგება უფრო მარტივი ტიპის არაერთი ცვლადისგან. მაგალითად, შემდეგი განაცხადით

```
vector<int> v(10);
```

გაკეთებული v ობიექტი შედგება 10 მთელი ტიპის ცვლადისგან. მაგრამ **string** და **vector** ტიპებიც მარტივია იმ აზრით, რომ მათი ობიექტები შედგება მხოლოდ ერთი რომელიმე ტიპის რამდენიმე ცვლადისგან (ან რამდენიმე ობიექტისგან, როგორც ვექტორების ვექტორის შემთხვევაში).

სამაგიეროდ, ამ ორ ტიპიდან თითოეულს (კლასს) უკვე აქვს შესაძლებლობა, რომ მათმა ობიექტებმა მიმართონ სპეციალურ ფუნქციებს, რომლებსაც მეთოდებს ვუწოდებთ და რომლებიც სპეციალურად ამ კლასისთვისაა შექმნილი ან იმპლემენტირებული.

C++ ენა პროგრამისტს აძლევს საშუალებას, რომ მან შექმნან პრაქტიკულად ნებისმიერი რეალური ობიექტის შესაბამისი პროგრამული ობიექტი, რომელიც შედგება სხვა, უფრო მარტივი ობიექტებისგან, ან ამავე კლასის ობიექტის მისამართებისგან, და რომელშიც გარდა ობიექტებისა, წევრებად შედის ფუნქციები (მეთოდები). ისევე როგორც ჩვენთვის ნაცნობი ტიპების შემთხვევაში, კლასის ობიექტი კომპილერს აცნობებს თუ:

- რა ზომის მეხსიერება სჭირდება ობიექტს;
- რა ინფორმაციის შენახვა შეუძლია მას;
- რა მოქმედებები შეიძლება განხორციელდეს მათზე.

განაცხადი კლასზე

როდესაც C++ ენაში ახალი კლასი იქმნება, როგორც წესი, კლასზე განაცხადი კეთდება თავსართ `.h (header)` ფაილში, ხოლო კლასის იმპლემენტაცია, ანუ სრული განსაზღვრა ხდება `.cpp` ფაილში, როგორც ეს გავაკეთეთ მე–13 ლაბორატორიულ მეცადინეობაზე.

სიმარტივისთვის, ზოგჯერ განაცხადს და იმპლემენტაციას მოვათავსებთ იმავე ფაილში. რომელშიც არის მთავარი `main()` ფუნქცია.

ცხადია, ჯერ კეთდება განაცხადი კლასზე, შემდეგ შეგვიძლია შევქმნათ ამ კლასის ობიექტები.

კლასზე განაცხადის გასაკეთებლად ვიყენებთ გასაღებ სიტყვას **`class`**, რომელსაც მოყვება კლასის (ახალი ტიპის) სახელი, ხოლო შემდეგ ფიგურული ფრჩხილების წყვილში ვათავსებთ განაცხადებს წევრ ცვლადებზე და ფუნქციებზე.

როგორც სხვა ნებისმიერი განაცხადი, კლასის განაცხადი უნდა დამთავრდეს წერტილ-მძიმით. მაგალითად:

```
class Circle
{
  public:
     double radius;
     Circle(double value);
     Circle();
     double Area();
     double Perimeter();
};
```

ყურადღება მივაქციოთ, რომ ამ კლასის ორ მეთოდს არ გააჩნია დასაბრუნებელი მნიშვნელობა: ესაა ორი კონსტრუქტორი. უფრო მეტი, ამ მეთოდების სახელი ემთხვევა კლასის სახელს. დასაბრუნებელი მნიშვნელობის მითითება არ არის საჭირო კიდევ ერთი ტიპის ფუნქციისთვის, რომლებსაც მეორე სემესტრში შევისწავლით დაწვრილებით, და რომლებსაც დესტრუქტორები ეწოდებათ.

კლასის მეთოდების იმპლემენტაციას, როდესაც ეს კლასის გარეთ ხდება (და ასეც უნდა იყოს, გარდა უმარტივესი შემთხვევებისა), აქვს თავისი წესი: იწერება კლასის სახელი, უშუალოდ მას მოყვება ოთხი წერტილი და შემდეგ მეთოდის განსაზღვრა (იმპლემენტაცია). მაგალითად:

```
Circle::Circle(double value){
    radius = value;
}
```

კლასის ობიექტზე განაცხადის გაკეთებისთვის საჭიროა კლასში დამუშავებული (ინკაფსულირებული) იყოს ისეთი კონსტრუქტორი ან კონსტრუქტორები, რომლებიც საჭიროა ობიექტების ინიციალიზაციისთვის.

ტერმინოლოგიური შეთანხმებები

C++ ენაში ყველაფერი უნდა გავაკეთოთ მხოლოდ პროგრამირების კარგი სტილის შესაბამისად. კერძოდ, ნებისმიერი სახელი მაქსიმალურად შინაარსიანი უნდა იყოს და მიგვანიშნებდეს შესაბამისი კლასის, ობიექტის თუ ცვლადის დანიშნულებაზე. მაგალითად, `Cat, Rectangle, Employee` არის კარგი სახელები კლასებისთვის. რაც შეეხება კლასის წევრ ცვლადებს, ბევრი

პროგრამისტი იყენებს მათთვის პრეფიქსს `its`. მაგალითად, `itsAge, itsWeight, itsSpeed, itsRadius`.
ამით ხაზი ესმება განსხვავებას წევრ ცვლადებსა და არაწევრ ცვლადებს შორის. სხვა პროგრამისტები იყენებენ განსხვავებულ პრეფიქსებს. ზოგი წერს `myAge, myWeight, mySpeed, myRadius`. ზოგიც უბრალოდ იყენებს ასოს `m`, ან `m_` (წევრებისთვის) და წერენ `mAge, mWeight, mSpeed, mRadius` ან `m_age, m_weight, m_speed, m_radius`. კლასების სახელებს პროგრამისტების უმეტესობა წერს მთავრული (დიდი) ლათუნური ასოთი.

ყველა გავრცელებული სტილის აღნიშვნა პრაქტიკულად შეუძლებელია. მით უმეტეს, რომ ძლიერ და ზრდად კომპანიებში მიღებული ტენდენციაა, რომ სტილებთან დაკავშირებით იქონიონ შინაური სტანდარტები. ეს აადვილებს ასეთი კომპანიის დეველოპერების საქმეს - ისინი ადვილად კითხულობენ თავისი კოლეგების დაწერილ კოდს. ეს ნიშნავს, რომ C++ ენაზე მომუშავე პროგრამისტები მზად უნდა იყონ იმუშავონ განსხვავებულ სტილებთან.

კონსტრუქტორები და განაცხადები ობიექტებზე

C++ ენის კარგი სტილის ერთ-ერთი თანამედროვე ტენდენცია გულისხმობს, რომ, თუ ეს შესაძლებელია, ცვლადის/ობიექტის ინიციალიზაცია სასურველია მოხდეს განაცხადის გაკეთბის მომენტში.

ამით აიხსნება ის დიდი ყურადღება, რაც ეთმობა კონსტრუქტორების განსაზღვრის საკითხს.

ფორმალურად, კონსტრუქტორის განსაზღვრა არაა სავალდებულო. მაგალითად, თუ ზემოთ აღწერილ წრის კლასის განაცხადს გადავაკეთებდით ასე:

```
class Circle
{
  public:
     double radius;
     double Area();
     double Perimeter();
};
```

მაშინ კომპილერი თვითონ შექმნის უპარამეტრო კონსტრუქტორს, რომელიც შექმნის განუსაზღვრელ რადიუსიან წრეს, ხოლო განაცხადის გაკეთების მომენტში ობიექტის ინიციალიზება შეუძლებელია.

მაგალითად, მთავარ ფუნქციაში, ჩანაწერი

```
Circle c;
```

ერთადერთი გზაა ამ კლასის ობიექტზე განაცხადის გაკეთებისა. თუ გვინდა განსაზღვრული მნიშვნელობა მივანიჭოთ რადიუსს, უნდა შევქმნათ შესაბამისი შეტყობინება, მაგალითად

```
c.radius = 11;
```

თუ გვინდა ინიციალიზაცია მოხდეს განაცხადის გაკეთების მომენტში, როგორც ეს კეთდებოდა მარტივი ცვლადებისთვის

```
double x(11.3);
int n(1853);
```

და ა. შ., კლასში უნდა გვქონდეს შესაბამისი კონსტრუქტორი, რომელიც (რადგან კონსტრუქტორის სახელი ემთხვევა კლასის სახელს) გააქტიურდება კლასის სახელისა და ობიექტის შემდეგ ფრჩხილების გამოჩენისთანავე. მაგალითად, თუ წრის კლასზე განაცხადს მივცემთ სახეს:

```
class Circle
{
  public:
     double radius;
     Circle(double value);
     double Area();
     double Perimeter();
};
```

და კონსტრუქტორის იმპლემენტაციას მოვახდენთ შემდეგნაირად:

```
Circle::Circle(double value)
{
    radius = value;
}
```

მაშინ განაცხადი

```
    Circle c(111);
```

ხდება ვალიდური: იგი შექმნის წრის ობიექტს, სახელად c, და ამ წრის რადიუსს გაუტოლებს 111-ს.

ყურადღება მივაქციოთ იმ გარემოებას, რომ ჩვენ უნდა შევქმნათ ყველა ის კონსტრუქტორი, რომლის გამოყენებასაც ვაპირებთ. ბოლო შემთხვევაში, იმის გამო რომ ჩვენ აღვწერეთ ცხადი (ანუ პარამეტრიანი) კონსტრუქტორი, კომპილერი აღარ ქმნის უპარამეტროს და ამიტომ ასეთი შეტყობინება:

```
    Circle x;
```

უკვე შეცდომაა. ამიტომ, თუ გვინდა ასეთი განაცხადიც გამოვიყენოთ, კლასის განაცხადში უნდა იყოს უპარამეტრო კონსტრუქტორიც, როგორც ეს იყო სულ პირველ მაგალითში.

კონსტრუქტორების თემა იმდენად მდიდარია და მნიშვნელოვანი, რომ ასეთ მარტივ მაგალითშიც შეგვიძლია განვავითაროთ და შევქმნათ განსხვავებული და შინაარსიანი კონსტრუქტორები.

დავუშვათ, გვინდა რომ პროგრამაში განაცხადი გავაკეთოთ წრის ობიექტებზე, მათგან ზოგიერთის მონაცემებს შევიყვანთ ფაილიდან, ზოგიერთის მონაცემს კლავიატურიდან, ხოლო ზოგიერთის ინიციალიზებას მოვახდენთ განაცხადის გაკეთბის მომენტში, პარამეტრიანი კონსტრუქტორით.

```
#include <iostream>
#include <fstream>
using namespace std;

class Circle
{
  public:
    double radius;
    Circle(double value);
    Circle();
    Circle(ifstream & );
    Circle(istream & );
    double Area();
    void showCircle(void);
};
Circle :: Circle(double value){
```

```
    radius = value;
}
Circle :: Circle() {radius = 0.0; }
Circle :: Circle(ifstream & x ){
   x >> radius;
}
Circle :: Circle(istream & x ){
   cout<<"ShemoitaneT wris radiusi: ";
   x >> radius;
}

double Circle :: Area(){
   return 3.1416 * radius * radius;
}
void Circle :: showCircle(){
   cout << "Radius: " << radius <<endl;
   cout << "Area: " << Area() <<endl;
   cout << endl;
}

int main()
{
   Circle C1(2.4);
   C1.showCircle();

   Circle C2;
   C2.showCircle();

   ifstream fin("Circles.txt");
   cout<<"\nInformacia radiusis Sesaxeb Semodis failidan\n\n";

   Circle A(fin);
   A.showCircle();

   Circle B(fin);
   B.showCircle();

   Circle F(cin);
   F.showCircle();

   system("pause");
   return 0;
}
```

Circles.txt	გამოტანის ეკრანი
25 2 123 15.1 9.8	Radius: 2.4 Area: 18.0956 Radius: 0 Area: 0 Informacia radiusis Sesaxeb Semodis failidan Radius: 25 Area: 1963.5

	Radius: 2 Area: 12.5664 ShemoitaneT wris radiusi: 1 Radius: 1 Area: 3.1416 Press any key to continue . . .

წვდომა კლასის წევრებზე, ინტერფეისი

ყველა წევრი, რაც წერია გასაღები სიტყვა **public**: -ის შემდეგ, არის ღია ნებიმიერი ობიექტისთვის, რაც ნიშნავს, რომ

```
Circle c(111);
```

შეტყობინების შემდეგ ნებადართულია შემდეგი სახის შეტყობინებები:

```
...
c.radius = 21;
...
cout << c.Area() << endl;
```

ასეთი რამე მხოლოდ ძალიან მცირე ზომის კლასებისთვის არის დასაშვები და ისიც არა ყოველთვის.

დიდი ზომის, რთული კლასის შემთხვევაში, მეთოდების და ველების უმეტესობა ტექნიკური ხასიათისაა, და ძირითადი მოქმედებები, რაც უნდა განხორციელდეს კლასზე, არ მოითხოვს რომ მისი ყველა წევრი იყოს ღია.

მაგალითად, ეს იმის ანალოგიურია, რომ პლეერის ინტერფეისი შედგება რამდენიმე ღილაკისგან (გადამრთველები და მარეგულირებლები) და ეს საკმარისია მისი გამოყენებისთვის, მაშინ როდესაც ამ ღილაკების გარდა იგი ბევრ სხვა დეტალს შეიცავს, რომლებზეც წვდომა არ გვაქვს. ანუ, პლეერის გამართული და კორექტული ფუნქციონირებისთვის უმჯობესია, რომ მისი შიგნეულობა დახურულია და მის მახასიათებლებზე წვდომა ხდება ღილაკების საშუალებით. ამის ანალოგიურად, კლასის გამართული ფუნქციონირებისთვის აუცილებელია, რომ მის წევრებზე წვდომა ხორციელდებოდეს არა უშუალოდ (მაგ. `c.radius = 21;`), არამედ გაშუალებულად, სპეციალურად ამისთვის შექმნილი მეთოდების საშუალებით. ღია წევრები წარმოადგენენ კლასის ინტერფეისს (იგივეს რაც ღილაკია პლეერისთვის).

ამ საკითხებს შემდეგში დიდი ყურადღება დაეთმობა ობიექტზე-ორიენტირებული პროგრამირების კურსში.

პოინტერი კლასის ობიექტზე

რაც უფრო დიდი არის კლასი, უფრო მეტი მნიშვნელობა ენიჭება მისი ობიექტების პოინტერების გამოყენებას, რადგან პოინტერი იკავებს მხოლოდ 4 ბაიტს, ხოლო ობიექტმა შეიძლება მეგაბაიტები დაიკავოს. უფრო კონკრეტულები რომ ვიყოთ, განვიხილოთ ასეთი ამოცანა. ვთქვათ ფაილში `"volcanoes.txt"` წერია სამი ვულკანის მონაცემი – სახელი, სიმაღლე და ადგილმდებარეობა. მაგალითად :

```
vezuv                   1277      italy
kluchevskaia_sopka      4750      kamchatka
fuziama                 3776       japan
```

ჩვენი ამოცანაა ვიპოვოთ ყველაზე მაღალი ვულკანი და დავბეჭდოთ მისი მონაცემები.

ამ ამოცანის ამოსახსნელად ჯერ გამოვიყენოთ იგივე გზა, რაც გვქონდა პრაქტიკულ მეცადინეობაზე, ოღონდ გავითვალისწინოთ აგრეთვე ფაილიდან ობიექტების კონსტრუირების ზემოთ აღნიშნული შესაძლებლობა. შემდეგ, იგივე ამოცანა გავაკეთოთ უკვე პოინტერის გამოყენებით და შევაფასოთ განსხვავება.

```
#include <iostream>
#include <string>
#include <fstream>
using namespace std;

class Volcano
{
  public:
    string name;
    int height;
    string location;
    Volcano(ifstream & );
     void printVolcano(void);
};
Volcano :: Volcano(ifstream & x ) {
   x >> name >> height >> location;
}
void Volcano :: printVolcano(){
         cout<< name <<", its height is " << height <<" meters"<<endl
                <<"It is located in "<< location<<endl;
       }
int main()
{
   ifstream fin("volcanoes.txt");

   Volcano A(fin);
   Volcano B(fin);
   Volcano C(fin);
   Volcano maxVolcano(A);

   if(maxVolcano.height < A.height)
           maxVolcano = A;
   if(maxVolcano.height < B.height)
           maxVolcano = B;
   cout<<"Maximum height has volcano\n";
   maxVolcano.printVolcano();
   system("pause");
   return 0;
}
```

შედეგად ვღებულობთ:

```
Maximum height has volcano
kluchevskaia_sopka, its height is
4750 meters
It is located in kamchatka
Press any key to continue . .  .
```

ამ პროგრამაში, ჩვენ სამჯერ (ერთხელ შეტყობინებაში `Volcano maxVolcano(A);` ორჯერ `if` შეტყობინების ტანში) მივანიჭეთ ობიექტს სხვა ობიექტის მნიშვნელობა და განაცხადი გავაკეთეთ 4 ობიექტზე. ობიექტები მინიჭება შრომატევადი ოპერაციაა, რაც უფრო გრძელია ობიექტი, მით უფრო მეტი დრო არის საჭირო მის შესასრულებლად, და მით უფრო მეტი ადგილია საჭირო მეხსიერებაში მისი განთავსებითვის.

ახლა განვიხილოთ განსხვავებული მიდგომა, რომ განაცხადი გავაკეთოთ სამ ობიექტზე, და დავიმახსოვროთ არა ობიექტი, არამედ მისი მისამართი. მისამართიდან ობიექტის წევრებთან წვდომა ხდება ისრების საშუალებით. მთავარ ფუნქციას ექნება სახე:

```
int main()
{
   ifstream fin("volcanoes.txt");

   Volcano A(fin);
   Volcano B(fin);
   Volcano C(fin);
   Volcano* addressOfMmax(&A);

   if(addressOfMmax->height < B.height)
           addressOfMmax = &B;
   if(addressOfMmax->height < C.height)
           addressOfMmax = &C;
   cout<<"Maximum height has volcano\n";
   addressOfMmax->printVolcano();
   system("pause");
   return 0;
}
```

შედეგი იგივეა რაც ზემოთ. განსხვავება ისაა, რომ ჩვენ განაცხადი გავაკეთეთ სამ ობიექტზე და ერთ პოინტერზე (ტიპის მიუხედავად, ნებისმიერი პოინტერი მეხსიერებაში იკავებს მხოლოდ 4 ბაიტს), და სამჯერ პოინტერს მივანიჭეთ სხვა პოინტერის მნიშვნელობა. ოთხბაიტიანი ცვლადების მნიშვნელობების მინიჭება გაცილებით ეკონომიურია, ვიდრე ობიექტებისთვის მნიშვნელობების მინიჭება. მიდგომებს შორის განსხვავება განსაკუთრებით თვალსაჩინო ხდება დიდ ამოცანებში, სადაც დიდი რაოდენობა დიდი ზომის ობიექტები მონაწილეობს. ასეთ შემთხვევებში, როცა შესაძლებელია, ჯობია ვიმუშავოთ ობიექტების მისამართებთან, იმ პირობით რომ დაცული იქნება უსაფრთხოების გარკვეული წესები.

პრაქტიკული მეცადინეობა № 1

პრაქტიკული მეცადინეობის თემები:

- გამოსახულების ჩაწერა C++-ზე
- გამოსახულების ტიპის განსაზღვრა
- მონაცემთა ძირითადი ტიპები
- არითმეტიკული ოპერატორები
- გამოსახულების მნიშვნელობის გამოთვლა და ტიპის დადგენა.

მასალა დამოუკიდებლად გაცნობისათვის:

1. რიცხვებისა და გამოსახულებების (ცხრილის მარცხენა სვეტი) C++-ზე ჩაწერის (ცხრილის მარჯვენა სვეტი) ნიმუშები:

№	გამოსახულება	გამოსახულება C++-ზე
1	0, 3845	`0.3845`
2	$\frac{1}{6}$	`0.166667`
3	$-\frac{18}{4}$	`-4.5`
4	-257,325	`-257.325`
5	0,0000568	`5.68e-005`
6	$3a-72b$	`3*a-72*b`
7	$-9xy$	`-9*x*y`
8	$0,6n+4,25$	`0.6*n+4.25`
9	$2b^2+3c-1$	`2*b*b+3*c-1`
10	$-8m^3+4m^2-5m+6$	`-8*m*m*m+4*m*m-5*m+6`
11	$1,25ay^2+0,8(b-y)$	`1.25*a*y*y +0.8*(b-y)`
12	$9a^2-4ab+5b^2$	`9*a*a-4*a*b+5*b*b`
13	$(5k+2n)(k-4,85n)$	`(5*k+2*n)*(k-4.85*n)`
14	$(a-b)^2$	`(a-b)*(a-b)` ან `a*a-2*a*b+b*b`
15	$(x-y)-7(x+y)^3$	`(x-y)-7*(x+y)*(x+y)*(x+y)`
16	$\frac{a-0,184b}{ab}$	`(a-0.184*b)/(a*b)`
17	$\left[12z-7(x+y)+1\right]\cdot\frac{y}{z}$	`(12*z-7*(x+y)+1)*y/z`
18	$\frac{ab}{b+c}-\frac{a+d}{a+b}$	`a*b/(b+c)-(a+d)/(a+b)`

2. C++-ზე ჩაწერილი გამოსახულებების (ცხრილის მარცხენა სვეტი) ჩვეულებრივი ფორმით ჩაწერის ნიმუშები:

№	გამოსახულება C++-ზე	გამოსახულება
1	`0.25*x*x+4*x*y+3.7*y*y`	$0,25x^2+4xy+3,7y^2$
2	`k*m/(5*m-2*k)`	$\frac{km}{5m-2k}$
3	`(a+c)/x*y+z/(b+x)`	$\frac{(a+c)y}{x}+\frac{z}{b+x}$

4	`d/(a*b)+(b+a)/d`	$\frac{d}{ab}+\frac{b+a}{d}$
5	`d/a*b+(b+a)/d`	$\frac{db}{a}+\frac{b+a}{d}$
6	`(a*b/5-27*(c+d))/(0.43*n)`	$\frac{\frac{ab}{5}-27(c+d)}{0{,}43n}$

3. ზოგიერთი საჭირო ფორმულა:
   1) სამკუთხედის ფართობი: $S=\frac{ah}{2}$, სადაც a სამკუთხედის გვერდია, h კი მასზე დაშვებული სიმაღლე.
   2) წრის ფართობი: $S=\pi R^2$, სადაც R წრის რადიუსია.
   3) წრეწირის სიგრძე: $L=2\pi R$.
   4) მართკუთხედის ფართობი და პერიმეტრი: $S=ab$ და $P=2(a+b)$, სადაც a და b მართკუთხედის გვერდებია.
   5) კვადრატის ფართობი და პერიმეტრი: $S=x^2$ და $P=4x$, სადაც x კვადრატის გვერდია.

სააუდიტორიო სამუშაო:

1. ჩაწერეთ C++-ზე შემდეგი რიცხვები და გამოსახულებები:
   ა) `1542; -67281;` $-\frac{17}{4}$; $\frac{29}{5}$;
   ბ) `36,2247; -0,0000892` - ფიქსირებული წერტილით და სამეცნიერო ფორმით
   გ) $3a+4b+5{,}75$; $-9(x+y)-2{,}3^2$; $5c^2-82ab^3+a^2$; $0{,}75(k-n)+36(m+n)$; $x[\,23(x+y)-3{,}5(y-x)\,]$;
   დ) $\frac{a-b}{15}$; $\frac{-u+3{,}4z}{uz}$; $\frac{y}{x}(n-k)-\frac{73x}{24y}$; $\frac{k(x-y)+m(x+y)}{(y-x)(k+m)}$; $\frac{-\frac{c+d}{3}-8cd}{d-c}$.

2. C++-ზე ჩაწერილი გამოსახულებები ჩაწერეთ ჩვეულებრივი ფორმით:
   ა) `-4.38*a+123.5*b;` ბ) `1/(x*y)-7*y*y;` გ) `(n-m)/(2*m+1.5*n);`
   დ) `0.009*(2*(a+b)-a*b*b);` ე) `-d+2*(3*(a+b)+c);`
   ვ) `5.3/(1.9*(a-c)+d))+b;` ზ) `-36*x/(y*y+x*y+x*x);`
   თ) `u/(z-0.3*u)-0.05/u*z;` ი) `9.2*(k-2*n)/k-7.9/((n+1)*(n-2)+3).`

3. გააკეთეთ განაცხადი C++-ის ზოგიერთი ტიპის ცვლადზე და განაცხადშივე მოახდინეთ ცვლადების ინიციალიზება (საწყისი მნიშვნელობების მინიჭება):

```
string name = "Giorgi";
int age = 17;
double consult_time = 15.30;
char decimal_point = '.';
bool is_odd = true;
```

```
string name("Giorgi");
int age(17);
double consult_time(15.30);
char decimal_point('.');
bool is_odd(true);
```

4. ცვლადების მნიშვნელობების მიხედვით განსაზღვრეთ მათი ტიპი
   ა) `0.000745;` ბ) `'A';` გ) `"I'm a first year student";` დ) `'?';`

ე) `false;` ვ) `-3275;` ზ) `true;` თ) `492.0275;`
ი) `"The C++ Programming Language";` კ) `9.`

5. გამოთვალეთ C++-ზე ჩაწერილი შემდეგი გამოსახულებების მნიშვნელობები:

ა) `n/2,` თუ მოცემული გვაქვს განაცხადი `int n = 7;`
ბ) `k/2,` თუ მოცემული გვაქვს განაცხადი `double k = 7;`
გ) `7%2;` დ) `-12%3;` ე) `-(10%4).`

6. განსაზღვრეთ C++-ზე ჩაწერილი გამოსახულების ტიპი (მთელი თუ ნამდვილი)

ა) `(4-9)*5;` ბ) `3+2.5;` გ) `25/4;` დ) `25./4;` ე) `19%4;` ვ) `-1.5*(7+3).`

დამოუკიდებელი სამუშაო:

1. ჩაწერეთ C++-ზე შემდეგი რიცხვები და გამოსახულებები:

1) ა) `-6750;` ბ) `-π;` გ) `0,04;` დ) `921,054;` ე) `-0,000075;`
2) ა) `10x;` ბ) `y+48;` გ) `3,25a-14b;` დ) `42p^2-8;` ე) `c^3;`
3) ა) $\frac{b-a}{3}$; ბ) `(x+y)^2;` გ) c^2+cd+d^2; დ) `-5ax^2+4,25bx-3c;`
4) ა) $\frac{k}{mn}+\frac{mn}{k}$; ბ) $\frac{a-b}{y_1}+\frac{y_2}{a+b}$; გ) $\left(x_1^2+y_1^2\right)\left(x_2^2-y_2^2\right)$; დ) $\frac{c+b}{9a}(b-c)$;

ე) $\left[3(p-q)+5(p+q)\right]z$; ვ) `2,7{x+[(x-2y)^2+5,09y]};` ზ) $\frac{(a+1)^2}{a^2-1}$;

5) ა) $\frac{1-2y+y^2}{y^3+1}$; ბ) $\frac{1}{ac+3b-ab}$; გ) $\frac{\frac{1}{n}+\frac{1}{m}}{n-m}$; დ) $10\frac{x+y}{\frac{1}{x}+\frac{1}{y}}$; ე) $-\frac{a-c}{b+d}-\frac{bc}{a+c}$;

6) ა) $\frac{a-b}{b}+2\frac{c+d}{d}+\frac{ad-bc}{cd}$; ბ) $7\frac{(y-z)^2}{y^2}+9\frac{(z+y)^2}{yz^2}$; გ) $\frac{m+\frac{n}{m-n}}{5-\frac{m+n}{m}}$.

2. C++-ზე ჩაწერილი გამოსახულებები ჩაწერეთ ჩვეულებრივი ფორმით:
ა) `k*n/(n+k);` ბ) `x-y/3*z;` გ) `a*b/(c-a)*(b+c);`
დ) `(12*t*(x+y)-z*p)/d;` ე) `m/(6*(k-n)+(3*m-n)/k);`
ვ) `(5.4*(a*a+b)-d)/b*b*d;` ზ) `0.06*(a-b)*(a-b)/((a+b)*(a+b)-1.25);`
თ) `(c/(b+d)-8*a)/(a/(d-b)+b/a*c).`

3. ჩაწერეთ C++-ზე ფორმულები (იხ. **მასალა დამოუკიდებლად გაცნობისათვის**, პუნქტი 3)

4. ცვლადების მნიშვნელობების მიხედვით განსაზღვრეთ მათი ტიპი

ა) `'a';` ბ) `-27;` გ) `true;` დ) `"I study C++";` ე) `-101.0037;` ვ) `'9';`
ზ) `false;` თ) `50074;` ი) `3.14159;` კ)`"Bjarne Stroustrup".`

5. განსაზღვრეთ C++-ზე ჩაწერილი გამოსახულებების მნიშვნელობა და ტიპი:
ა) `(3+7.)*12;`
ბ) `c/5+8.35,` თუ `c` მთელი რიცხვია და უდრის `15`-ს;
გ) `(14%3+2)/5;`
დ) `(a+b+c)/3,` თუ მოცემული გვაქვს აღწერა `int a=2, b=1, c=4;`
ე) `(m+n)/2.0,` თუ მოცემული გვაქვს აღწერა `int n=3, m=4;`
ვ) `14-x/4,` თუ `x` ნამდვილი რიცხვია და უდრის `9`-ს.

ლაბორატორიული მეცადინეობა № 1

ლაბორატორიული სამუშაოს გეგმა:

1. დაპროგრამების ინტეგრირებული გარემოს გაცნობა (20 წთ)
2. მარტივი ამოცანების აკრეფა, გამართვა, შესრულება, დამახსოვრება (შენახვა) (30 წთ)

დაპროგრამების ინტეგრირებული გარემოს გაცნობა

- გააქტიურეთ Visual C++ 2010-ის გარემო.
- შექმენით ახალი პროექტი (გზა: Start Page -> New Project -> Win32, და Win32-ში გაავააქტიუროთ Win32 Console Application), Name ველში აკრიფეთ პროექტის სახელი, მაგალითად A და დააჭირეთ OK ღილაკს (ან Enter -ს). გამოსულ Win32 Application Wizard ფანჯრის Next ღილაკზე დაჭერით გადადით Application Settings ფანჯარაში, მონიშნეთ თვისება Empty Project და დააჭირეთ ღილაკს Finish.
- ჩამოშალეთ View მენიუ და აირჩიეთ Solution Explorer. გახსნილ Solution Explorer ფანჯარაში გავხსნათ Source Files საქაღალდის კონტექსტური მენიუ (თაგვის მარჯვენა ღილაკით). აირჩიეთ Add -> New Item -> C++ File(.cpp), Name ველში აკრიფეთ პროგრამის სახელი, მაგალითად prog1 და დააჭირეთ Add ღილაკს (ან Enter -ს).
- გაიხსნება ტექსტური ფანჯარა სახელით prog1.cpp, რომელშიც უნდა აკრიფოთ თქვენი პროგრამის კოდი.
- აკრეფილი კოდის გაშვებისთვის: build მენიუში ირჩევთ build solution, შემდეგ, თუ შეცდომების სია ცარიელია, მიღებული შედეგების ეკრანზე გამოტანისთვის, Debug მენიუს ჩამონათვალში აირჩიეთ Start Without Debugging. კომპილერის კითხვაზე, გვინდა თუ არა პროგრამის გამართვა, უპასუხეთ დადებითად. შედეგად მიიღებთ პასუხს შავ ფანჯარაში (კონსოლში).
  შენიშვნა: პროგრამის გაშვება შეიძლება სხვადასხვანაირად. თუ რომელიმე სხვა გზას იყენებთ, მაშინ პროგრამაში უნდა გააქტიუროთ დაკომენტირებული //system("PAUSE"); შეტყობინება.

ლაბორატორიული სამუშაო:

ამოცანა 1: დაბეჭდეთ ეკრანზე თქვენი სახელი და გვარი. პროგრამას დაურთეთ საწყისი კომენტარები.

შესაბამის C++-პროგრამას აქვს სახე:

```
//////////////////////////////////////////
//    ავტორი:
//    პროგრამა: სტუდენტის ვინაობის ბეჭდვა
//    თარიღი: 06/20/2011
//////////////////////////////////////////
#include <iostream>
using namespace std;
int main()
{
   cout<<"me var . . .\n";
   //system("PAUSE");
   return 0;
}
```

დავალება:

ა) შეიტანეთ პროგრამაში ცვლილება: დაბეჭდეთ სახელი, გვარი და მისამართი.

ბ) სახელი, გვარი და მისამართი დაბეჭდეთ ცალ-ცალკე სტრიქონზე.

გ) ჩასვით გამოსატანი ტექსტის – მისამართის – ბოლოში და შემდეგ შუაში რიგრიგობით სიმბოლოები: `\n, \a, \t, \b, \r`. თითოეულ შემთხვევაში გაუშვით პროგრამა და გააანალიზეთ შედეგი.

ამოცანა 2: მოცემული მთელი `n` რიცხვისთვის დაბეჭდეთ მისი მეხუთედის მთელი ნაწილი და ზუსტი მნიშვნელობა.

შესაბამის C++-პროგრამას აქვს სახე:

```
//////////////////////////////////////////////
//    ავტორი:
//    პროგრამა: მთელი n რიცხვის მეხუთედის პოვნა
//    თარიღი: 06/20/2011
//////////////////////////////////////////////
#include <iostream>
using namespace std;
int main()
{
   int n =4;
   cout<< n/5 <<endl;
   cout<< n/5. <<endl;
   system("PAUSE");
   return 0;
}
```

დავალება:

ა) შეასრულეთ პროგრამა და ახსენით მიღებული შედეგი.

ბ) პროგრამა გაუშვით `n`–ის შემდეგი მნიშვნელობებისთვის: `9, 17, 33` და გააანალიზეთ შედეგები.

ამოცანა 3. მოცემული ნამდვილი `d` რიცხვისთვის დაბეჭდეთ მისი მეცხრედი.

დამოუკიდებელი სამუშაო:

1. დაწერეთ პროგრამა, რომელიც ეკრანზე გამოიტანს შემდეგ ტექსტს:

   გაიღიმე! გაიღიმე! გაიღიმე!
   გაიღიმე! გაიღიმე!
   გაიღიმე!

2. დაწერეთ პროგრამა, რომელიც მოცემული მთელი `a` რიცხვისთვის დაბეჭდავს მისი მესამედის მთელ ნაწილს და ზუსტს მნიშვნელობა.ს. პროგრამა შეასრულეთ `a`–ს შემდეგი მნიშვნელობებისთვის: `2, 10, 25` და ახსენით მიღებული შედეგები.

პრაქტიკული მეცადინეობა № 2

პრაქტიკული მეცადინეობის თემები:

- სტანდარტული ტიპის ცვლადების ბეჭდვა
- არითმეტიკული და შედარების ოპერატორები

მასალა დამოუკიდებლად გაცნობისათვის:

1. სტანდარტული ტიპის ცვლადების ბეჭდვის ნიმუშები:

№	პროგრამის ფრაგმენტი	შედეგი
1	`cout <<1111<<5555<<"AAAA";`	11115555AAAA
2	`cout<<1111<<endl<<5555<<endl<<"AAAA"<<endl;`	1111 5555 AAAA
3	`cout<<11.11<<endl<<false<<endl<<true<<endl<<'a'<<endl;`	11.11 0 1 a
4	`double d = 12.00357;` `cout<< d <<endl<< 'd'<<endl;`	12.0036 d
5	`int a = 1111;` `cout<< 'a' <<endl<<a<<endl;`	a 1111

შენიშვნა: აქ და ქვემოთ, იმისათვის რომ ვნახოთ პროგრამის ფრაგმენტის შედეგი, იგი უნდა შევავსოთ სრულ პროგრამულ კოდამდე. მაგალითად, მეხუთე ფრაგმენტისთვის უნდა დავწეროთ:

```
#include <iostream>
using namespace std;
int main()
{
    int a = 1111;
    cout<< 'a' <<endl<<a<<endl;

    //system("pause");
    return 0;
}
```

ხოლო სხვა ნიმუშებში შევცვალოთ ერთი ფრაგმენტი მეორეთი.

2. ოპერატორების გამოყენების ნიმუშები:

№	პროგრამის ფრაგმენტი	შედეგი

1	`int b = 2;`
	`b+=3;`
	`cout<<"b gaxda "<<b<<endl;`
		b gaxda 5

2	`double k =5.0148;`
	`k+=6;`
	`cout<<"k udris "<<k<<endl;`
		k udris 11.0148

3	`int b =2; double p =2.5;`
	`b*=2;  p*=2;`
	`cout<<"b udris "<<b<<endl`
	`<<"p udris "<<p<<endl;`
		b udris 4
		p udris 5

4	`int n =20; double q =-25;`
	`n/=6;  q/=6;`
	`cout<<"n udris "<<n<<endl`
	`<<"q udris "<<q<<endl;`
		n udris 3
		q udris -4.16667

5	`int c =15;`	c udris 6

	c%=9; cout<<"c udris "<<c<<endl;	
6	string name = "George"; cout<< name <<endl; cout<< name + ' ' + "Lukas"<<endl;	George George Lukas
7	string name = "George"; name+=" Lukas"; cout<< name<<endl;	George Lukas
8	char c = 'F'; c+=1; cout<< c<<endl;	G

9	int m = 11;
	cout<<m++<<endl;
	cout<<m<<endl;
	cout<<++m<<endl<<m<<endl;
		11
		12
		13
		13

10	char c = 'A';
	cout<<c++;
	cout<<c;
	cout<<++c<<c<<endl;
		ABCC

3. როგორ მუშაობს შედარების ოპერატორები

№	პროგრამის ფრაგმენტი	შედეგი
1	cout<< (37 == 42) <<endl;	0 // მცდარი
2	cout<< (25 > 17) <<endl;	1 // ჭეშმარიტი
3	int x =7, y =5; cout<< (x >= y) <<endl;	1 // ჭეშმარიტი
4	int a =55, b =15; cout<< (a <= b) <<endl;	0 // მცდარი
5	string name = "George"; cout<<(name == "George")<<endl;	1 // ჭეშმარიტი
6	string name = "George"; cout<<(name != "George")<<endl;	0 // მცდარი
7	double p =1.051, q =1.05; cout<< (p > q) <<endl;	1 // ჭეშმარიტი
8	double x =-0.000045, y =-0.0045; cout<< (x <= y) <<endl;	0 // მცდარი

საუდიტორიო სამუშაო:

1. იპოვეთ m ცვლადის მნიშვნელობა, თუ მოცემული გვაქვს განაცხადი int m =4;

   ა) m +=5; ბ) m *=10; გ) m -=6; დ) m /=8; ე) m %=5;

2. იპოვეთ c ცვლადის მნიშვნელობა, თუ მოცემული გვაქვს განაცხადი char c ='K';

   ა) c +=5; ბ) c -=5; გ) c ='M'; დ) --c;

3. იპოვეთ s ცვლადის მნიშვნელობა, თუ გვაქვს განაცხადი string s ="String";

   ა) s +=" AAAA"; ბ) s = s+' '+"AAAA"+'.'; გ) s ="UUUH!";

4. იპოვეთ x ცვლადის მნიშვნელობა, თუ გვაქვს განაცხადი double x = 11.907;

   ა) x++; ბ) x =(x-1.907)/3; გ) x -=11.807; დ) x +=-5;

5. ვთქვათ, x და y int ტიპის ცვლადებია. იპოვეთ მათი მნიშვნელობები:

   ა) x =(7-3)*9; ბ) y =(4+7)/3*2; გ) y =x =(10+8)/6; დ) y =4+5*(x =7/3);

ე) `x =(8+2)*2.5;` ვ) `y =(2+3)*2.5;` ზ) `y = (int)5.5+4.7;`

6. ა) `s` ცვლადს მიანიჭეთ მთელი `a, b` და `c` რიცხვების საშუალო არითმეტიკული. ზუსტი შედეგის მისაღებად როგორი ტიპის უნდა იყოს `s`?

ბ) `k` ცვლადს მიანიჭეთ მთელი `n` და `m` რიცხვების სხვაობის მეცხრედი. ზუსტი შედეგის მისაღებად როგორი ტიპის უნდა იყოს `k`?

გ) `C` ცვლადს მიანიჭეთ მთელი `A` და `B` რიცხვების ჯამის 4-ზე გაყოფისას მიღებული რიცხვის მთელი ნაწილი. დაადგინეთ `C` ცვლადის ტიპი.

7. როგორ გავარკვიოთ ორი ცვლადიდან რომლის მნიშვნელობაა მეტი? ახსენით რას დაბეჭდავს შემდეგი პროგრამა.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
   int n =3, m =2;
   cout<<(n == m)<<'\t'
       <<(n > m)<<endl;
   char p ='B', c ='Q';
   cout<<(p != c)<< '\t'
       <<(p <= c)<<endl;
   double x =10.5, y =-0.25;
   cout<<(x < y)<<'\t'
       <<(x >= y)<<endl;
   string S ="Anano", Q ="Aniko";
   cout<<(S > Q)<<'\t'
       <<(S < Q)<<endl;
   //system("PAUSE");
   return 0;
}
```

პასუხი:

```
0       1
1       1
0       1
0       1
Press any key to continue . . .
```

8. რა მნიშვნელობებს მიიღებს `x` და `y` ცვლადები?

```
int x =5, y =3;
x =y; y =x;
```

9. ახსენით სიტყვებით, რას აკეთებს კოდის შემდეგი ორი ფრაგმენტი:

ა)
```
int x =9, y =5, z;
z = x;
x = y;
y = z;
cout<<x<<' '<<y<<endl;
```

ბ)
```
int x =9, y =5;
x = y + x;
y = x - y;
x = x - y;
cout<<x<<' '<<y<<endl;
```

10. მოცემულია სამნიშნა რიცხვი `n`.

ა) დაბეჭდეთ `n` რიცხვის ციფრთა ჯამი.

```
#include <iostream>
using namespace std;
int main(){
   int n, sum;
   cout<<"Enter integer\n";
   cin>>n;
   sum =n/100+(n/10)%10+n%10;
   cout<<"sum = "<<sum<<endl;
   //system("PAUSE");
   return 0;
}
```

პროგრამის შედეგია:

```
Enter integer
537
sum = 15
Press any key to continue . . .
```

ბ) მთელ k ცვლადს მიანიჭეთ სამნიშნა m რიცხვის შებრუნებული რიგით აღებული ციფრებისგან შედგენილი რიცხვი და დაბეჭდეთ.

```
#include <iostream>
using namespace std;
int main(){
   int m, k;
   cout<<"Enter m"<<endl;
   cin>>m;
   k =100*(m%10)+10*((m/10)%10)+m/100;
   cout<<"k = "<<k<<endl;
   //system("PAUSE");
   return 0;
}
```

პროგრამის შედეგია:

```
Enter m
357
k = 753
Press any key to continue . . .
```

დამოუკიდებელი სამუშაო:

1. ა) A ცვლადს მიანიჭეთ მთელი B და D რიცხვების ჯამის მესამედი. ზუსტი შედეგის მისაღებად როგორი ტიპის უნდა იყოს A?
   ბ) განსაზღვრეთ d ცვლადის ტიპი და d –ს მიანიჭეთ p რიცხვის მეოთხედისა და q რიცხვის მეექვსედის სხვაობა, თუ :
   - p და q მთელი რიცხვებია;
   - p და q ნამდვილი რიცხვებია.
2. იპოვეთ y ცვლადის მნიშვნელობა, თუ მოცემული გვაქვს განაცხადი `int x = 7, y = 9;`

   ა) `y =(x == y);` ბ) `y =(x < y);` გ) `y =(x > y);` დ) `y = x <= y;`
   ე) `x = x != y;` ვ) `x += y;` ზ) `y *= x-1;` თ) `y /= x+1;` ი) `y += 2*x-y.`

3. შეამოწმეთ კომპიუტერზე და გააანალიზეთ მიღებული პასუხები (შეავსეთ ცხრილი):

№	პროგრამის ფრაგმენტი	შედეგი

1	`int x =1, z;`
	`z =x++;`
	`cout<<"x = "<<x`	

	<<"\tz = "<<z<<endl;	

2	int x =1, z;
	z =++x;
	cout<<"x = "<<x
	<<" z = "<<z<<endl;	

3	char s ='B', t; t =s--; cout<<"s = "<<s <<"\tt = "<<t<<endl;	

4	char s ='B', t;
	t =--s;
	cout<<"s = "<<s
	<<"\nt = "<<t<<endl;	

5	double d =15.674, q;
	q =++d -10;
	cout<<"d = "<<d
	<<" q = "<<q<<endl;	

6	double d =15.674, q;
	q =d-- +10;
	cout<<"d = "<<d
	<<" q = "<<q<<endl;	

7	string S = "Hello", K ="Students"; S +=", " + K + '!'; cout<<"S = "<<S<<endl;	

8	bool b =false;
	cout<<++b<<endl;
	b -=1;
	cout<<b<<endl;	

4*. რა შედეგი მიიღება პროგრამის შემდეგი ფრაგმენტის შესრულებით?

```
cout << (0.3 == 3*0.1)<< endl;
```

მითითება: მიღებული პასუხის ახსნისთვის ჩაამატეთ პროგრამაში

```
#include <iomanip>
```

და დაბეჭდეთ 3*0.1 –ის მნიშვნელობა 20 ათწილადი ნიშნის სიზუსტით:

```
cout << fixed << setprecision(20);
cout << 3*0.1 << endl;
```

5. ახსენით, რას დაბეჭდავს პროგრამის შემდეგი ფრაგმენტი და რატომ:

ა)
```
string A ="stumari", B ="studenti";
cout << (A < B) << endl << (A != B) << endl;
```

ბ)
```
string C ="kargi ";
cout << (C + "amindi" >= C + "ambavi") << endl;
```

გ)
```
string K ="Word", M ="Excel";
cout << (K > M) << '\t'<<(K =M) << endl;
cout << (K == M) << endl;
```

6. ახსენით, რას აკეთებს კოდის შემდეგი სამი ფრაგმენტი:

ა)
```
char p ='!', c ='?', d;
d = p; p = c; c = d;
cout << p << ' ' << c << endl;
```

ბ)
```
bool t = true, f = false, d;
d = f; f = t; t = d;
cout << boolalpha << "t gaxda " << t
```

```
        << endl << "f gaxda " << f << endl;
```

გ)
```
string S ="Gel", Q ="Euro", P;
P = S; S = Q; Q = P;
cout<<S<< '\t'<<Q<<endl;
```

7. შეამოწმეთ, რას დაბეჭდავს შემდეგი პროგრამა და ახსენით:

```
#include <iostream>
using namespace std;
int main()
{
   char x ='*';
   cout<<x<<' '<<(int)x<<endl;
   int p =127;
   cout<<(char)p<<" simbolos kodi = "
       << p << endl;
   //system("PAUSE");
   return 0;
}
```

8. d ცვლადს რიგრიგობით მიანიჭეთ a რიცხვის ყველა ციფრი, თუ:
   ა) a არის ოთხნიშნა მთელი რიცხვი;
   ბ) a არის ხუთნიშნა მთელი რიცხვი.

9. მოცემულია სამნიშნა N რიცხვი. დაბეჭდეთ N რიცხვის ციფრები შებრუნებული რიგით.

ლაბორატორიული მეცადინეობა № 2

ლაბორატორიული სამუშაოს აღწერა:

- სტრიქონების წაკითხვა და ბეჭდვა
- მთელი რიცხვების წაკითხვა
- სამი რიცხვის საშუალო არითმეტიკულის გამოთვლა

ლაბორატორიული სამუშაო:

ამოცანა 1. სტრიქონის შეტანა კლავიატურიდან და გამოტანა ეკრანზე.

```
//////////////////////////////////////
//  ავტორი:
//  პროგრამა: სახელის წაკითხვა და ბეჭდვა
//  თარიღი:
//////////////////////////////////////
#include <iostream>
#include <string>
using namespace std;
int main()
{
   cout<<"Please enter your first name"
         " (followed by 'Enter'):\n";
   string first_name;
   cin >> first_name;
   cout << "Hello, " << first_name << "!\n";
   //system("PAUSE");
   return 0;
}
```

დავალება: გაუშვით პროგრამა შესრულებაზე: შეიტანეთ კლავიატურიდან თქვენი სახელი, ბოლოს დააჭირეთ Enter –ს (რითაც დააფიქსირებთ შეტანილ ინფორმაციას). გააანალიზეთ მიღებული შედეგი.

ამოცანა 2. სტრიქონის და მთელი რიცხვის შეტანა კლავიატურიდან და გამოტანა ეკრანზე.

```
////////////////////////////////////////////////
//  ავტორი:
//  პროგრამა: სახელის და ასაკის წაკითხვა და ბეჭდვა
//  თარიღი:
////////////////////////////////////////////////
#include <iostream>
#include <string>
using namespace std;
int main(){
   cout<<"Please enter your first name"
         " and age:\n";
   string first_name;
   int age;
   cin >> first_name >> age;
   cout << "Hello, " << first_name << "!\n"
       << "You are " << age << " years old" << endl;
   //system("PAUSE");
   return 0;
}
```

დავალება: გაუშვით პროგრამა შესრულებაზე: შეიტანეთ კლავიატურიდან თქვენი სახელი და ასაკი, შეტანა დაასრულეთ Enter –ზე დაჭერით. გააანალიზეთ მიღებული შედეგი.

ამოცანა 3. პროგრამა "გეკითხებათ" თქვენ სახელსა და გვარს და შემდეგ ბეჭდავს მისალმების სტრიქონს.

```
//////////////////////////////////////////////
//  ავტორი:
//  პროგრამა: სახელისა და გვარის წაკითხვა და ბეჭდვა
//  თარიღი:
//////////////////////////////////////////////
#include <iostream>
#include <string>
using namespace std;
int main(){
   cout<<"Please enter your first name:\n";
   string first_name;
   cin >> first_name;
   cout << "Please enter your last name:\n";
   string last_name;
   cin >> last_name;
   cout << "Hello, " << first_name <<' '
       << last_name<<endl;
   //system("PAUSE");
   return 0;
}
```

დავალება:

ა) გაუშვით პროგრამა შესრულებაზე: შეიტანეთ კლავიატურიდან თქვენი სახელი და შემდეგ გვარი. გააანალიზეთ მიღებული შედეგი.

ბ) გადაწერეთ პროგრამა: პირველი 6 სტრიქონი შეცვალეთ შემდეგი 3 -ით

```
cout<<"Please enter your first and last names\n";
string first_name, last_name;
cin>>first_name>>last_name;
```

კვლავ გაუშვით პროგრამა შესრულებაზე და შეიტანეთ კლავიატურიდან თქვენი სახელი და გვარი. გააანალიზეთ მიღებული შედეგი.

ამოცანა 4. დაბეჭდეთ სხვადასხვა ტიპებისთვის მათი წარმოდგენის უდიდესი და უმცირესი მნიშვნელობები

```
//////////////////////////////////////////
//  ავტორი:
//  პროგრამა: უდიდესი და უმცირესი სიდიდეები
//  თარიღი:
//////////////////////////////////////////
```

ა) მთელი ტიპებისა:

```
#include <iostream>
#include <climits>
using namespace std;
int main()
{
   cout << INT_MIN << '\t' << INT_MAX << endl;
   cout << 0 << '\t' << UINT_MAX << endl;
   cout << CHAR_MIN <<'\t'<< CHAR_MAX << endl;
   cout << 0<<'\t' << UCHAR_MAX << endl;
   cout << SHRT_MIN <<'\t' << SHRT_MAX << endl;
   cout << 0<<'\t' <<USHRT_MAX<<endl;
   cout << LONG_MIN <<'\t' << LONG_MAX << endl;
   cout << 0 <<'\t' <<ULONG_MAX << endl;
```

```
    cout << sizeof(long long) << endl;
    cout << LLONG_MIN << '\t' << LLONG_MAX << endl;
    cout << 0 << '\t' << ULLONG_MAX << endl;
    //system("PAUSE");
    return 0;
}
```

ბ) ნამდვილი ტიპებისა:

```
#include <iostream>
#include <cfloat>
using namespace std;
int main()
{
    cout << FLT_MIN <<' \t' << FLT_MAX << endl;
    cout << DBL_MIN<<'\t' << DBL_MAX << endl;
    //system("PAUSE");
    return 0;
}
```

დავალება: გაუშვით პროგრამა შესრულებაზე: შეიტანეთ კლავიატურიდან სამი მთელი რიცხვი და გააანალიზეთ მიღებული შედეგი.

დამოუკიდებელი სამუშაო:

1. მოცემული სამი მთელი რიცხვისთვის, x ცვლადს მიანიჭეთ მათი საშუალო არითმეტიკული და დაბეჭდეთ იგი.

შესაბამისი C++ –პროგრამის სახეა:

```
////////////////////////////////////////////////
//  ავტორი:
//  პროგრამა: სამი რიცხვის საშუალო არითმეტიკული
//  თარიღი:
////////////////////////////////////////////////
#include <iostream>
using namespace std;
int main(){
    int a, b, c;
    cout<<"Enter 3 integers :\n";
    cin>>a>>b>>c;
    double x;
    x =(a+b+c)/3.;        // საშუალო
    cout << "The average is " << x << endl;
    //system("pause");
    return 0;
}
```

დავალება: გაუშვით პროგრამა შესრულებაზე: შეიტანეთ კლავიატურიდან სამი მთელი რიცხვი და გააანალიზეთ მიღებული შედეგი.

2. რა მოხდება, თუ მთელი რიცხვი უფრო მეტი აღმოჩნდება, ვიდრე ამას ამ რიცხვისათვის არჩეული ტიპი ითვალისწინებს?

გაარჩიეთ პროგრამა:

```
#include <iostream>
#include <climits>
using namespace std;
int main(){
```

```
    int x =INT_MAX, y =INT_MIN;
    cout<<"x = "<<x<<'\t'
        <<"x + 1 = " << x + 1 << '\t'
        <<"x + 2 = " << x + 2 << endl;
    cout<<"y = " << y <<'\t'
        <<"y - 1 = " << y – 1 << '\t'
        <<"y - 2 = " << y – 2 << endl;
    //system("pause");
    return 0;
}
```

დავალება:

ა) შეასრულეთ მოცემული პროგრამა. გააანალიზეთ მიღებული შედეგები.

ბ) შეასწორეთ პროგრამა და შეასრულეთ იგივე ოპერაციები უნიშნო მთელი ტიპის რიცხვებისთვის (ტიპი **unsigned int**). გაითვალისწინეთ, რომ ამ შემთხვევაში x =UINT_MAX, y =0.

პრაქტიკული მეცადინეობა № 3

პრაქტიკული მეცადინეობის თემები:

- **if** შეტყობინება, **if-else** შეტყობინება
- ოპერატორი **? :**

მასალა დამოუკიდებლად გაცნობისათვის:

1. **if, if-else** შეტყობინებებით განხორციელებული განშტოების რამდენიმე ნიმუში:

№	პროგრამის ფრაგმენტი	შედეგი

1	`cout<<"1111\n";`
	`if(5 > 3)`
	`  cout<<"2222\n";`
	`cout<<"3333\n";`
		1111
		2222
		3333

2	`cout<<"1111\n";`
	`if(15 <= 3)`
	`  cout<<"2222\n";`
	`cout<<"3333\n";`
		1111
		3333

3	`cout<<"1111\n";`
	`if(15 <= 3)`
	`  cout<<"2222\n";`
	`else`
	`  cout<<"3333\n";`
	`cout<<"4444\n";`
		1111
		3333
		4444

4	`int a =11;`
	`cout<<"1111\n";`
	`if(a >= 5)`
	`  cout<<"2222\n";`
	`else`
	`  cout<<"3333\n";`
	`cout<<"4444\n";`
		1111
		2222
		4444

6	`if( true )  // იგივეა რაც if(1)`
	`  cout<<"yes";`
	`else`
	`  cout<<"no";`
		yes

7	`if(0)  // იგივეა რაც if( false )`
	`  cout<<"yes";`
	`else`
	`  cout<<"no";`
		no

9	`int k = 7;`
	`if(k > 0)`
	`  cout<<k<<" > 0"<<endl;`
	`else`
	`  cout<<k<<" <= 0"<<endl;`
		7 > 0

10	`int a =10,b =111;`
	`if(a > b)`
	`  cout<<a<<" > "<<b<<endl;`
	`else`
	`  cout<<a<<" <= "<<b<<endl;`
		10 <= 111

11	`int x =8;`
	`if(x % 2 == 0)`
	`  cout<<x<<" is even\n";`
	`else`
	`  cout<<x<<" is odd\n";`
		8 is even

12	`int x =9;`
	`if(x % 2 == 0)`
	`  cout<<x<<" is even\n";`
	`else`
	`  cout<<x<<" is odd\n";`
		`9 is odd`

13	`int x =12;`
	`if(x % 3 != 0)`
	`  cout<<x<<" ar aris 3-is jeradi\n";`
	`else`
	`  cout<<x<<" 3-is jeradia\n";`
		`12 3-is jeradia`

14	`int x =14;`
	`if(x % 3 != 0)`
	`  cout<<x<<" ar aris 3-is jeradi\n";`
	`else`
	`  cout<<x<<" 3-is jeradia\n";`
		`14 ar aris 3-is jeradi`

2. ჩადგმული **if-else** შეტყობინების გამოყენების რამდენიმე ნიმუში (გავითვალისწინოთ, რომ **else** შეტყობინება ეკუთვნის მის წინ მდგარ პირველივე **if** შეტყობინებას):

№	პროგრამის ფრაგმენტი	შედეგი

1	`if(5 >= 0)`
	`  if(3 > 2) cout<<"yes";`
	`  else cout<<"no";`
		`yes`

2	`if(5 >= 0)`
	`  if(2 > 3) cout<<"yes";`
	`  else cout<<"no";`
		`no`

3	`if(5 < 0)`
	`  if(2 > 3) cout<<"yes";`
	`  else cout<<"no";`	

4	`if(5 < 0)`
	`  if(2 > 3) cout<<"yes";`
	`  else cout<<"no";`
	`else cout<<"this";`
		`this`

3. **if-else** შეტყობინებაში ლოგიკური ოპერატორების გამოყენების რამდენიმე ნიმუში:

№	პროგრამის ფრაგმენტი	შედეგი

1	`if( 5 > 3  && 123 <= 3736 )`
	`// ეს სტრიქონი კომპილერისთვის იგივეა რაც if( 1 && 1 )`
	`  cout<<"1111\n";`
	`else`
	`  cout<<"2222\n";`
		`1111`

2	`if( 5 > 333  && 123 <= 3736 )`
	`// ეს სტრიქონი კომპილერისთვის იგივეა რაც if( 0 && 1 )`
	`  cout<<"1111\n";`
	`else`
	`  cout<<"2222\n";`
		`2222`

3	`if( 5 > 333  || 123 <= 3736 )`
	`// ეს სტრიქონი კომპილერისთვის იგივეა რაც if( 0 || 1 )`
	`  cout<<"1111\n";`
	`else`
	`  cout<<"2222\n";`
		`1111`

4	`if( 5 > 333  || 123 > 3736 )`
	`// ეს სტრიქონი კომპილერისთვის იგივეა რაც if( 0 || 0 )`
	`   cout<<"1111\n";`
	`else`
	`   cout<<"2222\n";`
		2222

სააუდიტორიო სამუშაო:

1. დაადგინეთ და დაბეჭდეთ

ა) ორ მთელ რიცხვს შორის უდიდესი

```
#include <iostream>
using namespace std;
int main(){
  int a =21, b =15, max;
  if(a > b) max =a;
  else max =b;
  cout<<"udidesi = "<<max<<endl;
  system("pause");
  return 0;
}
```

პასუხი :

```
udidesi = 21
Press any key to continue . . .
```

ბ) ორ სიმბოლოს შორის უდიდესის ამორჩევა **?** **:** ოპერატორის გამოყენებით

```
#include <iostream>
using namespace std;
int main(){
   char p ='A', c ='M', max;
   max = p > c ? p : c;
   cout<<"udidesi aris "<<max<<endl;
   system("pause");
   return 0;
}
```

პასუხი:

```
udidesi aris M
Press any key to continue . . .
```

2. იპოვეთ და დაბეჭდეთ ორ სტრიქონს შორის უმცირესი

```
#include <iostream>
#include <string>
using namespace std;
int main(){
   string S ="netbook", P ="ipod", min;
   min =S;
   if(P < min) min =P;
   cout<<"umciresi striqonia \""
       <<min<<"\""<<endl;
   system("pause");
   return 0;
}
```

პასუხი:

```
umciresi striqonia "ipod"
Press any key to continue . . .
```

3. იპოვეთ და დაბეჭდეთ სამ ნამდვილ ცვლადს შორის უმცირესი

```
#include <iostream>
using namespace std;
int main(){
   double x =1.5,y =2.0,z =-1.2, min;
   min =x;
```

```
    if(y < min) min =y;
    if(z < min) min =z;
    cout<<"min = "<<min<<endl;
    system("pause");
    return 0;
}
```

პასუხი:

```
min = -1.2
Press any key to continue . . .
```

4. იპოვეთ და დაბეჭდეთ y ცვლადის მნიშვნელობა:

ა) $y=\begin{cases} 1-x^2, & \text{თუ } x>-2 \\ 5x+7, & \text{თუ } x\le -2 \end{cases}$

```
#include <iostream>
using namespace std;
int main(){
    int x =2, y;
    if(x >- 2) y =1-x*x;
    else  y =5*x+7;
    cout<<"y = "<<y<<endl;
    system("pause");
    return 0;
}
```

პასუხი:

```
y = -3
Press any key to continue . . .
```

ბ) $y=\begin{cases} x, & \text{თუ } -5\le x\le 5 \\ 2*x-2, & \text{წინააღმდეგ შემთხვევაში} \end{cases}$

```
#include <iostream>
using namespace std;
int main(){
    int x =10, y;
    if(x >= -5 && x <= 5) y =x;
    else  y =2*x-2;
    cout<<"y = "<<y<<endl;
    system("pause");
    return 0;
}
```

პასუხი:

```
y = 18
Press any key to continue . . .
```

გ) $y=\begin{cases} \dfrac{x}{x-5}, & \text{თუ } x\ge 0\ (x\ne 5) \\ 5, & \text{თუ } x=5 \\ 4x^2, & \text{თუ } x<0 \end{cases}$

```
#include <iostream>
using namespace std;
int main(){
    double x =7, y;
    if(x >= 0)
       if(x == 5)y =5;
       else y =x/(x-5);
    else y =4*x*x;
    cout<<"y = "<<y<<endl;
    system("pause");
```

```
    return 0;
}
```

პასუხი:

```
y = 3.5
Press any key to continue . . .
```

დამოუკიდებელი სამუშაო:

1. მიუთითეთ შეცდომები, რომლებიც დაშვებულია პროგრამის შემდეგ ფრაგმენტებში:

ა)
```
int x =1,y =2;
if(x < 2) x =x+1; y =0;
else x =0; y =y+1;
cout<<"x = "<<x<<" y = "<<y<<endl;
```

ბ)
```
int x =-1,y =3;
if x>0 x =x+1;
else {x =0; y =y+1;}
cout<<"x = "<<x<<" y = "<<y<<endl;
```

გ)
```
#include <iostream>
using namespace std;
int main(){
  int x =1,y =2;
   if(x == 1)
      if(y == 1)
         cout<<"x = "<<x<<" y = "<<y<<endl;
   else cout<<"x != 1\n";
  system("pause");
   return 0;
}
```

პროგრამის შესრულების შედეგია :

```
x != 1
Press any key to continue . . .
```

რაც არასწორია. შეცადეთ პროგრამა შეასწოროთ.

2. რა იქნება პროგრამის შემდეგი ფრაგმენტის შესრულების შედეგი? (შეავსეთ ცხრილი)

№	პროგრამის ფრაგმენტი	შედეგი

1	`int x, y =-3;`
	`if(y != 0) x =10;`
	`cout<<"x = "<<x<<endl;`	

2	`int x, y =-3;`
	`if(y == 0) x =10;`
	`else x =5;`
	`cout<<"x = "<<x<<endl;`	

3	`int  x =1,y =1,z;`
	`if(x > 0 && y <= 1)`
	`  z =x+y;`
	`else`
	`  z =y-x;`
	`cout<<"z = "<<z<<endl;`	

4	`int  x =1,y =1,z;`
	`if(x < 0 && y <= 1)`
	`  z =x+y;`
	`else`
	`  z =y-x;`
	`cout<<"z = "<<z<<endl;`	

3. იპოვეთ და დაბეჭდეთ y ცვლადის მნიშვნელობა. პროგრამა შეასრულეთ `int x` –ის სხვადასხვა მნიშვნელობებისთვის:

ა) $y = \begin{cases} x+1, & \text{თუ } x \geq 5 \\ x-1, & \text{თუ } x < 5 \end{cases}$

ბ) $y = \begin{cases} x^2+5, & \text{თუ } x > 2 \\ -2-x, & \text{თუ } x \leq 2 \end{cases}$

გ) $y = \begin{cases} 1, \text{ თუ } x < -1 \text{ ან } x > 1 \\ 2x-1, \text{ წინააღმდეგ შემთხვევაში} \end{cases}$

დ) $y = \begin{cases} \dfrac{1}{x+5}, \text{ თუ } x \geq 0 \\ 5x^2, \text{ თუ } x < 0 \ (x \neq -5) \\ 0, \text{ თუ } x = -5 \end{cases}$

4. შეადარეთ მოცემული ორი ნამდვილი რიცხვი და დაბეჭდეთ მათ შორის უმცირესი. თუ რიცხვები ტოლია დაბეჭდეთ შესაბამისი გზავნილი.

5. მოცემული მთელი ტიპის ერთი რიცხვი გაყავით მეორეზე და შედეგი დაბეჭდეთ. თუ გამყოფი ნულის ტოლია დაბეჭდეთ შესაბამისი გზავნილი.

6. მოცემულია მთელი რიცხვი `k`. გავზარდოთ მისი მნიშვნელობა 5–ით, თუ იგი ლუწია, ხოლო თუ კენტია – გავზარდოთ 3–ჯერ და დავბეჭდოთ.

7. ქვემოთ მოცემული პროგრამა პოულობს `x` რიცხვის მოდულს:

```
#include <iostream>
using namespace std;
int main(){
 int x =-7, modulx;
   modulx = x >= 0 ? x : -x;
   cout<<x<<" -is moduli udris "<<modulx
       <<" -s"<<endl;
   system("pause");
   return 0;
}
```

პასუხი:

```
-7 -is moduli udris 7 -s
Press any key to continue . . .
```

ა) შეასრულეთ იგი `x`–ის სხვადასხვა მნიშვნელობებისათვის და დარწმუნდით მის სისწორეში;

ბ) დაწერეთ პროგრამა, რომელიც ამოხსნის იგივე ამოცანას **`if-else`** შეტყობინების გამოყენებით.

8. დაწერეთ პროგრამა, რომელიც სამ მთელ რიცხვს შორის აირჩევს უმცირესს და დაბეჭდავს მას.

9*. `k` ცვლადს მივანიჭოთ იმ მეოთხედის ნომერი, რომელშიც ძევს წერტილი `A(x,y)` `(xy!=0)`. დაწერეთ პროგრამა და განიხილეთ შემთხვევები: ა) `A(-1,3);` ბ) `A(2,-5);` გ) `A(-4,-7);` დ) `A(7,4).`

10*. ცნობილია, რომ მოცემული ოთხი `a, b, c, d` რიცხვიდან სამი ტოლია, ერთი კი განსხვავებული. `n` ცვლადს მივანიჭოთ განსხვავებული რიცხვის სიდიდე. დაწერეთ პროგრამა და ჩაატარეთ მისი ტესტირება სხვადასხვა ოთხეულებისთვის.

ლაბორატორიული მეცადინეობა № 3

ლაბორატორიული სამუშაოს აღწერა:

- **char** ტიპი
- **if-else** შეტყობინება
- ASCII კოდების შესახებ

ლაბორატორიული სამუშაო:

ამოცანა 1. გაუშვით შემდეგი პროგრამა შესრულებაზე:

```
//////////////////////////////////////////
//  ავტორი:
//  პროგრამა: ascii.cpp, რომელიც გვიჩვენებს
//            სიმბოლოების კოდებს
//////////////////////////////////////////
#include <iostream>
using namespace std;
int main()
{
   char x1 ='a', x2 ='b', x3 ='c';
   cout << "simbolo " << x1 << "  kodi " << (int)x1 << endl;
   cout << "simbolo " << x2 << "  kodi " << (int)x2 << endl;
   cout << "simbolo " << x3 << "  kodi " << (int)x3 << endl;
   system("pause");
   return 0;
}
```

დავალება:

1. ჩაატარეთ პროგრამის შედეგის ანალიზი.
2. განაცხადი ცვლადებზე შეცვალეთ შემდეგით:
   ა) **char** x1='0', x2='1', x3='2';
   ბ) **char** x1='K', x2='L', x3='M';

   ორივე შემთხვევაში შეასრულეთ პროგრამა და დარწმუნდით თქვენს დასკვნებში.

ამოცანა 2. გაუშვით შემდეგი პროგრამა შესრულებაზე:

```
//////////////////////////////////////////
//  ავტორი:
//  პროგრამა: სიმბოლური და მთელი ტიპის
//            ცვლადების შეკრება
//////////////////////////////////////////
#include <iostream>
using namespace std;
int main()
{
    char x ='a'; int y =2, k;
    k = x+y;
    cout<<"k = "<<k<<endl
        <<"k = "<<char(k)<<endl;
    system("pause");
    return 0;
}
```

დავალება:

ა). გააანალიზეთ პროგრამის შედეგი.

ბ). შეცვალეთ x –სა და y –ის მნიშვნელობები, კვლავ შეასრულეთ პროგრამა და დარწმუნდით თქვენს დასკვნებში.

ამოცანა 3. ვიპოვოთ ორ სიმბოლოს შორის უდიდესი. ამოვბეჭდოთ მისი მნიშვნელობა და კოდი (**შენიშვნა:** ორ სიმბოლოს შორის უდიდესია ის, რომლის კოდიც მეტია).

```
//////////////////////////////////////////
//  ავტორი:
//  პროგრამა: სიმბოლოებს შორის უდიდესის
//              პოვნა
//////////////////////////////////////////
#include <iostream>
using namespace std;
int main()
{
   char x ='a', y ='k', m;
   // ორ სიმბოლოს შორის უდიდესის კოდი
   if( x > y) m =x;
   else m =y;
   cout<<(int)m<<endl;
   // ორ სიმბოლოს შორის უდიდესი
   cout<<m<<endl;
   system("pause");
   return 0;
}
```

დავალება:

შეცვალეთ პროგრამა ისე, რომ `x` –სა და `y` –ის მნიშვნელობების შეტანა ხდებოდეს კლავიატურიდან, კვლავ შეასრულეთ პროგრამა და დარწმუნდით მის სისწორეში.

დამოუკიდებელი სამუშაო:

1. დაადგინეთ `'d'` და `'z'` სიმბოლოთა კოდები.
2. დაწერეთ პროგრამა, რომელიც გაარკვევს კოდების დიაპაზონს:
   ა)`'A'`–დან `'Z'` სიმბოლომდე. (**მითითება:** საკმარისია დაბეჭდოთ `'A'` და `'Z'`-ის კოდები).
   ბ)`'0'`–დან `'9'` სიმბოლომდე.
3. დაადგინეთ 120-ის ტოლი კოდის მქონე სიმბოლო.
4. იპოვეთ ორ სიმბოლოს შორის უმცირესი. დაბეჭდეთ მისი მნიშვნელობა და კოდი.
5. იპოვეთ სამ სიმბოლოს შორის უდიდესი. დაბეჭდეთ მისი მნიშვნელობა და კოდი.

პრაქტიკული მეცადინეობა № 4

პრაქტიკული მეცადინეობის თემები:

- მათემატიკური ფუნქციების გამოყენება
- ოპერატორი **sizeof**
- ამორჩევის **switch** შეტყობინება. **break** შეტყობინება

მასალა დამოუკიდებლად გაცნობისათვის:

1. ზოგიერთი მათემატიკური ფუნქცია:
   1) `abs(x)` ფუნქციაა, რომელიც გამოითვლის **მთელი** x რიცხვის მოდულს - **მთელ** რიცხვს. `fabs(x)` ფუნქცია გამოითვლის **ნამდვილი** x რიცხვის მოდულს - **ნამდვილ** რიცხვს. ამ ფუნქციებით სარგებლობისათვის საჭიროა `#include<cmath>` დირექტივის ჩართვა.

№	პროგრამის ფრაგმენტი	შედეგი
1	`cout<<abs(-21)<<endl;`	21
2	`cout<<abs(345)<<endl;`	345
3	`cout<<abs(0)<<endl;`	0
4	`cout<<fabs(-32.25)<<endl;`	32.25
5	`cout<<fabs(98.00457)<<endl;`	98.0046
6	`cout<<fabs(-1)<<endl;`	1

   2) ფუნქცია `sqrt(x)` გამოითვლის **არაუარყოფითი ნამდვილი** x რიცხვიდან კვადრატულ ფესვს - **ნამდვილ** რიცხვს. ფუნქცია აღწერილია `cmath` ფაილში.

№	პროგრამის ფრაგმენტი	შედეგი
1	`cout<<sqrt(2.25)<<endl;`	1.5
2	`cout<<sqrt(-2.25)<<endl;`	-1.#IND
3	`cout<<sqrt(36.)<<endl;`	6
4	`cout<<sqrt(-16.0)<<endl;`	-1.#IND
5	`cout<<sqrt(0.)<<endl;`	0

   3) ფუნქცია `pow(x,y)` გამოითვლის x^y მნიშვნელობას - **ნამდვილ** რიცხვს, სადაც x და y - **ნამდვილი** რიცხვებია. საჭიროებს ფუნქცია აღწერილია `cmath` ფაილში.

№	პროგრამის ფრაგმენტი	შედეგი
1	`cout<<pow(1.44, 0.5)<<endl;`	1.2
2	`cout<<pow(2.0, -3)<<endl;`	0.125
3	`cout<<pow(3.0, 3)<<endl;`	27
4	`cout<<pow(-0.7, 2)<<endl;`	0.49
5	`cout<<pow(-5, 4.0)<<endl;`	625
6	`cout<<pow(-0.2, -3)<<endl;`	-125
7	`cout<<pow(-16,0.5)<<endl;`	-1.#IND

2. ოპერატორი **sizeof**:
   მოცემული კომპიუტერის მეხსიერებაში მოცემული ტიპის ობიექტის ზომის გარკვევა შეიძლება **sizeof** ოპერატორის საშუალებით. პროგრამაში შეგვიძლია ვისარგებლოთ **int** ტიპის მინიმალური `INT_MIN` და მაქსიმალური `INT_MAX` მუდმივების მნიშვნელობებით, **unsigned int** (უნიშნო მთელი) ტიპის მაქსიმალური `UINT_MAX` მუდმივის მნიშვნელობით, **char** ტიპის მინიმალური `CHAR_MIN` და მაქსიმალური `CHAR_MAX` მუდმივების მნიშვნელობებით. ამ მუდმივების მნიშვნელობებით სარგებლობისთვის ჩავრთოთ `#include<climits>` დირექტივა.

float ტიპის მუდმივების მნიშვნელობებია - მინიმალური FLT_MIN და მაქსიმალური FLT_MAX, **double** ტიპის მუდმივების მნიშვნელობებია - მინიმალური DBL_MIN და მაქსიმალური DBL_MAX. ამ მუდმივების მნიშვნელობებით სარგებლობისთვის ჩავრთოთ #include<cfloat> დირექტივა.

№	პროგრამის ფრაგმენტი	შედეგი

1	`// int ტიპის ზომა`
	`cout<<sizeof(int)<<" baiti\n";`
	`// int ტიპის რიცხვების დიაპაზონი`
	`cout<<INT_MIN<<" : "<<INT_MAX<<endl;`
		`4 baiti`
		`-2147483648 : 2147483647`

2	`// unsigned int ტიპის ზომა`
	`cout<<sizeof(unsigned int)<<" baiti\n";`
	`// unsigned int ტიპის რიცხვების დიაპაზონი`
	`cout<<0<<" : "<<UINT_MAX<<endl;`
		`4 baiti`
		`0 : 4294967295`

3	`// char ტიპის ზომა`
	`cout<<sizeof(char)<<" baiti\n";`
	`// char ტიპის ცვლილების დიაპაზონი`
	`cout<<CHAR_MIN<<" : "<<CHAR_MAX<<endl;`
		`1 baiti`
		`-128 : 127`

4	`// unsigned char ტიპის ზომა`
	`cout<<sizeof(unsigned char)<<" baiti\n";`
	`// char ტიპის ცვლილების დიაპაზონი`
	`cout<<0<<" : "<<UCHAR_MAX<<endl;`
		`1 baiti`
		`0 : 255`

5	`// float ტიპის ზომა`
	`cout<<sizeof(float)<<" baiti\n";`
	`// float ტიპის რიცხვების დიაპაზონი`
	`cout<<FLT_MIN<<" : "<<FLT_MAX<<endl;`
		`4 baiti`
		`1.17549e-038 : 3.40282e+038`

6	`// double ტიპის ზომა`
	`cout<<sizeof(double)<<" baiti\n";`
	`// double ტიპის რიცხვების დიაპაზონი`
	`cout<<DBL_MIN<<" : "<<DBL_MAX<<endl;`
		`8 baiti`
		`2.22507e-308 : 1.79769e+308`

3. მაგალითები **switch** შეტყობინების გამოყენებაზე

№	პროგრამის ფრაგმენტი	შედეგი

1	`int a =5;`
	`switch(a){`
	`  case 3: cout<<"a aris 3\n"; break;`
	`  case 5: cout<<"a aris 5\n"; break;`
	`  case 8: cout<<"a aris 8\n"; break;`
	`}`
		`/* დაიბეჭდება */`
		`a aris 5`

2	`int x =7;`
	`switch(x){`
	`  case 1: cout<<"x = 1\n"; break;`
	`  case 3: cout<<"x = 2\n"; break;`
	`  default: cout<<"x != 1, x != 3\n"; break;`
	`}`
		`/* დაიბეჭდება */`
		`x != 1, x != 3`

3	`int n =3;` `switch(n){`	`/* დაიბეჭდება */`

	`  case 1: cout<<"Anna\n"; break;`
	`  case 3: cout<<"Irakli\n"; break;`
	`  default: cout<<"Giorgi\n"; break;`
	`}`
		`Irakli`

4	`int n =3;`
	`switch(n){`
	`  case 1: cout<<"Anna\n"; break;`
	`  case 3: cout<<"Irakli\n";`
	`  default: cout<<"Giorgi\n"; break;`
	`}`
		`/* დაიბეჭდება */`
		
		`Irakli`
		`Giorgi`

5	`int n =1, m;`
	`switch(n){`
	`  case 1: m =n+1; cout<<"m = "<<m<<endl;`
	`  case 2: m =3*n; cout<<"m = "<<m; break;`
	`  default: m =n; cout<<"m = "<<m; break;`
	`}`
		`/* დაიბეჭდება */`
		
		`m = 2`
		`m = 3`

6	`int n =1, m;`
	`switch(n){`
	`  case 1: m =n+1; cout<<"m = "<<m<<endl;`
	`  case 2: m =3*n; cout<<"m = "<<m<<endl;`
	`  default: m =n; cout<<"m = "<<m<<endl;break;`
	`}`
		`/* დაიბეჭდება */`
		`m = 2`
		`m = 3`
		`m = 1`

10	`char c ='A';`
	`switch(c){`
	`  case 'A': cout<<"Aleko\n";  break;`
	`  case 'K': cout<<"Kote\n";  break;`
	`  case 'M': cout<<"Mari\n";  break;`
	`  default: cout<<"Gio\n";`
	`}`
		`/* დაიბეჭდება */`
		
		`Aleko`

12	`char c ='K';`
	`switch(c){`
	`  case 'A': cout<<"Aleko\n";  break;`
	`  case 'K': cout<<"Kote\n";`
	`  case 'M': cout<<"Mari\n";`
	`  default: cout<<"Gio\n";`
	`}`
		`/* დაიბეჭდება */`
		
		`Kote`
		`Mari`
		`Gio`

13	`char p;`
	`cin>>p;`
	`switch(p){`
	`  case 'A': case 'O':`
	`  case 'I': case 'E':`
	`  case 'U': cout<<"Xmovani asoa\n"; break;`
	`  default: cout<<"Ar aris xmovani aso\n";`
	`}`
		`/* შეტანა */`
		
		`B`
		`/* გამოტანა */`
		
		`Ar aris xmovani aso`

14	`int a, b =2, c;`
	` cin>>a;`
	` switch( b+a ){`
	`  case 6: c =3;  break;`
	`  case 12: c =2*a-1;  break;`
	`  default: c =a-b;`
	` }`
	` cout<<"c = "<<c<<'\n';`
		`//შეტანა  //გამოტანა`
		`4  c =3`
		
		`10  c =19`
		
		`120  c =118`

საუდიტორიო სამუშაო:

1. ჰერონის ფორმულის მეშვეობით გამოთვალეთ სამკუთხედის ფართობი, თუ მოცემულია სამკუთხედის a,b და c გვერდები. ჰერონის ფორმულის თანახმად,

$$S=\sqrt{p(p-a)(p-b)(p-c)}, \quad \text{სადაც} \quad p=(a+b+c)/2.$$

```
#include <iostream>
#include <cmath>
using namespace std;
int main(){
   double a, b, c, p, s;
   cout<<"SemoitaneT a, b, c gverdebis sidideebi :\n";
   cin>>a>>b>>c;
   //თუ ეს გვერდები ქმნიან სამკუთხედს, მაშინ ვისარგებლოთ ჰერონის ფორმულით
   if( a< b+c && b< a+c && c< a+b )
   {
      p =(a+b+c)/2;
      s =sqrt(p*(p-a)*(p-b)*(p-c));
      cout<<"partobi = "<<s<<endl;
   }
   else
   cout<<"aseTi gverdebis mqone samkuTxedi ar arsebobs\n";
   system("pause");
   return 0;
}
```

პროგრამის შესრულების შედეგია:

```
SemoitaneT a, b, c gverdebis sidideebi :
3 4 5
partobi = 6
Press any key to continue . . .
```

2. იპოვეთ და დაბეჭდეთ u ცვლადის მნიშვნელობა, თუ x და y – მთელი რიცხვებია და:

$$u=\begin{cases} x+y, & \text{თუ } |x-y|=3; \\ 2(x+y), & \text{თუ } |x-y|=6; \\ y-3x, & \text{თუ } |x-y|=7; \\ 10, & \text{წინააღმდეგ შემთხვევაში} \end{cases}$$

```
#include <iostream>
using namespace std;
int main(){
  int u, x, y;
  cout<<"x = ";
  cin>>x;
  cout<<"y = ";
  cin>>y;
  if(abs(x - y) == 3) u =x + y;
  else if(abs(x - y) == 6) u = 2*(x + y);
       else if(abs(x - y) == 7) u = y - 3*x;
            else u = 10;
  cout<<"u = "<<u<<endl;
  system("pause");
```

```
  return 0;
}
```

პროგრამის შესრულების შედეგია:

```
x = 3
y = 5
u = 10
Press any key to continue . . .
```

3. იგივე ამოცანა გავაკეთოთ **switch** შეტყობინების გამოყენებით:

```
#include <iostream>
using namespace std;
int main(){
  int u, x, y;
  cout<<"x = ";
  cin>>x;
  cout<<"y = ";
  cin>>y;
  switch( abs(x - y) ){
     case 3: u =x + y; break;
     case 6: u = 2*(x + y); break;
     case 7: u = y - 3*x; break;
     default: u = 10;
  }
  cout<<"u = "<<u<<endl;
  system("pause");
  return 0;
}
```

პროგრამის შესრულების შედეგია:

```
x = -7
y = -4
u = -11
Press any key to continue . . .
```

4. დაწერეთ პროგრამა, რომელიც კლავიატურიდან შეტანილი სიმბოლო-ციფრისთვის გაარკვევს და დაბეჭდავს წარმოადგენს იგი კენტ ციფრს, თუ ლუწს.

```
#include <iostream>
using namespace std;
int main(){
  char cifri;
  cout<<"shemoitaneT simbolo - cifri\n";
  cin>>cifri;
  switch(cifri){
     case '0': case '2':
     case '4': case '6':
     case '8':
          cout<<cifri<<" aris luwi\n"; break;
     case '1': case '3':
     case '5': case '7':
     case '9':
          cout<<cifri<<" aris kenti\n"; break;
     default:
          cout<<cifri<<" ar aris cifri.\n"
                      "Scade Tavidan!\n";
  }
```

```
    system("pause");
    return 0;
}
```

პროგრამის შესრულების შედეგია:

```
shemoitaneT simbolo - cifri
7
7 aris kenti
Press any key to continue . . .
```

5. კლავიატურიდან შეიტანეთ სტუდენტის შეფასება, რომელიც მან მიიღო გამოცდაზე (A, B, C, D, E). დაბეჭდეთ ქულების შესაბამისი დიაპაზონი.

```
#include <iostream>
using namespace std;
int main()
{
    char shefaseba;
    cin>>shefaseba;
    switch(shefaseba)
    {   case 'A': cout<<"91 - 100\n"; break;
        case 'B': cout<<"81 - 90\n"; break;
        case 'C': cout<<"71 - 80\n"; break;
        case 'D': cout<<"61 - 70\n"; break;
        case 'E': cout<<"51 - 60\n"; break;
        default: cout<<"ver chaabareT\n";
    }
   system("pause");
   return 0;
}
```

პროგრამა შესრულების შედეგებია:

```
A
91 - 100
Press any key to continue . . .
```

```
F
ver chaabareT
Press any key to continue . . .
```

დამოუკიდებელი სამუშაო:

1. შეადგინეთ პროგრამა, რომელიც გამოითვლის და დაბეჭდავს მანძილს 0X ღერძის ორ A და B წერტილს შორის.

2. შეადგინეთ პროგრამა, რომელიც გამოითვლის და დაბეჭდავს მანძილს სიბრტყის ორ A(-1.5,2.0) და B(1,-2.5) წერტილს შორის. $A(x_1,y_1)$ და $B(x_2,y_2)$ წერტილებს შორის მანძილი განისაზღვრება ფორმულით

$$d=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}\ .$$

3. შეადგინეთ პროგრამა, რომელიც გამოითვლის და დაბეჭდავს მანძილს სივრცის ორ – A(-1.5,2.0,1.2) და B(1,-2.5,0.5) წერტილს შორის.

$$d=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}\ .$$

4*. ჰერონის ფორმულის მეშვეობით გამოთვალეთ სამკუთხედის ფართობი, თუ მოცემულია სამკუთხედის წვეროების კოორდინატები.

5. გამოთვალეთ $2x^5+3x^3-7$-ის მნიშვნელობა, თუ x=2.1.

6. დაწერეთ პროგრამა, რომელიც შეამოწმებს, არის თუ არა მოცემული n წელი ნაკიანი. წელიწადს ეწოდება ნაკიანი, თუ მისი გამომსახველი რიცხვი 4-ის ჯერადია, მაგრამ 100-ის

ჯერადი წლებიდან ნაკიანია მხოლოდ 400-ის ჯერადები. მაგალითად, 1700, 1800 და 1900 – არაა ნაკიანი წლები, ხოლო 2000 – ნაკიანი წელია. n წელიწადის ნაკიანობას შევამოწმებთ შემდეგი პირობით:

```
n%4 == 0 && n%100 != 0 || n%400 == 0
```

7. რას დაბეჭდავს პროგრამის ფრაგმენტი,

```
int n, m;
cin>>n;
switch(n){
   case 1: m =n+1; cout<<"m = "<<m; break;
   case 2: m =3*n; cout<<"m = "<<m; break;
   default: m =n; cout<<"m = "<<m; break;
}
```

თუ შევიტანთ n -ის შემდეგ მნიშვნელობებს: ა) 1; ბ) 2; გ) 3.

8*. დაწერეთ პროგრამა, რომელიც დაითვლის და დაბეჭდავს კორექტულ შედეგს, თუ შევიტანთ ორ მთელ რიცხვს და შემდეგ არითმეტიკული მოქმედების ბინარულ ოპერატორს (`'+'`, `'-'`, `'*'` ან `'/'`).

ლაბორატორიული მეცადინეობა № 4

ლაბორატორიული სამუშაოს აღწერა:

- ჩადგმული **if - else**
- **switch** ოპერატორის გამოყენება

ლაბორატორიული სამუშაო:

ამოცანა 1. კვადრატული განტოლების ამოხსნა.

```
//////////////////////////////////////
//  ავტორი:
//  პროგრამა: კვადრატული განტოლების
//            ამოხსნა
//////////////////////////////////////
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
   double a,b,c,d,x1,x2;
   cout<<"SeitaneT gantolebis a, b da c koeficientebi\n";
   cin>>a>>b>>c;
   d =b*b-4*a*c;
   if(d < 0)
     cout<<"\ngantolebas namdvili amonaxsni ar gaaCnia\n";
   else if(d == 0)
        {
          x1 =-b/(2*a);
          cout<<"\ngantolebis amonaxsnebia:  "
                "x1 = x2 = "<<x1<<'\n';
        }
        else
        {
          x1 =(-b-sqrt(d))/(2*a);
          x2 =(-b+sqrt(d))/(2*a);
          cout<<"\ngantolebis amonaxsnebia:  "
               "x1 = "<<x1<<'\t'<<"x2 = "<<x2<<'\n';
        }
     system("pause");
     return 0;
}
```

დავალება:

ა) შეასრულეთ პროგრამა შემდეგი მონაცემებისთვის: 1) `a =1, b =4, c =4;` 2) `a =1, b =5, c =6;` 3) `a =1, b =2, c =9.`

ბ) შეიძლება თუ არა ამ ამოცანაში გამოვიყენოთ **switch** ოპერატორი?

გ) პროგრამა იმუშავებს სწორად, თუ შეიტანთ ნულისაგან განსხვავებულ `a` –ს. შეასრულეთ პროგრამა მონაცემებისთვის `a =0, b =2, c =3;` და ახსენით მიღებული შედეგი. როგორ გადავწეროთ პროგრამა, რომ მან კორექტულად იმუშავოს `a =0` შემთხვევაშიც?

ამოცანა 2. დაწერეთ პროგრამა, რომელიც კლავიატურიდან შეტანილ პატარა ასოსთვის გაარკვევს ხმოვანია იგი თუ თანხმოვანი და დაბეჭდავს შესაბამის დასკვნას.

```
//////////////////////////////////
//  ავტორი:
//  პროგრამა: ხმოვანი თუ თანხმოვანი?
//////////////////////////////////
```

```
#include <iostream>
using namespace std;
int main(){
  char aso;
  cout<<"shemoitaneT patara aso\n";
  cin>>aso;
  switch(aso){
     case 'a': case 'o':
     case 'i': case 'e':
     case 'u': cout<<aso<<" aris xmovani\n"; break;
     default: cout<<aso<<" aris Tanxmovani\n";
  }
  system("pause");
  return 0;
}
```

დავალება:

ა) შეასრულეთ პროგრამა სხვადასხვა პატარა ასოებისთვის და დარწმუნდით მის სისწორეში

ბ) რა მოხდება, თუ კლავიატურიდან პატარა ასოს ნაცვლად შევიტანთ სხვა სიმბოლოს? როგორ გავასწოროთ პროგრამის კოდი, რომ ამ შემთხვევაში პროგრამამ დაგვიბეჭდოს სათანადო გზავნილი? **მითითება:** პროგრამის ტექსტში **switch** ოპერატორამდე ჩაამატეთ შემდეგი ფრაგმენტი

```
if(aso < 'a' || aso > 'z')
   cout<<aso<<" ar aris patara aso.\n"
             "Try again!\n";
else
```

კვლავ შეასრულეთ პროგრამა ნებისმიერი სიმბოლოსთვის და დარწმუნდით, რომ იგი ყოველთვის იმუშავებს კორექტულად.

დამოუკიდებელი სამუშაო:

1. გაარჩიეთ ვალუტის გაცვლის პროგრამა, რომელსაც ლარებში შემოტანილი თანხის სიდიდე გადაჰყავს მოთხოვნის შესაბამისად დოლარში (D), ევროში (E) ან ინგლისურ ფუნტში (P).

```
#include <iostream>
using namespace std;
int main(){
  double amount, convert;
  char letter;
  cout<<"Enter amount in GEL : ";
  cin>>amount;
  cout<<"Enter the first letter of currency\n"
        "you wish to convert to (D, E or P): ";
  cin>>letter;
  switch(letter){
     case 'D': convert = amount*0.588408;
               cout<<amount<<" GEL == "<<convert<<" USD\n";
               break;
     case 'E': convert = amount*0.413449;
               cout<<amount<<" GEL == "<<convert<<" EUR\n";
               break;
     case 'P': convert = amount*0.365252;
               cout<<amount<<" GEL == "<<convert<<" GBP\n";
               break;
  }
  system("pause");
  return 0;
```

```
}
```

2. გაარჩიეთ პროგრამა, რომელსაც მოთხოვნის შესაბამისად გადაჰყავს სანტიმეტრები (c) დიუმებში (i) ან პირიქით.

```
#include <iostream>
using namespace std;
int main()
{
   // number of centimeters in an inch
   const double cm_per_inch = 2.54;
   int length; // length in inches or centimeters
   char unit;
   cout<<"Please enter a length followed by a letter (c or i):\n";
   cin>>length>>unit;
   switch (unit) {
     case 'i' :
         cout<<length<<" in == "<<cm_per_inch*length<<" cm\n";
         break;
     case 'c' :
         cout<<length<<" cm == "<<length/cm_per_inch<<" in\n";
         break;
     default:
         cout<<"Sorry, I don't know a unit called '"<<unit<<"'\n";
         break;
   }
   system("pause");
   return 0;
}
```

3. გაარჩიეთ პროგრამა, რომელიც "ხსნის" გადასასვლელზე ქცევის წესებს შუქნიშნის ჩართული ფერის მიხედვით (შემოიტანეთ `Yellow, Green` ან `Red`).

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
  string Traffic_light_color;
  cout<<"What color of a traffic light is included? ";
  cin>>Traffic_light_color;
  switch (Traffic_light_color[0]){
       case 'Y': cout<<"Be ready !\n"; break;
       case 'G': cout<<"Go !\n"; break;
       case 'R': cout<<"Stop !\n";
  }
  system("pause");
  return 0;
}
```

4. გაარჩიეთ პროგრამა, რომელიც თვის ნომრის მიხედვით ბეჭდავს თუ წელიწადის რომელ დროს ეკუთვნის მოცემული თვე.

```
#include <iostream>
using namespace std;
int main()
{
  int month_number;
  cout<<"Enter month's number ";
```

```
  cin>>month_number;
  switch(month_number){
    case 12:
    case 1:
    case 2: cout<<"Winter\n"; break;
    case 3:
    case 4:
    case 5: cout<<"Spring\n"; break;
    case 6:
    case 7:
    case 8: cout<<"Summer\n"; break;
    default: cout<<"Autumn\n";
  }
  system("PAUSE");
  return 0;
}
```

5. **switch** შეტყობინების გამოყენებით გამოთვალეთ მთელი d ცვლადის მნიშვნელობა, თუ

$$d=\begin{cases} a\%b+10, & a-c=3; \\ a+b-b/4, & a-c=6; \\ a-b+2c, & a-c=9. \end{cases}$$

a, b და c მთელი რიცხვები შეიტანეთ კლავიატურიდან. შეამოწმეთ თქვენი პროგრამის მუშაობა შემდეგი საწყისი მონაცემებისათვის:
1). a =2, b =3, c =5; 2). a =2, b =2, c =3; 3). a =2, b =1, c =4.

6. **switch** შეტყობინების გამოყენებით გამოთვალეთ u ცვლადის მნიშვნელობა, თუ

$$u=\begin{cases} 3x-y, & x+2y=1; \\ y-x, & x+2y=2; \\ x+y, & x+2y=3. \end{cases}$$

თუ კი $x+2y$ არ უდრის არც 1-ს, არც 2-ს და არც 3-ს, მაშინ u-ს მივანიჭეთ $x-y$-ის მნიშვნელობა. *x* და y რიცხვები შეიტანეთ კლავიატურიდან.

პრაქტიკული მეცადინეობა № 5

პრაქტიკული მეცადინეობის თემები:

- **while** შეტყობინება
- ვექტორის (vector) template-სპეციალიზაციები სხვადასხვა ტიპისთვის

მასალა დამოუკიდებლად გაცნობისათვის:

1. მაგალითები **while** შეტყობინების გამოყენებაზე:

№	პროგრამის ფრაგმენტი	შედეგი

1	`int counter =0;`
	`while (counter < 3){`
	`  cout<<"Hello, Students!\n";`
	`  counter++;`
	`}`
		`Hello, Students!`
		`Hello, Students!`
		`Hello, Students!`

2	`int number =1;`
	`while (number <= 3){`
	`  cout<<"number = "<<number<<endl;`
	`  number++;`
	`}`
		`number =1`
		`number =2`
		`number =3`

3	`char aso ='F';`
	`while (aso >= 'A'){`
	`  cout<<aso<<' ';`
	`  aso--;`
	`}`
		`F E D C B A`

4	`while ( true )` `  cout<<"usasrulo ganmeoreba\n";`	`//პირობა ყოველთვის ჭეშმარიტია:` `//პროცესი არ დასრულდება`

5	`int n =7;`
	`while ( true ){`
	`  cout<<"n = "<<n<<endl;`
	`  n--;`
	`  if(n == 4) break;`
	`}`
		`n =7`
		`n =6`
		`n =5`

6	`int n =2;`
	`while (n <= 6){`
	`  cout<<"n = "<<n<<endl;`
	`  n +=2;`
	`}`
		`n =2`
		`n =4`
		`n =6`

7	`int n =2;`
	`while ( true ){`
	`  cout<<"n = "<<n<<endl;`
	`  if(n == 6) break;`
	`  n +=2;`
	`}`
		`n =2`
		`n =4`
		`n =6`

8	`int counter =1;`
	`double x =1.2;`
	`while ( true ){`
	`  cout<<x<<endl;`
	`  x +=0.1;`
	`  if(counter == 5) break;`
	`  counter ++;`
	`}`
		`1.2`
		`1.3`
		`1.4`
		`1.5`
		`1.6`

9	`while ( false )`
	`  cout<<"Sesruldeba?\n";`
	`cout<<"ar Sesrulda\n";`
		`ar Sesrulda`
		`// პირობა თავიდანვე მცდარია:`
		`//შეტყობინება არ შესრულდება`

2. მაგალითები **vector** –ის სხვადასხვა სპეციალიზაციის გამოყენებაზე:

№	პროგრამის ფრაგმენტი	შედეგი
1	cout<<"test #1 of vector"<<endl; vector<int> vect1; cout<<"size of vect1 : "<<vect1.size()<<endl;	//ცარიელი ვექტორი test #1 of vector size of vect1 : 0

2	cout<<"test #2 of vector"<<endl;
	vector<int> vector1;
	//ორ ელემენტს ვამატებთ ვექტორში
	vector1.push_back(7777);
	vector1.push_back(2345);
	cout<<"first element is :"<<vector1[0]<<endl;
	cout<<"second element is :"<<vector1[1]<<endl;
		test #2 of vector
		first element :7777
		second element :2345

3	cout<<"test #3 of vector"<<endl;
	vector<int> vector1(3);
	cout<<"vector1.size()="<<vector1.size()<<endl;
	cout<<"first element is:"<<vector1[0] <<endl;
	cout<<"second element is:"<<vector1[1]<<endl;
	cout<<"3-th element is:"<<vector1[2]<<endl;
	cout<<"4-rd element is:"<<vector1[3]<<endl;
		//სამი ელემენტის ადგილი
		//იჯავშნება და ივსება
		// ნულებით
		test #3 of vector
		vector1.size()=3
		first element is:0
		second element is:0
		3-th element is:0
		//Debug Assetion
		// Failed!
		//Expression: vector
		// subscript out of
		// range

4	cout<<"test #4 of vector"<<endl;
	vector<double> vector1(2);
	vector1[0] = 11.11;
	vector1[1] = 22.22;
	cout<<"first element:"<<vector1[0] <<endl;
	cout<<"second element:"<<vector1[1]<<endl;
		test #4 of vector
		first element:11.11
		second element:22.22

5	cout<<"test #5 of vector"<<endl;
	vector<double> vector1(2);
	vector1[0] = 11.1;
	vector1[1] = 22.2;
	vector1.push_back(77.77);
	vector1.push_back(23.45);
	cout<<"vector1[0]="<<vector1[0] <<endl;
	cout<<"vector1[1]="<<vector1[1]<<endl;
	cout<<"vector1[2]="<<vector1[2]<<endl;
	cout<<"vector1[3]="<<vector1[3]<<endl;
		test #5 of vector
		vector1[0]=11.1
		vector1[1]=22.2
		vector1[2]=77.77
		vector1[3]=23.45

6	cout<<"test #6 of vector"<<endl;
	string str;
	vector<string> a(2);
	a[0] = "Good ";
	a[1] = "By";
	str = a[0]+' '+a[1];
	cout<<str<<endl;
		test #6 of vector
		Good By

7	cout<<"test #6 of vector"<<endl;
	string str;
	vector<string> vctr;
	vctr.push_back("Good");
	vctr.push_back("By!");
	str = vctr[0]+' '+vctr[1];
	cout<<str<<endl;
		test #6 of vector
		Good By!

საუდიტორიო სამუშაო:

1. კლავიატურიდან შეიტანეთ 5 მთელი რიცხვი და ჩაწერეთ ვექტორში. შემდეგ დაბეჭდეთ ვექტორის ელემენტები ინდექსების ჩვენებით:

```
#include <iostream>
#include <vector>
using namespace std;
int main(){
  vector<int> a;
  int number, index = 0;
  while(index < 5){
     cin>> number;
     a.push_back(number);
     index++;
  }
  index = 0;
  while(index < a.size()){
     cout<<"a[ "<<index<<" ] = "<<a[index]<<endl;
     index++;
  }
  //system("PAUSE");
  return 0;
}
```

პროგრამის შესრულების შედეგია:

```
1 2 3 4 5
a[ 0 ] = 1
a[ 1 ] = 2
a[ 2 ] = 3
a[ 3 ] = 4
a[ 4 ] = 5
Press any key to continue . . .
```

2. კლავიატურიდან შეიტანეთ 10 ნამდვილი რიცხვი (შეტანა დაამთავრეთ სიმბოლოს აკრეფით) და ჩაწერეთ ვექტორში. შემდეგ იპოვეთ ამ რიცხვების ჯამი და დაბეჭდეთ.

```
#include <iostream>
#include <vector>
using namespace std;
int main(){
  vector<double> v;
  double x;
  while(cin>>x)
     v.push_back(x);
  double sum =0;
  int i = 0;
  while(i < v.size()){
     sum += v[i];
     i++;
  }
  cout<<"jami = "<<sum<<endl;
  //system("PAUSE");
  return 0;
}
```

პროგრამის შესრულების შედეგია:

```
1.2 0.3 12.25 -11.25 3.0 20.5 -10 -5.5 0 2.125 |
jami = 12.625
Press any key to continue . . .
```

3. დაწერეთ პროგრამა, რომელიც კლავიატურიდან შემოტანილ მთელ რიცხვებს შორის იპოვის და დაბეჭდავს უდიდესს.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int main(){
  vector<int> v;
  int number;
  while(cin>>number)
     v.push_back(number);
  sort(v.begin(), v.end());
  cout<<"udidesi = "<<v[v.size()-1]<<endl;
  //system("PAUSE");
  return 0;
}
```

პროგრამის შესრულების შედეგია:

```
9 14 0 -3 27 -15 8 9 25 *
udidesi = 27
Press any key to continue . . .
```

4. კლავიატურიდან შემოიტანეთ წინადადება, რომელშიც სიტყვები გამოყოფილია ჰარით. შეტანა დაამთავრეთ ახალ სტრიქონზე Ctrl+z კლავიშების კომბინაციის აკრეფით. დაბეჭდეთ სიტყვების რაოდენობა და ყველა განსხვავებული სიტყვა.

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <string>
using namespace std;
int main(){
  vector<string> sentence;
  string word;
  while(cin>>word)
     sentence.push_back(word);
  cout<<"Number of words = "<<sentence.size()<<endl;
  sort(sentence.begin(), sentence.end());
  cout<<sentence[0]<<endl;
  int i =1;
  while(i < sentence.size()){
     if(sentence[i-1] != sentence[i])
       cout<<sentence[i]<<endl;
     ++i;
  }
  //system("PAUSE");
  return 0;
}
```

პროგრამის შესრულების შედეგია:

```
a plan a map a dog
^Z
Number of words = 6
a
dog
map
plan
Press any key to continue . . .
```

დამოუკიდებელი სამუშაო:

1. რას დაბეჭდავს შემდეგი ფრაგმენტი? ახსენით და შემდეგ შეამოწმეთ კომპიუტერზე:

№	პროგრამის ფრაგმენტი

1	`int k =1; char p ='A';`
	`  while( k < 4 ){`
	`  cout<<p++<<endl;`
	`  ++k;`
	`  }`

2	`int x =5;`
	`  while ( -25 || 0 ){`
	`  cout<<"ar dasruldeba\n";`
	`  x++;`
	`  }`

3	`int x =5;`
	`  while ( 10-10 && x > 7 ) {`
	`  cout<<"ar shesruldeba\n";`
	`  x++;`
	`  }`

4	`int n =7, s =30;`
	`  while ( true ){`
	`  s -=n;`
	`  n--;`
	`  if(n == 4) break;`
	`  }`
	`cout<<"s = "<<s<<endl;`

5	`int n =2, p =1;`
	`  while( n <= 7 ){`
	`  ++n;`
	`  if( n%2 == 0) continue;`
	`  p *=n;`
	`  }`
	`  cout<<"p ="<<p<<endl;`

2. გაარჩიეთ პროგრამა, რომელიც "temperature.in" ფაილიდან შემოტანილ საშუალო თვიური ტემპერატურის 12 მაჩვენებლის (თვეების) მიხედვით ბეჭდავს წელიწადის საშუალო ტემპერატურას.

```
#include <iostream>
#include <fstream>
using namespace std;
int main(){
  double temps, S =0;
  ifstream fin ("temperature.in");
  while( fin >> temps )
     S += temps;
  cout << "sashualo temperatura = "<< S/12 <<endl;
  //system("PAUSE");
  return 0;
}
```

პროგრამის შესრულების შედეგია:

temperature.in ფაილი	გამოსატანი ეკრანი
12.5 15 14.5 16 18 25.5 35 32.5 34.5 28 20 17	sashualo temperatura = 22.375 Press any key to continue . . .

3. გაარჩიეთ პროგრამა, რომელიც კლავიატურიდან შემოტანილ სტრიქონში ითვალის ასო 'a'–ს რაოდენობას.

```
#include <iostream>
#include <string>
using namespace std;
int main(){
  string Str;
  int counter =0, i =0;
  getline(cin, Str);
  while(i < Str.size()){
     if( Str[i] == 'a' )
       counter++;
     ++i;
  }
  cout<< "striqonSi aris " << counter
      << " aso 'a' "<< endl;
  //system("PAUSE");
  return 0;
}
```

პროგრამის შესრულების შედეგია:

```
a man a cat a dog
striqonSi aris 5 aso 'a'
Press any key to continue . . .
```

დავალება:

ა). შეასრულეთ პროგრამა სხვადასხვა სტრიქონისთვის და დარწმუნდით მიღებული შედეგების სისწორეში;

ბ). პროგრამის `getline(cin, Str);` შეტყობინება შეცვალეთ `cin >> Str;` – ით და ისევ შეასრულეთ პროგრამა. შეგექმნათ პრობლემა?

დაიმახსოვრეთ: თუ სტრიქონი შედგება რამოდენიმე სიტყვისაგან (ანუ შეიცავს სიტყვების გამყოფებს - ჰარის ან/და ტაბულაციის - სიმბოლოებს) კლავიატურიდან მის შესატანად გამოიყენება ფუნქცია `getline` ფორმატით `getline(cin, <სტრიქონის სახელი>)`, რადგან `cin >> <სტრიქონის სახელი>` -ის შემთხვევაში წაიკითხება მხოლოდ პირველი სიტყვა (პირველივე გამყოფ სიმბოლომდე).

4. კლავიატურიდან შეიტანეთ მთელი რიცხვები, ჩაწერეთ სათანადო ვექტორში და შემდეგ დაბეჭდეთ მათი ჯამის მეოთხედი.

5. კლავიატურიდან შეიტანეთ მთელი რიცხვები, ჩაწერეთ სათანადო ვექტორში და შემდეგ დაბეჭდეთ მათი საშუალო არითმეტიკული.

6. კლავიატურიდან შეიტანეთ სიმბოლოთა მიმდევრობა, ჩაწერეთ სათანადო ვექტორში და შემდეგ დაბეჭდეთ შეტანილი სიმბოლოების რაოდენობა.

7. კლავიატურიდან შეიტანეთ 12 მთელი რიცხვი და დაბეჭდეთ მათი საშუალო არითმეტიკული.

8*. კლავიატურიდან შეიტანეთ ტექსტი – წერტილით დამთავრებული რამოდენიმე წინადადება, პირველ წინადადებაში დაადგინეთ: `'a'` სიმბოლო გვხვდება მეტად თუ `'b'`.

9. კლავიატურიდან შეიტანეთ რიცხვების რაოდენობა - `m`, შემდეგ `m` ცალი მთელი რიცხვი და დაბეჭდეთ მათი ნამრავლი.

10. კლავიატურიდან შეიტანეთ მთელი რიცხვი `m`, შემდეგ დაბეჭდეთ `m` რიცხვის ფაქტორიალი. რა სახის პრობლემა შეიძლება შეგექმნათ?

ლაბორატორიული მეცადინეობა № 5

ლაბორატორიული სამუშაოს აღწერა:

- ფაილიდან რიცხვების, სტრიქონების და სიმბოლოების წაკითხვა
- პროგრამის შედეგის ფაილში გამოტანა

ლაბორატორიული სამუშაო:

ამოცანა 1. `data.txt` ფაილი შეიცავს 3 სიმბოლოს. დაბეჭდეთ ეს სიმბოლოები `output.txt` ფაილში შებრუნებული რიგით.

მითითება:

- შექმენით პროექტი სახელით `lp 5.1`, დაამატეთ მასში `5-1.cpp` ფაილი, რომელშიც აკრიფეთ პროგრამა.
- დაამატეთ პროექტში ტექსტური ფაილი `data.txt` შემდეგი გზით: `Project -> Add New Item -> Text File(.txt); Name` ველში ჩაწერეთ ფაილის სახელი `data` (გაფართოების გარეშე) და დააჭირეთ `Add` ღილაკს (ან `Enter` -ს).
- ჩაწერეთ `data.txt` ფაილში 3 სიმბოლო.

შესაბამის პროგრამას აქვს სახე:

```
//////////////////////////////////////////////////
//  ავტორი:
//  პროგრამა: ფაილიდან სიმბოლოების წაკითხვა და
//            სხვა ფაილში შებრუნებული რიგით ჩაწერა
//////////////////////////////////////////////////
#include <iostream>
#include <fstream>
using namespace std;
int main(){
   char a, b, c;
   ifstream ifs("data.txt");
   ifs>>a>>b>>c;
   ofstream ofs("output.txt");
   ofs<<c<<' '<<b<<' '<<a;
   return 0;
}
```

პროგრამის გაშვების შემდეგ `lp 5.1` პროექტის შიგა `lp 5.1` საქაღალდეში ავტომატურად შეიქმნება `output.txt` ფაილი. გახსენით იგი (მაგალითად, `File -> Open -> File` და გახსენით `output.txt` ფაილი), გააანალიზეთ პროგრამის მუშაობის შედეგი. მაგალითად,

ფაილი `data.txt`	ფაილი `output.txt`
XYO	O Y X

მითითება: ერთდროულად გახსნილი რამოდენიმე ფაილის შემთხვევაში `CTRL+F6` კლავიშების კომბინაცია ახორციელებს გადართვას მათ შორის.

ამოცანა 2. `monacemebi.in` ფაილში ჩაწერილია 3 მთელი რიცხვი. `pasuxi.out` ფაილში დაბეჭდეთ ამ რიცხვების საშუალო არითმეტიკული.

შესაბამის პროგრამას აქვს სახე:

```
//////////////////////////////////////////////////
//  ავტორი:
//  პროგრამა: ფაილიდან წაკითხული მთელი რიცხვების
//        საშუალო არითმეტიკულის სხვა ფაილში ბეჭდვა
//////////////////////////////////////////////////
#include <iostream>
```

```
#include <fstream>
using namespace std;
int main(){
   int a, b, c;
   double average;
   ifstream fin( "monacemebi.in" );
   fin>>a>>b>>c;
   average =(a+b+c)/3.;
   ofstream fout( "pasuxi.out" );
   fout<<a<<' '<<b<<' '<<c
       <<" ricxvebis sashualo ariTmetikuli = "
       <<average;
   return 0;
}
```

შეასრულეთ პროგრამა რამდენიმეჯერ რიცხვთა სხვადასხვა სამეულებისათვის. შეამოწმეთ `pasuxi.out` ფაილში მიღებული შედეგები.

ამოცანა 3. `input.in` ფაილის პირველ სტრიქონში ჩაწერილია სტუდენტის გვარი, მეორეში კი – მისი სახელი. `output.out` ფაილის პირველ სტრიქონში ჩაწერეთ ამ სტუდენტის სახელი, მეორეში კი გვარი.

შესაბამის პროგრამას აქვს სახე:

```
//////////////////////////////////////////////
// ავტორი:
// პროგრამა: ფაილიდან წაკითხული სტრიქონების
//        შებრუნებული რიგით სხვა ფაილში ბეჭდვა
//////////////////////////////////////////////
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main(){
   string saxeli, gvari;
   ifstream fin("input.in");
   ofstream fout("output.out");
   fin>> gvari >> saxeli;
   fout<< saxeli << endl << gvari;
   fin.close();
   fout.close();
   return 0;
}
```

დავალება: გაარკვიეთ რა მოხდება, თუ `input.in` ფაილში გვარი და სახელი ჩაწერილია ერთ სტრიქონში, გამოყოფილია ჰარით და გვინდა დავბეჭდოთ გვარი და სახელი `output.out` ფაილში შებრუნებული რიგით: ა). ერთ სტრიქონად; ბ) ორ სტრიქონად.

ამოცანა 4. `ricxvebi.txt` ფაილში ჩაწერილია 3 მთელი რიცხვი. `shedegi.txt` ფაილში გადაიტანეთ მხოლოდ ის რიცხვები, რომლებიც ეკუთვნის `[10,30]` შუალედს.

შესაბამის პროგრამას აქვს სახე:

```
//////////////////////////////////////////////////////////////
//  ავტორი:
//  პროგრამა: ფაილის სტრიქონებში მოცემული რიცხვებიდან
//       [10,30] შუალედში მოთავსებული რიცხვების სხვა ფაილში ბეჭდვა
//////////////////////////////////////////////////////////////
#include <iostream>
#include <fstream>
using namespace std;
```

```
int main(){
   int ricxvi;
   ifstream fin( "ricxvebi.txt" );
   ofstream fout( "shedegi.txt" );
   fin >>ricxvi;
   if(ricxvi >=10 && ricxvi <=30) fout<<ricxvi<<endl;
   fin >>ricxvi;
   if(ricxvi >=10 && ricxvi <=30) fout<<ricxvi<<endl;
   fin >>ricxvi;
   if(ricxvi >=10 && ricxvi <=30) fout<<ricxvi<<endl;
   fin.close();
   fout.close();
   return 0;
}
```

დავალება:

1. შეასრულეთ პროგრამა და გააანალიზეთ `shedegi.txt` ფაილი.
2. გადაწერეთ პროგრამა განმეორების შეტყობინების გამოყენებით, შეასრულეთ იგი და გააანალიზეთ შედეგი. **მითითება:** პროგრამაში 3 –ჯერ გამეორებული შეტყობინებები

```
fin >>ricxvi;
if(ricxvi >=10 && ricxvi <=30) fout<<ricxvi<<endl;
```

შეცვალეთ შემდეგი ფრაგმენტით:

```
int mTvleli = 1;
while( mTvleli <= 3 ){
   fin >>ricxvi;
   if(ricxvi >=10 && ricxvi <=30) fout<<ricxvi<<endl;
   mTvleli++;
 }
```

დამოუკიდებელი სამუშაო:

1. მოცემული ფაილი შეიცავს 3 სიმბოლოს. დაადგინეთ სიმბოლოთა შორის უმცირესი და დაბეჭდეთ ეს სიმბოლო და მისი კოდი სხვა ფაილში.
2. მოცემული ფაილი შედგება 2 სტრიქონისგან. თითოეულში ჩაწერილია მთელი რიცხვი. მეორე ფაილში ჩაწერეთ ამ რიცხვების
   ა) ჯამი; ბ) ნამრავლი;
   გ) საშუალო გეომეტრიული.
   მითითება: $a_1, a_2, \ldots, a_n$ რიცხვების საშუალო გეომეტრიული გამოითვლება შემდეგი ფორმულით: საშუალო გეომეტრიული $= \sqrt[n]{a_1 \cdot a_2 \cdot \ldots \cdot a_n}$.
3. მოცემული ფაილი შეიცავს სტრიქონს. მეორე ფაილში ჩაწერეთ ამ სტრიქონის სიგრძის მნიშვნელობა.
4. ერთ ფაილში, სახელად `weight.in` წერია სამი სახის პროდუქტის წონა. მეორე ფაილში, სახელად `price.in` წერია ამ პროდუქტების ფასები. მესამე ფაილში გამოიტანეთ მოცემული რაოდენობის პროდუქტების შეძენაზე დახარჯული თანხის ოდენობა.

   გაარჩიეთ და გააანალიზეთ შესაბამისი C++–პროგრამა:

```
#include <iostream>
#include <fstream>
using namespace std;
int main(){
  double a, b, c =0;
  ifstream ifs1("weight.in");
  ifstream ifs2("price.in");
```

```
  while( ifs1 >> a, ifs2 >> b )
     c += a*b;
  ofstream ofs("answer.out");
  ofs<<c;
  return 0;
}
```

პროგრამის მუშაობის შედეგია:

ფაილი weight.in	ფაილი price.in	ფაილი answer.out
100 50.5 200	3.5 10 7.5	2355

პრაქტიკული მეცადინეობა № 6

პრაქტიკული მეცადინეობის თემები:

- **for** შეტყობინება
- რამდენიმე მარტივი ფუნქცია
- შემთხვევითი რიცხვების გენერირება
- სხვადასხვა ტიპის ვექტორთან მუშაობა

მასალა დამოუკიდებლად გაცნობისათვის:

1. მაგალითები **for** შეტყობინების გამოყენებაზე:

№	პროგრამის ფრაგმენტი	შედეგი

1	`for(int i=1; i<4; i++)`
	`  cout << "i = " << i << endl;`
		`i =1`
		`i =2`
		`i =3`

2	`for(int i=3; i>=2; i--)` `  puts("iswavle C++");`	`iswavle C++` `iswavle C++`
3	`for(char c='K'; c>'D'; --c)` `  cout << c << ' ';`	`K J I H G F E`

4	`for(int n=5; n<=15; n+=5)`
	`  cout << "n = " << n << endl;`
		`n =5`
		`n =10`
		`n =15`

5	`for(char p='A'; p<'F'; ++p)` `  cout << p << ' ';`	`A B C D E`

6	`for(int a=5; a>=2; a--){`
	`  if(a%3 == 0) break;`
	`  cout << "a = " << a << endl;`
	`}`
		`a =5`
		`a =4`

7	`for(int a=5; a>=2; a--){`
	`  if(a%3 == 0) continue;`
	`  cout << "a = " << a << endl;`
	`}`
		`a =5`
		`a =4`
		`a =2`

8	`for(int i=1; i<50; i++){`
	`  if (i%7 != 0) continue;`
	`  cout << I << ' ';`
	`}`
		`7 14 21 28 35 42 49`

9	`for(int i=7; i<50; i+=7)` `  cout << I << ' ';`	`7 14 21 28 35 42 49`
10	`for(int x=10; x>5; x++)` `  puts("usasrulo ganmeoreba");`	პირობა ყოველთვის ჭეშმარიტია: პროცესი არ დასრულდება

11	`for(int k=5; true; k++){`
	`  if(k%2 == 0) continue;`
	`  cout << "k = " << k << endl;`
	`  if(k == 9) break;`
	`}`
		`k =5`
		`k =7`
		`k =9`

12	`for(int n=5; n<3; n++)` `  cout << "n = " << n << endl;`	პირობა თავიდანვე მცდარია: შეტყობინება არ შესრულდება

2. `time(x)` ფუნქციით სარგებლობისათვის საჭიროა `#include <ctime>` ბრძანების პროგრამაში ჩართვა. `time(NULL)` წარმოადგენს სისტემის მიმდინარე კალენდარულ დროს, გაზომილს წამებში. დროის ათვლა მიმდინარეობს გარკვეული თარიღიდან.

#	პროგრამის ფრაგმენტი	შედეგი	გაშვების დრო
1	`cout<<time(NULL)<<endl;`	1319571546	11:39 PM
2	`cout<<time(NULL)<<endl;`	1319571606	11:40 PM
3	`cout<<time(NULL)<<endl;`	1319571906	11:45 PM

3. ფუნქცია `rand()` აბრუნებს მთელი ფსევდოშემთხვევითი რიცხვის გენერირებას. ყოველი მიმართვისას ფუნქცია აბრუნებს **მთელ** რიცხვს `[0,RAND_MAX]` შუალედიდან. `RAND_MAX`, ჩვეულებრივ, არის `SHRT_MAX –ის` ტოლი.

#	პროგრამის ფრაგმენტი	შედეგი
1	`cout<<rand()<<endl;` `cout<<rand()<<endl;` `cout<<rand()<<endl;`	41 18467 6334

2	`/* იგივე ფრაგმენტი შესრულდა 2 საათის შემდეგ */`
	`cout<<rand()<<endl;`
	`cout<<rand()<<endl;`
	`cout<<rand()<<endl;`
		
		41
		18467
		6334

4. სხვადასხვა ტიპის ვექტორთან მუშაობის მაგალითები:

№	პროგრამის ფრაგმენტი	შედეგი

1	`/* ვექტორში ჩავწეროთ სამი რიცხვი და შემდეგ დავბეჭდოთ ვექტორის ელემენტები */`
	`vector<int> v;`
	`v.push_back(23);`
	`v.push_back(-70);`
	`v.push_back(101);`
	`for(int i=0; i<v.size(); i++ )`
	`  cout << v[i]<< endl;`
		23
		-70
		101

2	`/* კლავიატურიდან შევიტანოთ ორი რიცხვი და ჩავწეროთ ვექტორში */`
	`vector<int> a;`
	`int number;`
	`for(int m=1; m<=2; m++ ){`
	`  cin>>number;`
	`  a.push_back(number);`
	`}`
	`cout<<"first element is "<< a[0] <<endl;`
	`cout<<"second element is "<< a[1]<<endl;`
	`cout<<"vector contains "<< a.size()`
	`  <<" elements\n";`
		// შეტანა
		10 -100
		
		// გამოსატანი ეკრანი
		first element is 10
		second element is -100
		vector contains 2 elements

3	`/* ვექტორში ჩავწეროთ სამი სიტყვა და შემდეგ დავბეჭდოთ ვექტორის ელემენტები შებრუნებული რიგით*/`
	`vector<string> words;`
	`words.push_back("Mariam");`
	`words.push_back("Giorgi");`
	`words.push_back("Elene");`
	`for(int i=words.size()-1; i>=0; i-- )`
	`  cout << words[i] << endl;`
		Elene
		Giorgi
		Mariam

4	/* ვექტორში ჩავწეროთ სამი ნამდვილი რიცხვი, შემდეგ დავითვალოთ და დავბეჭდოთ მათი ჯამი */
	vector<double> reals;
	reals.push_back(2.25);
	reals.push_back(10.75);
	reals.push_back(-5);
	double sum =0;
	for(int k=0; k<reals.size(); ++k)
	sum += reals[k];
	cout << "jami = " << sum << endl;
		jami = 8

5	/* ვექტორში ჩავწეროთ 5 შემთხვევითი რიცხი და შემდეგ დავბეჭდოთ ვექტორის ელემენტები*/
	vector<int> vec;
	int number;
	for(int count=1; count<=5; count++){
	number =rand();
	vec.push_back(number);
	}
	for(int i=0; i<vec.size(); i++)
	cout << vec[i] << endl;
		41
		18467
		6334
		26500
		19169

სააუდიტორიო სამუშაო:

1. ვექტორში ჩავწეროთ პირველი 10 ნატურალური რიცხვის კვადრატი და შემდეგ დავბეჭდოთ ეს ვექტორი.

```
#include <iostream>
#include <vector>
using namespace std;
int main(){
  vector<int> vec;
  for(int n=1; n<11; n++)
     vec.push_back(n * n);
  for(int i=0; i<vec.size(); i++)
     cout << vec[i] << '\t';
  cout<<endl;
  system("PAUSE");
  return 0;
}
```

პროგრამის შესრულების შედეგია:

```
1       4       9       16      25      36      49      64      81      100
Press any key to continue . . .
```

2. შექმენით ფუნქცია, რომლის პროტოტიპია `int square(int k);` და რომელიც გამოითვლის ნატურალური რიცხვის კვადრატს. გამოიყენეთ ფუნქცია წინა ამოცანის ამოხსნისთვის.

```
#include <iostream>
#include <vector>
using namespace std;
int square (int );
int main(){
  vector<int> vec;
  for(int n=1; n<11; n++)
     vec.push_back( square (n) );
  for(int i=0; i< vec.size(); i++)
```

```
      cout << vec[i] << '\t';
   cout<<endl;
   system("PAUSE");
   return 0;
}
int square (int k){
    return k * k;
}
```

3. შექმენით კიდევ ერთი ფუნქცია პროტოტიპით `void printVector(vector<int> );` რომელიც დაბეჭდავს ვექტორის ელემენტებს. შემდეგ გამოიყენეთ ეს ფუნქცია წინა ამოცანაში.

```
#include <iostream>
#include <vector>
using namespace std;
int square (int );
void printVector(vector<int> );
int main(){
   vector<int> vec;
   for(int n=1; n<11; n++)
      vec.push_back( square (n) );
   printVector( vec );
   system("PAUSE");
   return 0;
}
int square (int k){
    return k * k;
}
void printVector(vector<int> x){
    for(int i=0; i< x.size(); i++)
      cout << x[i] << '\t';
    cout << endl;
}
```

4. ჩაწერეთ ვექტორში N შემთხვევითი მთელი რიცხვი და დაბეჭდეთ ვექტორი. შემდეგ დაალაგეთ ვექტორის ელემენტები ზრდადობით და ისევ დაბეჭდეთ ვექტორი. პროგრამაში გამოიყენეთ ადრე შექმნილი ფუნქცია `printVector`.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
void printVector(vector<int> );
int main(){
   vector<int> vec;
   int N;
   cout<<"ShemoitaneT ricxvebis raodenoba : ";
   cin>>N;
   for(int count=1; count<=N; count++)
      vec.push_back(rand());
   printVector( vec );
   sort( vec.begin(), vec.end() );
   printVector( vec );
   system("PAUSE");
   return 0;
}
void printVector(vector<int> x){
```

```
    for(int i=0; i<x.size(); i++)
      cout<<x[i]<<"  ";
    cout<<endl;
}
```

პროგრამის შესრულების შედეგია:

```
ShemoitaneT ricxvebis raodenoba : 5
41  18467  6334  26500  19169
41  6334  18467  19169  26500
Press any key to continue . . .
```

დამოუკიდებელი სამუშაო:

1. გაარჩიეთ პროგრამა, რომელიც ითვლის 310 –იდან 452 –მდე 7-ის ჯერადი რიცხვების ჯამს.

```
#include <iostream>
using namespace std;
int main(){
    int a;          // მთელი რიცხვი
    int s =0;       // 7-ის ჯერადი რიცხვების ჯამი
    //რიცხვებისთვის 310-დან 452-ის ჩათვლით, ბიჯით 1, სრულდება:
    for(a = 310; a <= 452; a++)
       if( a%7 == 0 ) s += a;  //თუ რიცხვი 7-ის ჯერადია დავამატოთ იგი ჯამს
    cout<<"s = "<<s<<endl;
    system("pause");
    return 0;
}
```

პროგრამის შესრულების შედეგია:

```
s =7630
Press any key to continue . . .
```

2. ამოცანის პირობაა: შეკრიბეთ შემთხვევითი რიცხვები, ვიდრე ჯამი არ გახდება 10 –ის ჯერადი, ოღონდ ჯამში არ გაითვალისწინოთ 13-ზე დამთავრებული რიცხვები. გაარჩიეთ შესაბამისი C++ –ის პროგრამა.

```
#include <iostream>
using namespace std;
int main (){
     int a;         // შემთხვევითი რიცხვი
     int s =0;      // რიცხვების ჯამი
     while ( true ){
        a =rand();
        if(a%100 == 13)
           continue;
        s +=a;
        if(s%10 == 0)
           break;
     }
     cout<< "s = " << s <<'\n';
     system("PAUSE");
     return 0;
}
```

პროგრამის შესრულების შედეგია:

```
s =554810
Press any key to continue . . .
```

3. ჩაწერეთ სათანადო ვექტორში კლავიატურიდან შემოტანილი 5 სიტყვა და შემდეგ დაბეჭდეთ ვექტორის ელემენტები. ვექტორის ელემენტების დასაბეჭდათ შექმენით და გამოიყენეთ ფუნქცია პროტოტიპით **void** printWords(vector<string>);

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;
void printWords(vector<string> );
int main(){
  vector<string> W;
  string word;
  for(int count=1; count<=5; count++){
     cin>>word;
     W.push_back(word);
  }
  printWords(W);
  system("PAUSE");
  return 0;
}
void printWords(vector<string> a){
   for(int i=0; i<a.size(); i++ )
      cout<<a[i]<<endl;
}
```

პროგრამის შესრულების შედეგია:

```
Tbilisi Batumi KuTaisi Gori Telavi
Tbilisi
Batumi
KuTaisi
Gori
Telavi
Press any key to continue . . .
```

4. რას დაბეჭდავს შემდეგი ფრაგმენტი? ახსენით და შემდეგ შეამოწმეთ კომპიუტერზე:

№	პროგრამის ფრაგმენტი

1	`int k; char p;`
	`for(k=1, p='A'; k<9; ++k)`
	`  cout << p++ << endl;`

2	`int x;`
	`for(x=10; -10 || 0; x++)`
	`  puts("ar dasruldeba");`

3	`int x;`
	`for(x=10; x>3 && 5-5; x++)`
	`  puts("ar shesruldeba");`

4	`int s =30;`
	`for(int n =7; true; --n){`
	`  s -=n;`
	`  if(n == 4) break;`
	`}`
	`cout << "s = " << s << endl;`

5	`int n =5, p =1;`
	`for(  ; n<=11;  n++){`
	`  if( n%2 == 0) continue;`
	`  p *=n;`
	`}`
	`cout << "p = " << p << endl;`

5. კლავიატურიდან შევიტანოთ მთელი რიცხვები, ჩავწეროთ სათანადო ტიპის ვექტორში და შემდეგ დავბეჭდოთ მათი ჯამის მეოთხედი.

6. კლავიატურიდან შევიტანოთ მთელი რიცხვები, ჩავწეროთ სათანადო ტიპის ვექტორში და შემდეგ დავბეჭდოთ მათი საშუალო არითმეტიკული.

7. კლავიატურიდან შეიტანეთ სიმბოლოთა მიმდევრობა, ჩაწერეთ სათანადო ტიპის ვექტორში და შემდეგ დაბეჭდეთ შეტანილი სიმბოლოების რაოდენობა.

8. იპოვეთ და დაბეჭდეთ 7-დან 32-მდე რიცხვების ჯამი.

9. იპოვეთ 14 –იდან 102 –მდე კენტი რიცხვების ჯამი.

10. იპოვეთ 4 –იდან 123 –მდე ლუწი რიცხვების ჯამი.

11. იპოვეთ 5 –იდან 103 –მდე 5-ის ჯერადი რიცხვების ჯამის მეოთხედი.

12. იპოვეთ `m`-იდან `n`-მდე ნატურალურ რიცხვთა ჯამი. `m` და `n` შემოიტანეთ კლავიატურიდან.

13. `a` და `b` – მთელი რიცხვებია. `c` ცვლადს მიანიჭეთ `a`-დან `b`-მდე რიცხვების საშუალო არითმეტიკული, და შემდეგ დაბეჭდეთ. `a` და `b` შეიტანეთ კლავიატურიდან.

14. კლავიატურიდან შევიტანოთ 12 მთელი რიცხვი და დავბეჭდოთ მათი საშუალო არითმეტიკული.

15. იპოვეთ შემთხვევითი რიცხვების, გარდა ლუწი რიცხვებისა, საშუალო არითმეტიკული ვიდრე მათი ჯამი არ გახდება 21 –ის ჯერადი.

16. იპოვეთ შემთხვევითი რიცხვების, გარდა 5-ის ჯერადი რიცხვებისა, საშუალო არითმეტიკული ვიდრე მათი ჯამი არ გახდება 19 –ის ჯერადი.

17. რამდენჯერმე დაბეჭდეთ `cout << (1.0*rand())/RAND_MAX << endl;` რა ტიპის შემთვევით რიცხვებს ქმნის `(1.0*rand())/RAND_MAX` ? რომელი დიაპაზონიდან?

18*. წინა სავარჯიშოს საფუძველზე, როგორ შევქმნათ შემთხვევითი ნამდვილი რიცხვები დიაპაზონიდან `[a.b]`?

ლაბორატორიული მეცადინეობა № 6

ლაბორატორიული სამუშაოს აღწერა:

- `rand()` და `time()` ფუნქციების გამოყენება
- შემთხვევითი რიცხვების შექმნა და ბეჭდვა

ლაბორატორიული სამუშაო:

ამოცანა 1. დაბეჭდეთ სამი მთელი შემთხვევითი რიცხვი `[0;100]` შუალედიდან.

შესაბამის პროგრამას აქვს სახე:

```
//////////////////////////////////////////
// ავტორი:
// პროგრამა: სამი შემთხვევითი რიცხვის [0;100]
//              შუალედიდან ბეჭდვა
//////////////////////////////////////////
#include <iostream>
using namespace std;
int main(){
  cout<<"random numbers:\n";
  cout<<rand()%101<<' '<<rand()%101<<' '
      <<rand()%101<<endl;
  system("PAUSE");
  return 0;
}
```

დავალება:

გადაწერეთ პროგრამა ისე, რომ შესაძლებელი გახდეს `N` ცალი შემთხვევითი რიცხვის `[0;100]` შუალედიდან დაბეჭდვა, თუ `N`–ის მნიშვნელობას შევიტანთ კლავიატურიდან.

ამოცანა 2. დაწერეთ პროგრამა, რომელიც ბეჭდავს მიმდინარე დროს წამებში და წუთებში..

შესაბამის პროგრამას აქვს სახე:

```
//////////////////////////////////////
// ავტორი:
// პროგრამა: მიმდინარე დროის წამებში და
//              წუთებში ბეჭდვა
//////////////////////////////////////
#include <iostream>
#include <ctime>
using namespace std;
int main(){
  cout<<"mimdinare dro (second): "<<time(NULL)<<endl;
  cout<<"mimdinare dro (minute): "<<time(NULL)/60<<endl;
  system("PAUSE");
  return 0;
}
```

დავალება:

დაამატეთ პროგრამაში შეტყობინება, რომელიც დაბეჭდავს მიმდინარე გროს საათებში.

ამოცანა 3. შექმენით `10` შემთხვევითი მთელი რიცხვი `[65,90]` დიაპაზონიდან – დიდი ინგლისური ასოების კოდები. დაბეჭდეთ ამ კოდების შესაბამისი სიმბოლოები. დასაბეჭდად გამოიყენეთ თქვენ მიერ განსაზღვრული ფუნქცია. შემდეგ დაბეჭდეთ სიმბოლოებს შორის უმცირესი და მისი კოდი.

შესაბამის პროგრამას აქვს სახე:

```
//////////////////////////////////////////////
// ავტორი:
// პროგრამა: 10 შემთხვევითი რიცხვის შექმნა და მათი
//              შესაბამისი სიმბოლოების ბეჭდვა
//////////////////////////////////////////////
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
void printChar(vector<int> );
int main(){
  vector<int> vec;
  int code;
  for(int count=1; count<=10; count++){
     code =rand()%(91 -65) + 65;
     vec.push_back(code);
  }
  printChar(vec);
  sort( vec.begin(), vec.end() );
  printChar(vec);
  cout<<"umciresia "<<(char)vec[0]
      <<" kodiT "<<vec[0]<<endl;
  system("PAUSE");
  return 0;
}
void printChar(vector<int> m){
   for(int i=0; i<m.size(); i++ )
      cout<<(char)m[i]<<' ';
   cout<<endl;
}
```

დავალება:

შეცვალეთ პროგრამა ისე, რომ მან დაბეჭდოს შემთხვევითი კოდების მქონე სიმბოლოები დალაგებული კლებადობით. **მითითება:**

```
sort( vec.begin(), vec.end() );
```

ალგორითმის ნაცვლად გამოიყენეთ მისი სხვა სახე:

```
sort( vec.rbegin(), vec.rend() );
```

შესაბამისად შეცვალეთ უმცირები სიმბოლოს და მისი კოდის ბეჭდვის შეტყობინებაც.

ამოცანა 4. დაწერეთ პროგრამა, რომელიც შექმნის 25 შემთხვევით პატარა ასოს (`['a','z']` შუალედიდან) და შემდეგ დაბეჭდავს უდიდესის კოდს. **მითითება:** ისარგებლეთ ფორმულით:

```
rand()%('z'+1- 'a') + 'a';
```

დამოუკიდებელი სამუშაო:

1. მოცემული ფაილი შეიცავს 3 სიმბოლოს. სიმბოლოთა შორის უმცირესი და უდიდესი მათ კოდებთან ერთად დაბეჭდეთ სხვა ფაილში. ამოცანა გააკეთეთ ორი ხერხით: 1) ვექტორის და დალაგების ალგორითმის გამოყენების გარეშე; 2) სათანადო ტიპის ვექტორის და დალაგების ალგორითმის გამოყენებით.
2. `75` შემთხვევითი რიცხვი დიაპაზონში `[-15,80]` დაბეჭდეთ კლებადობით.
3. კლავიატურიდან შემოიტანეთ `N` რაოდენობის ნამდვილი რიცხვი. დაბეჭდეთ დადებით რიცხვებს შორის უმცირესი და მისი რიგითი ნომერი.

პრაქტიკული მეცადინეობა № 7

პრაქტიკული მეცადინეობის თემები:

- ფუნქციის პარამეტრები, დასაბრუნებელი მნიშვნელობა
- ფუნქციის პროტოტიპი, განსაზღვრა
- ფუნქციის გამოძახება

მასალა დამოუკიდებლად გაცნობისათვის:

1. მარტივი ფუნქციების და ძირითად პროგრამაში მათი გამოძახების ნიმუშები.
   მითითება: იმისათვის რომ ვნახოთ შექმნილი ფუნქციის მუშაობის შედეგი, მოყვანილი ფრაგმენტები უნდა შევავსოთ სრულ პროგრამამდე. მაგალითად, პირველი ფრაგმენტისთვის უნდა დავწეროთ:

```
#include <iostream>
using namespace std;
int my_abs(int );
int main(){
      int a, n;
      cin>>a;
      n = my_abs(a);
      cout<<n<<endl;
   system("pause");
   return 0;
}
   int my_abs(int x){
       if(x >= 0) return x;
       return -x;
   }
```

ხოლო სხვა ნიმუშებში შევცვალოთ ფუნქციის სათაური (პროტოტიპი), `main()`-ის ფრაგმენტი და ფუნქციის განსაზღვრა შესაბამისად.

№	ფუნქცია	ფუნქციის გამოძახება - main()-ის ფრაგმენტი

1	/*ფუნქცია პოულობს მთელი რიცხვის მოდულს*/
	int my_abs(int x){
	if(x >= 0) return x;
	return -x;
	}
		int main(){
		int a, n;
		cin>>a;
		n = my_abs(a);
		cout<<n<<endl;
		. . .
		}

2	/* ფუნქცია პოულობს ნამდვილი რიცხვის მოდულს */
	float my_fabs(float x){
	if(x < 0) return -x;
	return x;
	}
		int main(){
		float y, k;
		cin>>y;
		k = my_fabs(y);
		cout<<k<<endl;
		. . .
		}

3	/* ფუნქცია ადგენს რიცხვის ნიშანს */
	int my_sign(float x){
	if(x > 0) return 1;
	else if(x == 0) return 0;
	return -1;
	}
		int main(){
		float n; int s;
		cin>>n;
		s = my_sign(n);
		cout<<s<<endl;
		. . .
		}

4	/* ფუნქცია პოულობს 2 მთელი რიცხვიდან	int main(){ int m, n, max;

	უდიდესს */
	int my_max2(int a, int b){
	  if(a > b) return a;
	  return b;
	}
		cin>>m>>n;
		  max = my_max2(m,n);
		  cout<<max<<endl;
		  . . .
		}

5	/* ფუნქცია პოულობს 2 ნამდვილი რიცხვიდან უმცირესს */
	float my_fmin2(float x, float y){
	  return (x < y) ? x : y;
	}
		int main(){
		  float a, b, min;
		  cin>>a>>b;
		  min = my_fmin2(a,b);
		  cout<<min<<endl;
		  . . .
		}

6	/* ფუნქცია პოულობს 3 მთელი რიცხვიდან უმცირესს */
	int my_min3(int a, int b, int c){
	  int m = a;
	  if(b < m) m = b;
	  if(c < m) m = c;
	  return m;
	}
		int main(){
		  int k, n, m, min;
		  cin>>k>>n>>m;
		  min = my_min3(k,n,m);
		  cout<<min<<endl;
		  . . .
		}

7	/* ფუნქცია პოულობს 3 ნამდვილი რიცხვიდან უდიდესს */
	float my_fmax3(float a,float b,float c){
	  float m = a;
	  if(b > m) m = b;
	  if(c > m) m = c;
	  return m;
	}
		int main(){
		  float x, y, z, max;
		  cin>>x>>y>>z;
		  max = my_fmax3(x,y,z);
		  cout<<max<<endl;
		  . . .
		}

8	/* ფუნქცია ითვლის მთელი რიცხვის კვადრატს */
	int square(int a){
	  return a * a;
	}
		int main(){
		  int k;
		  cin>>k;
		  cout<<square(k)<<endl;
		  . . .
		}

9	/* ფუნქცია ითვლის ორი მთელი რიცხვის საშუალოს */
	double average(int n, int m){
	  return (n + m)/2.;
	}
		int main (){
		  int a, b; double rez;
		  cin>>a>>b;
		  rez = average(a,b);
		  cout<<rez<<endl;
		  . . .
		}

11	/* ფუნქცია პოულობს შემთხვევით რიცხვს [A,B] შუალედიდან */
	
	int my_rand(int A, int B){
	  return rand()%(B + 1 - A) + A;
	}
		int main(){
		  int A, B, number;
		  cin>>A>>B;
		  number = my_rand(A, B);
		  cout<<number<<endl;
		  . . .
		}

12	/* ფუნქცია ითვლის x რიცხვის მთელ n ხარისხს, $x \neq 0$, $n \geq 0$ */
	float my_pow(float x, int n){
	// precondition: n >= 0, x != 0
	  float d = 1; int i;
	  if(n == 0) return 1;
	  else{
		int main(){
		  float x, degree;
		  int n;
		  cin>>x>>n;
		  degree = my_pow(x, n);
		  cout<<degree<<endl;
		  . . .

	`for(i = 1; i <= n; i++)`
	`  d *= x;`
	`return d;`
	`}`
	`}`
		`}`

2. მარტივი `bool` ტიპის ფუნქციები (ფუნქცია-პრედიკატები) და ძირითად პროგრამაში მათი გამოძახების მაგალითები

№	ფუნქცია	ფუნქციის გამოძახება

1	`/* ფუნქცია ადგენს არის თუ არა მთელი N რიცხვი 3 –ის ჯერადი */`
	`int Multiple_3(int N){`
	`if(N%3 == 0) return 1;`
	`return 0;`
	`}`
		`int main(){`
		`int number, k;`
		`cin>>number;`
		`k = Multiple_3(number);`
		`if(k == 1)`
		`cout<<number<<" aris 3-is"`
		`" jeradi\n";`
		`else cout<<number<<" ar aris"`
		`" 3-is jeradi\n";`
		`. . .`
		`}`

2	`/* ფუნქცია ადგენს არის თუ არა მთელი A რიცხვი ლუწი */`
	`int IsEven(int N){`
	`return N%2 == 0;`
	`}`
		`int main(){`
		`int num;`
		`cin>>num;`
		`if( IsEven(num) )`
		`cout<<num<<" is even\n";`
		`else cout<<num<<" is't"`
		`" even\n";`
		`. . .`
		`}`

3	`/* ფუნქცია ადგენს არის თუ არა ნამდვილი D რიცხვი უარყოფითი */`
	`bool IsNegative(float D){`
	`if(D < 0)`
	`return true;`
	`return false;`
	`}`
		`int main(){`
		`float num;`
		`cin>>num;`
		`if( IsNegative(num) )`
		`cout<<num<<" is negative\n";`
		`else cout<<num<<" is't"`
		`" negative\n";`
		`. . .`
		`}`

4	`/* ფუნქცია ადგენს არის თუ არა მთელი N რიცხვი დადებითი და 5 –ის ჯერადი */`
	`bool func(int N){`
	`if(N > 0 && N%5 == 0)`
	`return true;`
	`return false;`
	`}`
		`int main(){`
		`int number;`
		`bool answer;`
		`cin>>number;`
		`answer = func(number);`
		`if(answer)`
		`puts("Yes");`
		`else puts("No");`
		`. . .`
		`}`

5	`/* ფუნქცია ადგენს არის თუ არა მისი პარამეტრი სიმბოლო-ციფრი */`
	`bool IsDigit(char p){`
	`if(p >= '0' && p <= '9')`
	`return true;`
	`return false;`
	`}`
		`int main(){`
		`char ch;`
		`cin>>ch;`
		`if( IsDigit(ch) )`
		`cout<<ch<<" aris "`
		`"simbolo-cifri\n";`
		`else cout<<ch<<" ar aris "`
		`"simbolo-cifri\n";`

		. . . }

6	/* ფუნქცია ადგენს არის თუ არა მისი პარამეტრი დიდი ასო */
	bool IsUpperLetter(char p){
	if(p >= 'A' && p <= 'Z')
	return true;
	return false;
	}
		int main(){
		char sym;
		cin>>sym;
		if(IsUpperLetter(sym))
		cout<<sym<<" aris"
		" didi aso\n";
		else cout<<sym<<" ar aris"
		" didi aso\n";
		. . .
		}

საუდიტორიო სამუშაო:

1. დაწერეთ ფუნქცია, რომელიც გამოითვლის გამოსახულებას $2x^3+3x^2-5x-10$, სადაც x მთელი რიცხვია. გამოიძახეთ ფუნქცია ძირითად პროგრამაში კლავიატურიდან შემოტანილი მთელი რიცხვისთვის და შედეგი დაბეჭდეთ.

```
#include <iostream>
using namespace std;
int func(int x);   // ფუნქციის პროტოტიპი (ფუნქციაზე განაცხადი)
int main(){
  int a, rezult;
  cout<<"ShemoitaneT mTeli ricxvi  ";
  cin>>a;
  rezult = func(a);   // ფუნქციის გამოძახება
  cout<<"Gamosaxuleba = "<<rezult<<endl;
  //system("PAUSE");
  return 0;
}
// ფუნქციის განსაზღვრა
int func(int x){
  return 2*x*x*x+3*x*x-5*x-10;
}
```

პროგრამა დაბეჭდავს:

```
ShemoitaneT mTeli ricxvi  5
Gamosaxuleba = 290
Press any key to continue . . .
```

2. შექმენით ფუნქცია პროტოტიპით `bool IsLetter(char p)`, რომელიც დაადგენს არის თუ არა მისი `p` პარამეტრი ასო. გამოიძახეთ ფუნქცია ძირითად პროგრამაში კლავიატურიდან შემოტანილი სიმბოლოსთვის და შედეგი დაბეჭდეთ.

```
#include <iostream>
using namespace std;
bool IsLetter(char p);
int main(){
  char sym;
  cout<<"ShemoitaneT simbolo : ";
  cin>>sym;
  if( IsLetter(sym) )
     cout<<sym<<" aris aso\n";
  else cout<<sym<<" ar aris aso\n";
  //system("PAUSE");
```

```
  return 0;
}
bool IsLetter(char p){
  if(p >= 'a' && p <= 'z'
       || p >= 'A' && p <= 'Z')
    return true;
  return false;
}
```

პროგრამა დაბეჭდავს:

```
ShemoitaneT simbolo : R
R aris aso
Press any key to continue . . .
```

3. numbers.dat ფაილში ჩაწერილია მთელი რიცხვები. დაითვალეთ, რამდენი ცხრიანია ფაილში.

```
#include <iostream>
#include <fstream>
#include <vector>
#include <algorithm>
using namespace std;
int main(){
  int number;
  vector<int> N;
  ifstream fin("numbers.dat");
  while(fin>>number)
     N.push_back(number);
  int quantity;
  quantity = count( N.begin(), N.end(), 9);
  cout<<quantity<<endl;
  //system("PAUSE");
  return 0;
}
```

პროგრამის შესრულების შედეგია:

numbers.dat ფაილი	გამოსატანი ეკრანი
1 2 3 4 9 6 7 8 9 10 11 12 13 9 15	3 Press any key to continue . . .

4. numbers.dat ფაილში ჩაწერილია მთელი რიცხვები. დაითვალეთ, მათგან რამდენია კენტი.

```
#include <iostream>
#include <fstream>
#include <vector>
#include <algorithm>
using namespace std;
bool isOdd(int x)       // ფუნქცია-პრედიკატზე განაცხადი
{                       // და მისი განსაზღვრა
   return x % 2 != 0;
}
int main(){
  int number;
  vector<int> N;
  ifstream fin("number.dat");
  while(fin>>number)
```

```
    N.push_back(number);
  int quantity;
  // აქ გამოიყენება ფუნქცია isOdd
  quantity = count_if( N.begin(), N.end(), isOdd);
  cout<<quantity<<endl;
  //system("PAUSE");
  return 0;
}
```

პროგრამის შესრულების შედეგია:

numbers.dat ფაილი	გამოსატანი ეკრანი
1 2 3 4 9 6 7 8 9 10 11 12 13 9 15	9 Press any key to continue . . .

დამოუკიდებელი სამუშაო:

1. დაწერეთ ფუნქცია, რომელიც გამოითვლის გამოსახულებას:
   ა) $x^3 - 4x^2 + 5x - 7$;
   ბ) $0.5x^2 + 2.5x + 12$.
   x მთელი რიცხვია. გამოიძახეთ ფუნქცია ძირითად პროგრამაში კლავიატურიდან შემოტანილი მთელი რიცხვისთვის და შედეგი დაბეჭდეთ.

2. მოცემულია პირობა $f(x) = \begin{cases} 2x + 34, & \text{თუ } x \geq 10, \\ x^2 - x - 1, & \text{წინააღმდეგ შემთხვევაში.} \end{cases}$

   დაწერეთ ფუნქცია, რომელიც გამოითვლის $f(x)$-ის მნიშვნელობას მთელი x რიცხვისთვის. გამოიძახეთ ფუნქცია ძირითად პროგრამაში კლავიატურიდან შემოტანილი მთელი რიცხვისთვის და შედეგი დაბეჭდეთ.

   გაარჩიეთ ამ ამოცანის შესაბამისი პროგრამა:

```
#include <iostream>
using namespace std;
int function(int x);
int main(){
  int a, rezult;
  cout<<"ShemoitaneT mTeli ricxvi  ";
  cin>>a;
  rezult =function(a);
  cout << "Gamosaxuleba = " << rezult << '\n';
  //system("PAUSE");
  return 0;
}
int function(int x){
  if(x >= 10) return 2*x+34;
  return x*x-x-1;
}
```

პროგრამის შესრულების შედეგია:

```
ShemoitaneT mTeli ricxvi  7
Gamosaxuleba = 41
Press any key to continue . . .
```

დავალება:

ა) შეასრულეთ პროგრამა კომპიუტერზე სხვადასხვა მთელი a რიცხვისთვის;
ბ) ახსენით function ფუნქციის ალგორითმი.

3. მოცემულია პირობა:

ა) $f(x)=\begin{cases} 5x^3-12, & \text{თუ } x>3, \\ x^2+2x+4, & \text{წინააღმდეგ შემთხვევაში.} \end{cases}$;

ბ) $f(x)=\begin{cases} 0.2x+3.45, & \text{თუ } x\geq 10, \\ x^2-x-1, & \text{წინააღმდეგ შემთხვევაში.} \end{cases}$;

გ) $f(x)=\begin{cases} 7x^2+3, & \text{თუ } x\geq 1, \\ x-25, & \text{თუ } x=0, \\ x^3+6x+8, & \text{თუ } x\leq -1. \end{cases}$;

დ) $f(x)=\begin{cases} 11.4x-0.3, & \text{თუ } x>5, \\ x^2-2.5x+9, & \text{თუ } x=5, \\ x^3-8, & \text{თუ } x<5. \end{cases}$.

დაწერეთ ფუნქცია, რომელიც გამოითვლის $f(x)$-ის მნიშვნელობას მთელი x რიცხვისთვის. გამოიძახეთ ფუნქცია ძირითად პროგრამაში კლავიატურიდან შემოტანილი მთელი რიცხვისთვის და შედეგი დაბეჭდეთ.

4. მოცემულია პირობა:

ა) $f(x)=\begin{cases} 100.5x^3+50, & \text{თუ } x\leq 20.5, \\ 20x-45.5, & \text{წინააღმდეგ შემთხვევაში.} \end{cases}$;

ბ) $f(x)=\begin{cases} 15x^2+30x+20, & \text{თუ } x\geq 10.5, \\ x^3-40x+50, & \text{თუ } x=10.5, \\ x+25.5, & \text{თუ } x\leq 10.5. \end{cases}$.

დაწერეთ ფუნქცია, რომელიც გამოითვლის $f(x)$-ის მნიშვნელობას ნამდვილი x რიცხვისთვის. გამოიძახეთ ფუნქცია ძირითად პროგრამაში კლავიატურიდან შემოტანილი ნამდვილი რიცხვისთვის და შედეგი დაბეჭდეთ.

5. დაწერეთ ფუნქცია, რომელიც დაადგენს:
   ა) არის თუ არა მთელი N რიცხვი უარყოფითი. N ფუნქციის პარამეტრია;
   ბ) არის თუ არა მთელი K რიცხვი 5 –ის ან 7 –ის ჯერადი. K ფუნქციის პარამეტრია;
   გ) არის თუ არა ნამდვილი M რიცხვი 10 –ზე ნაკლები. M ფუნქციის პარამეტრია;
   დ) ბოლოვდება თუ არა მთელი A რიცხვი 3 –ით ან 9 -ით. A ფუნქციის პარამეტრია;
   ე) წარმოადგენს თუ არა მისი პარამეტრი - სიმბოლო B - პატარა ასოს;
   ვ) არის თუ არა მთელი M რიცხვი სამნიშნა. M ფუნქციის პარამეტრია.

6. დაწერეთ ფუნქცია, რომელიც იპოვის:
   ა) N –დან M –მდე ჩათვლით კენტი რიცხვების საშუალო არითმეტიკულს. N და M ფუნქციის პარამეტრებია;
   ბ) C –დან D –მდე ჩათვლით ლუწი რიცხვების რაოდენობას. C და D ფუნქციის პარამეტრებია;
   გ) a –დან b –მდე ჩათვლით 7 –ის ჯერადი რიცხვების ჯამს. a და b ფუნქციის პარამეტრებია;
   დ) M –დან K –მდე ჩათვლით 3 –ის ჯერადი რიცხვების საშუალო არითმეტიკულს. M და K ფუნქციის პარამეტრებია.

7. შექმენით ფუნქცია, რომელიც:
   ა) გამოითვლის სამი მთელი რიცხვის საშუალო არითმეტიკულს. გამოიყენეთ ფუნქცია ძირითად პროგრამაში და დაბეჭდეთ მიღებული შედეგი;

ბ) დააბრუნებს მთელი რიცხვის კუბს. ეს რიცხვი ფუნქციის პარამეტრს წარმოადგენს. გამოიყენეთ ფუნქცია `main`-ში და დაბეჭდეთ 1-დან 100-მდე რიცხვთა კუბები;

გ) გამოითვლის მართკუთხედის ფართობს (მართკუთხედის სიგრძე და სიგანე – ფუნქციის პარამეტრებია). გამოიყენეთ ფუნქცია ძირითად პროგრამაში და შეადარეთ 2 მართკუთხედის ფართობი;

დ) დააბრუნებს 2 მთელ რიცხვს შორის უმცირესს. რიცხვები ფუნქციის პარამეტრებს წარმოადგენენ. გამოიყენეთ ფუნქცია ძირითად პროგრამაში და იპოვეთ 4 მთელ რიცხვს შორის უმცირესი;

ე) გამოითვლის სამკუთხედის ფართობს 3 გვერდის მიხედვით. გამოიყენეთ ფუნქცია ძირითად პროგრამაში და შეადარეთ 2 სამკუთხედის ფართობი;

ვ) დააბრუნებს 1-ს, თუ მისი პარამეტრი–სიმბოლო არის ხმოვანი ასო, წინააღმდეგ შემთხვევაში დააბრუნებს 0-ს. ძირითად პროგრამაში გამოიძახეთ ფუნქცია სხვადასხვა სიმბოლოებისათვის;

ზ) პარამეტრად მიიღებს სიმბოლოს, და თუ ეს სიმბოლო-ციფრია – დააბრუნებს მის კოდს, წინააღმდეგ შემთხვევაში კი დააბრუნებს –1-ს. გამოიყენეთ ფუნქცია ფაილში ჩაწერილი სიმბოლოების მიმდევრობისათვის.

8. გაარჩიეთ პროგრამა, რომელიც ხსნის შემდეგ ამოცანას:

უმცირესი მთელი რიცხვი, რომელიც მეტია $\log_2 N$ -ზე, წარმოადგენს ბიტების რაოდენობას მთელი N რიცხვის ორობით ჩანაწერში. დაწერეთ ფუნქცია, რომელიც გამოითვლის $\log_2 N$ -ზე მეტ უმცირეს მთელ რიცხვს. გამოიყენეთ ფუნქცია მთელი N რიცხვის ორობით წარმოდგენაში ბიტების რაოდენობის განსაზღვრისთვის.

```
#include <iostream>
using namespace std;
int my_log2(int N);
int main(){
  int N, degree;
  cin >> N;
  degree = my_log2(N);
  cout << degree << endl;
  system("PAUSE");
  return 0;
}
int my_log2(int N){
   int d;
   if(N < 0) N = -N;
   for(d =0; N > 0; d++, N /= 2);
   return d;
}
```

პროგრამის შესრულების შედეგია:

```
9
4
Press any key to continue . . .
```

დავალება:

ა) შეასრულეთ პროგრამა კომპიუტერზე სხვადასხვა მთელი N რიცხვისთვის;

ბ) ახსენით `my_log2` ფუნქციის ალგორითმი.

9. დაწერეთ ფუნქცია, რომელიც ითვლის:

ა) $1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{20}$ ჯამს; ბ) $1+\frac{1}{3}+\frac{1}{5}+\cdots+\frac{1}{n}$ ჯამს, სადაც კენტი n რიცხვი ფუნქციის პარამეტრია.

ლაბორატორიული მეცადინეობა № 7

ლაბორატორიული სამუშაოს აღწერა:

- ორ ცვლადს შორის უდიდესის დამდგენს ფუნქციის შექმნა
- ფუნქციის შექმნა, რომელიც დაადგენს, წარმოადგენს თუ არა სტრიქონი პალინდრომს

ლაბორატორიული სამუშაო:

ამოცანა 1: შექმენით ფუნქცია, რომელიც იპოვის 2 მთელი რიცხვიდან უდიდესს. ეს რიცხვები ფუნქციის პარამეტრებია. გამოიყენეთ ფუნქცია `main`-ში კლავიატურიდან შემოტანილი 2 მთელი რიცხვისთვის და დაბეჭდეთ მათ შორის უდიდესი.

პროგრამის სახე შემდეგია:

```
//////////////////////////////////////////
// პროგრამა: 2 მთელ რიცხვს შორის უდიდესის
//                პოვნა
//////////////////////////////////////////
#include <iostream>
using namespace std;
int max2Int(int , int );
int main(){
  int a, b;
  cout << "SemoitaneT 2 mTeli ricxvi : ";
  cin >> a >> b;
  int max = max2Int(a, b);
  cout << "udidesia " << max << endl;
  system("PAUSE");
  return 0;
}
int max2Int(int x, int y){
  return x > y ? x : y;
}
```

დავალება: შეასრულეთ პროგრამა სხვადასხვა `a` და `b` რიცხვებისთვის.

ამოცანა 2: შექმენით ფუნქცია, რომელიც იპოვის 2 სტრიქონიდან უდიდესს. ეს სტრიქონები ფუნქციის პარამეტრებია. გამოიყენეთ ფუნქცია `main`-ში კლავიატურიდან შემოტანილი 2 სტრიქონისთვის და დაბეჭდეთ მათ შორის უდიდესი.

პროგრამას აქვს შემდეგი სახე:

```
//////////////////////////////////////
// პროგრამა: 2 სტრიქონს შორის უდიდესის
//                პოვნა
//////////////////////////////////////
#include <iostream>
#include <string>
using namespace std;
string max2String(string , string );
int main(){
  string a, b;
  cout << "SemoitaneT 2 striqoni : ";
  cin >> a >> b;
  string max = max2String(a, b);
  cout << "udidesia " << max << endl;
  system("PAUSE");
  return 0;
}
string max2String(string x, string y){
```

```
  return x > y ? x : y;
}
```

დავალება: შეასრულეთ პროგრამა სხვადასხვა a და b სტრიქონებისთვის.

ამოცანა 3: დაწერეთ ფუნქცია, რომლის პარამეტრი არის სტრიქონი. ფუნქციამ უნდა დაადგინოს, წარმოადგენს თუ არა იგი პალინდრომს. სტრიქონი არის პალონდრომი, თუ ის ერთნაირად იკითხება როგორც მარჯვნიდან მარცხნივ, ისე მარცხნიდან მარჯვნივ. მაგალითად, ANNA – სტრიქონი-პალინდრომია. გამოიყენეთ ფუნქცია კლავიატურიდან შემოტანილი სტრიქონისთვის და დაბეჭდეთ შედეგი.

პროგრამის სახე შემდეგია:

```
//////////////////////////////////////////
// პროგრამა: სტრიქონსთვის დადგენა, არის თუ
//              არა იგი პალინდრომი
//////////////////////////////////////////
#include <iostream>
#include <string>
using namespace std;
bool Palindrom(string );
int main(){
  string S; bool p;
  cout << "Enter string\n";
  getline(cin, S);
  p = Palindrom(S);
  if( p ) cout << "Is Palindrom" << endl;
  else  cout << "Is't Palindrom" << endl;
  system("pause");
  return 0;
}
bool Palindrom(string K){
   int i, len;
   len = K.size();
   for(i=0; i<len/2; i++)
     if(K[i] != K[len-1-i])
        return false;
   return true;
}
```

დავალება:

ა) შეასრულეთ პროგრამა სხვადასხვა სტრიქონებისათვის, მაგალითად, ai ia, level, Ai Ia.

ბ) ახსენით Palindrom ფუნქციის ალგორითმი.

ამოცანა 4: ქვემოთ მოყვანილ პროგრამაში S სტრიქონის შებრუნებისთვის გამოიყენება C++ -ის სტანდარტული string კლასის ალგორითმი reverse:

```
//////////////////////////////////////////
// პროგრამა: სტრიქონის შებრუნება
//////////////////////////////////////////
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
int main(){
  string S ="Bjarne Stroustrup";
  reverse(S.begin(), S.end());
```

```
  cout << S << endl;
  //system("pause");
  return 0;
}
```

დავალება:

ა) გაეცანით კოდს, შეასრულეთ პროგრამა S სტრიქონის სხვადასხვა მნიშვნელობისთვის;

ბ) გამოიყენეთ ალგორითმი reverse იმის დასადგენად, წარმოადგენს თუ არა კლავიატურიდან შეტანილი S სტრიქონი პალინდრომს.

დამოუკიდებელი სამუშაო:

1. შექმენით ფუნქცია, რომელიც იპოვის 2 სიმბოლოდან უდიდესს. ეს სიმბოლოები ფუნქციის პარამეტრებია. გამოიყენეთ ფუნქცია main-ში კლავიატურიდან შემოტანილი 2 სიმბოლოსთვის და დაბეჭდეთ მათ შორის უდიდესი.

2. დაწერეთ ფუნქცია, რომელიც იპოვის 2 ნამდვილი რიცხვიდან უდიდესს. ეს რიცხვები ფუნქციის პარამეტრებია. გამოიყენეთ ფუნქცია main-ში კლავიატურიდან შემოტანილი 2 ნამდვილი რიცხვისთვის და დაბეჭდეთ მათ შორის უდიდესი.

3. დაწერეთ ფუნქცია, რომელიც გამოითვლის ნატურალური N რიცხვის ციფრთა ჯამს. გამოიძახეთ ფუნქცია main-ში კლავიატურიდან შეტანილი N რიცხვის ციფრთა ჯამის დასადგენად. მიღებული შედეგი დაბეჭდეთ.

გაარჩიეთ შესაბამისი პროგრამა:

```
#include <iostream>
using namespace std;
int DigitsSum(int );
int main(){
   int number, sum;
   cin>>number;
   sum = DigitsSum(number);
   cout << "Digits sum of " << number
        << " = " << sum << endl;
   system("pause");
   return 0;
}
int DigitsSum(int N){
   int s =0;
   while(N != 0){
     s += N%10;
     N /= 10;
   }
   return s;
}
```

დავალება:

ა) შეასრულეთ პროგრამა სხვადასხვა ნატურალური რიცხვისთვის.

ბ) ახსენით DigitsSum ფუნქციის ალგორითმი

გ) შეცვალეთ ფუნქციაში while(N != 0) სტრიქონი while(N) –ით, კვლავ შეასრულეთ პროგრამა და ახსენით შედეგი.

4*. დაწერეთ ფუნქცია, რომელიც დაადგენს, წარმოადგენს თუ არა ნატურალური m რიცხვი პალინდრომს. რიცხვი არის პალინდრომი, თუ ის ერთნაირად იკითხება როგორც მარცხნიდან მარჯვნივ ისე მარჯვნიდან მარცხნივ. მაგალითად, 12321 – პალინდრომია.

5.* დაწერეთ ფუნქცია, რომელიც დაადგენს არის თუ არა ნატურალური N რიცხვი ავტომორფული. რიცხვი არის ავტომორფული, თუ მისი კვადრატის ჩანაწერი ბოლოვდება ამავე რიცხვით. მაგალითად, 5, 6, 25 და ა.შ. – ავტომორფული რიცხვებია.

6*. დაწერეთ ფუნქცია, რომელიც დაადგენს არის თუ არა ნატურალური m რიცხვი სრულყოფილი. რიცხვი არის სრულყოფილი, თუ ის უდრის ყველა თავისი გამყოფის ჯამს, გარდა თავის თავისა. მაგალითად, რიცხვი 6 – სრულყოფილია, ვინაიდან 6=1+2+3.

7*. დაწერეთ ფუნქცია, რომელიც დაადგენს არის თუ არა ნატურალური N რიცხვი მარტივი.

8. ფიბონაჩის რიცხვები განისაზღვრება შემდეგი წესით: $F_0 = F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$, თუ $n \geq 2$. დაწერეთ ფუნქცია, რომელიც ჩაწერს ვექტორში პირველ K ფიბონაჩის რიცხვს და შემდეგ დაბეჭდავს ვექტორს.

გაარჩიეთ შესაბამისი C++– პროგრამა:

```
#include <iostream>
#include <vector>
using namespace std;
void Fibonacci_numbers(int );
int main(){
   int K;
   cout << "ShemoitaneT ricxvebis raodenoba : ";
   cin >> K;
   cout << "\nFibonachis pirveli " << K << " ricxvia\n\n";
   Fibonacci_numbers(K);
   system("pause");
   return 0;
}
void Fibonacci_numbers(int K){
  vector<int> F(K);
  F[0] = F[1] = 1;
  for(int i=2; i<K; i++)
     F[i] = F[i-1] + F[i-2];
  for(int i=0; i<K; i++)
     cout << F[i] << ' ';
  cout << endl;
}
```

დავალება:

ა) გაუშვით პროგრამა შესრულებაზე და შეიტანეთ ფიბონაჩის რიცხვების რაოდენობად 15. დარწმუნდით, რომ პროგრამამ დაბეჭდა:

```
Fibonachis pirveli 15 ricxvia
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
```

ბ) შეასრულეთ პროგრამა K –ს სხვადასხვა მნიშვნელობებისთვის.

9. დაწერეთ ფუნქცია, რომელიც:
   ა) იპოვის ფიბონაჩის მე-20 რიცხვს;
   ბ) იპოვის ფიბონაჩის N –ურ რიცხვს. N ფუნქციის პარამეტრია;
   გ) გამოითვლის ისეთი ფიბონაჩის რიცხვების ჯამს, რომლებიც არ აღემატება M –ს. M ფუნქციის პარამეტრია.

10. შექმენით ფუნქცია `void draw(int M);` რომელიც ერთ სტრიქონად გამოიტანს ეკრანზე M ცალ:
    ა) '*'-ს; ბ) '+'-ს; გ) '-'-ს; დ) სლეშს.

პრაქტიკული მეცადინეობა № 8

პრაქტიკული მეცადინეობის თემები:

- ფუნქციების გადატვირთვა
- ფუნქციები `insert` და `erase`
- ცვლადის მისამართის „აღება“, მისამართით ცვლადის მნიშვნელობის „აღდგენა“.

მასალა დამოუკიდებლად გაცნობისათვის:

1. გადატვირთული ფუნქციების ნიმუშები

№	ფუნქცია	ფუნქციის გამოძახება - main()-ის ფრაგმენტი

1	/*ფუნქცია ადგენს არის თუ არა რიცხვი დადებითი */
	bool IsPositive(int K){
	return K > 0;
	}
	bool IsPositive(double K){
	return K > 0;
	}
		int main(){
		int a =10, b =-30;
		double c =-2.5, d =0.0956;
		cout<<boolalpha
		<<a<<" is positive? "<<IsPositive(a)
		<<endl<<b<<" is positive? "<<IsPositive(b)
		<<endl<<c<<" is positive? "<<IsPositive(c)
		<<endl<<d<<" is positive? "<<IsPositive(d)
		<<endl;
		. . .
		}

2	/* ფუნქცია პოულობს რიცხვის მოდულს */
	int my_abs(int x){
	return x < 0 ? -x : x;
	}
	double my_abs(double x){
	return x < 0 ? -x : x;
	}
		int main(){
		int a =-17; double d =20;
		cout<<a<<" -is moduli aris "
		<<my_abs(a)<<endl
		<<d<<" -is moduli aris "
		<<my_abs(d)<<endl;
		. . .
		}

3	/* ფუნქცია პოულობს ორი რიცხვიდან უმცირესს */
	int myMin(int x, int y){
	return x < y ? x : y;
	}
	double myMin(double x,
	double y){
	return x < y ? x : y;
	}
		int main(){
		int a =123, b =39;
		double c =-1.75, d =0.48;
		cout<<a<<" da "<<b
		<<" -s Soris umciresia "
		<<myMin(a, b)<<endl
		<<c<<" da "<<d
		<<" -s Soris umciresia "
		<<myMin(c, d)<<endl;
		. . .
		}

4	/* ფუნქცია ითვლის რიცხვის კვადრატს */
	int square(int a){
	return a * a;
	}
	double square(double a){
	return a * a;
	}
		int main(){
		int x =11;
		double y =-0.7;
		cout<<x<<" -is kvadrati udris "
		<<square(x)<<endl
		<<y<<" -is kvadrati udris "
		<<square(y)<<endl;
		. . .
		}

5	/* ფუნქცია ითვლის ორი რიცხვის საშუალოს */
	double average(int n,int m){
	return (n + m)/2.;
	}
		int main(){
		int a =23, b =12;
		double c =5.5, d =2.25;
		cout<<a<<" da "<<b
		<<" risxvebis saSualo = "
		<<average(a, b)<<endl

```
double average(double n,
               double m){
   return (n + m)/2;
}
```

```
        <<c<<" da "<<d
        <<" risxvebis saSualo = "
        <<average(c, d)<<endl;
  . . .
}
```

2. ზოგიერთი სტანდარტული ფუნქციის გამოყენება ვექტორთან

№	ფუნქციის გამოძახება	ფუნქციის მუშაობის შედეგი

1	`int main(){`
	`  vector<int> v;`
	`  . . .`
	`//ვექტორის პირველ ელემენტად 7 –ის ჩასმა`
	`  v.insert(v.begin(),7);`
	`  . . .`
	`}`
		`//თავდაპირველი ვექტორი`
		`56 -37 302 47 0 300 -1 89`
		`//insert ფუნქციის მუშაობის შემდეგ`
		`7 56 -37 302 47 0 300 -1 89`

2	`int main(){`
	`  vector<int> V;`
	`  . . .`
	`/*ვექტორის მეოთხე (ინდექსით 3) ელემენტად 100–ის ჩასმა */`
	`  V.insert(V.begin()+3,100);`
	`  . . .`
	`}`
		`//თავდაპირველი ვექტორი`
		`56 -37 302 47 0 300 -1 89`
		`//insert ფუნქციის მუშაობის შემდეგ`
		`56 -37 302 100 47 0 300 -1 89`

3	`int main(){`
	`  vector<int> A;`
	`  . . .`
	`//ვექტორის შუა ელემენტად 200–ის ჩასმა`
	`A.insert(A.begin()+A.size()/2,200);`
	`  . . .`
	`}`
		`//თავდაპირველი ვექტორი`
		`56 -37 30 47 0 35 -1 89`
		`//insert ფუნქციის მუშაობის შემდეგ`
		`56 -37 30 47 200 0 35 -1 89`

4	`int main(){`
	`  vector<int> v;`
	`  . . .`
	`//ვექტორის პირველი ელემენტის წაშლა`
	`  v.erase(v.begin());`
	`  . . .`
	`}`
		`//თავდაპირველი ვექტორი`
		`56 -37 30 47 0 35 -1 89`
		`// erase ფუნქციის მუშაობის შემდეგ`
		`-37 30 47 0 35 -1 89`

5	`int main(){`
	`  vector<int> v;`
	`  . . .`
	`/*ვექტორის მეოთხე (ინდექსით 3) ელემენტის წაშლა */`
	`  v.erase(v.begin()+3);`
	`  . . .`
	`}`
		`//თავდაპირველი ვექტორი`
		`56 -37 30 47 0 35 -1 89`
		`// erase ფუნქციის მუშაობის შემდეგ`
		`56 -37 30 0 35 -1 89`

6	`int main(){`
	`  vector<int> v;`
	`  . . .`
	`//ვექტორის შუა ელემენტის წაშლა`
	`v.erase (v.begin()+v.size()/2);`
	`  . . .`
	`}`
		`//თავდაპირველი ვექტორი`
		`7 -14 30 15 -9 35 -1 56 0`
		`// erase ფუნქციის მუშაობის შემდეგ`
		`7 -14 30 15 35 -1 56 0`

საუდიტორიო სამუშაო:

1. გადატვირთეთ ფუნქცია Max, რომელიც დააბრუნებს რამდენიმე ცვლადს შორის უდიდესს. გამოიყენეთ ფუნქცია main-ში 2 მთელი რიცხვისთვის, 3 ნამდვილი რიცხვისთვის, 2 სტრიქონისთვის და 3 სიმბოლოსთვის და დაბეჭდეთ მათ შორის უდიდესი.

```
#include <iostream>
#include <string>
using namespace std;
int Max(int , int );
double Max(double , double , double );
string Max(string, string);
char Max(char , char , char );
int main(){
  int n =45, m =-7;
  double a =3.65, b =75.123, c = -3.143;
  char p ='F', q ='K', u = ',';
  string s1 ="C++ ena", s2 ="C ena";
  cout<<n<<" da "<<m<<" -s Soris udidesia: "
      << Max(n, m) << endl
      << a <<  ", " << b << " da "<< c <<" -s Soris udidesia: "
      << Max(a, b, c) << endl
      << p << ", " << q << " da "<< u << " -s Soris udidesia: "
      <<Max(p, q, u)<<endl
      << s1 << " da "<< s2 << " -s Soris udidesia: "
      << Max(s1, s2)<<endl;
  return 0;
}
int Max(int x, int y){
  return x > y ? x : y;
}
double Max(double x, double y, double z){
  double max = x;
  if(max < y)  max = y;
  if(max < z)  max = z;
  return max;
}
string Max(string x, string y){
  return x > y ? x : y;
}
char Max(char x, char y, char z){
  double max = x;
  if(max < y)  max = y;
  if(max < z)  max = z;
  return max;
}
```

პროგრამა დაბეჭდავს:

```
45 da -7 -s Soris udidesia: 45
3.65, 75.123 da -3.143 -s Soris udidesia:
75.123
F, K da , -s Soris udidesia: K
C++ ena da C ena -s Soris udidesia: C++ ena
Press any key to continue . . .
```

2. data.txt ფაილი შეიცავს მთელ რიცხვებს. ჩაწერეთ ისინი სათანადო სპეციფიკაციის ვექტორში და შემდეგ ჩასვით ვექტორის პირველ ელემენტად რიცხვი 49.

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;
void printVector(vector<int> );
```

```
int main(){
  vector<int> V;
  int number;
  ifstream fin("data.txt");
  while(fin>>number)
     V.push_back(number);
  printVector(V);
  V.insert(V.begin(),49);
  printVector(V);
  return 0;
}
void printVector(vector<int> x){
   for(int i=0; i<x.size(); i++)
       cout<<x[i]<<' ';
   cout<<endl;
}
```

პროგრამის შესრულების შედეგია:

data.txt ფაილი	გამოსატანი ეკრანი
56 -37 202 0 300 -1 89	56 -37 202 0 300 -1 89 49 56 -37 202 0 300 -1 89 Press any key to continue . . .

3. ქვემოთ მოყვანილი პროგრამები გვიბეჭდავენ სხვადასხვა ტიპის ცვლადების მისამართებს და ამ მისამართებზე განთავსებულ მნიშვნელობებს.

ა)

```
#include <iostream>
using namespace std;
int main(){
  int N =38;
  cout<<"Address of N (in hexodecimal) = "<<&N<<endl
      <<"Address of N (in decimal) = "<<(int)&N<<endl
      <<"Value of N = "<<N<<endl
      <<"Value of N = "<<*(&N)<<endl<<endl;
  int K =123;
  cout<<"Address of K = "<<(int)&K<<endl
      <<"Previous address = "<<(int)(&K - 1)<<endl
      <<"Next address = "<<(int)(&K + 1)<<endl
      <<"Next next address = "<<(int)(&K + 2)<<endl;
  cout<<"int tipi ikavebs "<<sizeof(int)<<" baits\n";
  cout<<"int tipi ikavebs "<<sizeof(N)<<" baits\n";
  system("PAUSE");
  return 0;
}
```

პროგრამა დაბეჭდავს:

```
Address of N (in hexodecimal) = 0012FF60
Address of N (in decimal) = 1245024
Value of N = 38
Value of N = 38

Address of K = 1245012
Previous address = 1245008
Next address = 1245016
Next next address = 1245020
int tipi ikavebs 4 baits
int tipi ikavebs 4 baits
Press any key to continue . . .
```

ბ)

```
#include <iostream>
using namespace std;
int main(){
  double D = 0.00748;
  cout<<"Address of D = "<<(int)&D<<endl
      <<"Value of D = "<<D<<endl
      <<"Value of D = "<<*(&D)<<endl;
  double Q =1.00987;
  cout<<"\nAddress of Q = "<<(int)&Q<<endl
      <<"Previous address = "<<(int)(&Q - 1)<<endl
      <<"Next address = "<<(int)(&Q + 1)<<endl
      <<"Next next address = "<<(int)(&Q + 2)<<endl;
  cout<<"double tipi ikavebs "<<sizeof(double)<<" baits\n";
  return 0;
}
```

პროგრამის შესრულების შედეგია:

```
Address of D = 2293616
Value of D = 0.00748
Value of D = 0.00748

Address of Q = 2293608
Previous address = 2293600
Next address = 2293616
Next next address = 2293624
double tipi ikavebs 8 baits
Press any key to continue . . .
```

გ)

```
#include <iostream>
using namespace std;
int main(){
  char C ='M';
  cout<<"Address of C = "<<(int)&C<<endl
      <<"Value of C = "<<C<<endl
      <<"Value of C = "<<*(&C)<<endl;
  char P ='J';
  cout<<"\nAddress of P = "<<(int)&P<<endl
      <<"Previous address = "<<(int)(&P - 1)<<endl
      <<"Next address = "<<(int)(&P + 1)<<endl
      <<"Next next address = "<<(int)(&P + 2)<<endl;
  cout<<"char tipi ikavebs "<<sizeof(C)<<" baits\n";
  return 0;
}
```

პროგრამის შესრულების შედეგია:

```
Address of C = 2293623
Value of C = M
Value of C = M

Address of P = 2293622
Previous address = 2293621
Next address = 2293623
Next address = 2293624
char tipi ikavebs 1 baits
Press any key to continue . . .
```

დამოუკიდებელი სამუშაო:

1. გადატვირთეთ ფუნქცია MIN, რომელიც დაადგენს 2 ცვლადიდან უმცირესს. გამოიყენეთ ფუნქცია main-ში 2 მთელი რიცხვისთვის, 2 ნამდვილი რიცხვისთვის, 2 სტრიქონისთვის და 2 სიმბოლოსთვის და დაბეჭდეთ შესაბამის ცვლადებს შორის უმცირესი.

2. გადატვირთეთ ფუნქცია min, რომელიც დაადგენს 3 ცვლადიდან უმცირესს. გამოიყენეთ ფუნქცია main-ში 3 მთელი რიცხვისთვის, 3 ნამდვილი რიცხვისთვის, 3 სტრიქონისთვის და 3 სიმბოლოსთვის და დაბეჭდეთ შესაბამის ცვლადებს შორის უმცირესი.

3. გადატვირთეთ ფუნქცია MAX, რომელიც დაადგენს 3 ცვლადიდან უდიდესს. გამოიყენეთ ფუნქცია main-ში 3 მთელი რიცხვისთვის, 3 ნამდვილი რიცხვისთვის, 3 სტრიქონისთვის და 3 სიმბოლოსთვის და დაბეჭდეთ შესაბამის ცვლადებს შორის უდიდესი.

4. გადატვირთეთ ფუნქცია neg, რომელიც დააბრუნებს რიცხვის მოპირდაპირე მნიშვნელობას. გამოიყენეთ ფუნქცია main-ში მთელი რიცხვისთვის, ნამდვილი რიცხვისთვის.

5. ყურადღებით გაარჩიეთ ამოცანა შემდეგი პირობით:
   გადატვირთეთ ფუნქცია Mult, ანუ შექმენით მისი 3 ვერსია:
   ა) ფუნქცია ბეჭდავს 2 ნამდვილი რიცხვის ნამრავლს;
   ბ) ფუნქცია ითვლის 3 მთელი რიცხვის ნამრავლს;
   გ) ფუნქცია ითვლის სტრიქონის 2-ჯერ გაზრდილ სიგრძეს.
   გამოიყენეთ ფუნქციები ძირითად პროგრამაში სათანადო მონაცემებისთვის.

```
#include <iostream>
#include <string>
using namespace std;
void Mult(double ,double );
int Mult(int ,int ,int );
int Mult(string );
int main(){
   int a, b, c;
   double f, t;
   string s;
   cout<<"Enter string:\n";
   getline(cin, s);
   cout<<"The doubled length of a string = "
       <<Mult(s)<<endl<<endl;
   cout<<"Enter 3 integers:\n";
   cin>>a>>b>>c;
   cout<<"Product of 3 integers = "
       <<Mult(a, b, c)<<endl;
   cout<<"\nEnter 2 reals:\n";
   cin>>f>>t;
   cout<<"Product of 2 reals = ";
   Mult(f, t);
   return 0;
}
void Mult(double a, double b){
   cout<<a*b<<endl;
}
int Mult(int a, int b, int c){
   return a*b*c;
}
int Mult(string k){
   return 2*k.size();
}
```

პროგრამის შესრულების შედეგია:

```
Enter string:
I'm string
The doubled length of a string = 20

Enter 3 integers:
12 5 3
```

```
Product of 3 integers = 180

Enter 2 reals:
1.2 4.0
Product of 2 reals = 4.8

Press any key to continue . . .
```

დავალება: შეასრულეთ პროგრამა კომპიუტერზე კლავიატურიდან შემავალი სხვადასხვა მონაცემებისთვის და დარწმუნდით მის სისწორეში. ***დაიმახსოვრეთ:*** C++ -ში შესაძლებელია ისეთი ფუნქციების გადატვირთვა, რომელთა ალგორითმები განსხვავდებიან.

6. Reals.txt ფაილი შეიცავს ნამდვილ რიცხვებს. ჩაწერეთ ისინი სათანადო სპეციფიკაციის ვექტორში და შემდეგ ჩასვით ვექტორის მეორე (ინდექსით 1) ელემენტად რიცხვი -0.72.

7. ამოცანის პირობაა: numbers.txt ფაილი შეიცავს მთელ რიცხვებს. ჩაწერეთ ისინი სათანადო სპეციფიკაციის ვექტორში და შემდეგ ჩასვით ვექტორის ბოლოსწინა ელემენტად რიცხვი 120.
გაარჩიეთ შესაბამისი C++ –ის პროგრამა:

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;
void printVector(vector<int> );
int main(){
  vector<int> V;
  int number;
  ifstream fin("data.txt");
  while(fin>>number)
     V.push_back(number);
  printVector(V);
  V.insert(V.end()-1,120);
  printVector(V);
  system("PAUSE");
  return 0;
}
void printVector(vector<int> x){
   for(int i=0; i<x.size(); i++)
       cout<<x[i]<<' ';
   cout<<endl;
}
```

პროგრამის შესრულების შედეგია:

numbers.txt ფაილი	გამოსატანი ეკრანი
7 -14 30 15 -9 35 -1 56 0	7 -14 30 15 -9 35 -1 56 0 7 -14 30 15 -9 35 -1 56 120 0 Press any key to continue . . .

8. ამოცანის პირობაა: sorted.txt ფაილი შეიცავს ზრდადობით დალაგებულ მთელ რიცხვებს. ჩაწერეთ ეს რიცხვები სათანადო სპეციფიკაციის ვექტორში და შემდეგ ჩასვით ვექტორში კლავიატურიდან შემოტანილი მთელი რიცხვი ისე, რომ დალაგება არ დაირღვეს.
გაარჩიეთ შესაბამისი C++ –ის პროგრამა:

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;
void printVector(vector<int> );
```

```
int main(){
  vector<int> v;
  int num;
  ifstream ifs("sorted.txt");
   while(ifs>>num)
     v.push_back(num);
  printVector(v);
  int ins;
  cout<<"Enter insert number : ";
  cin>>ins;
  if(ins > v[v.size()-1])v.push_back(ins);
  else
  {
    int i =0;
    while(ins > v[i]) i++;
    v.insert( v.begin()+ i, ins );
  }
  printVector(v);
  return 0;
}
void printVector(vector<int> x){
   for(int i=0; i<x.size(); i++)
       cout<<x[i]<<' ';
   cout<<endl;
}
```

პროგრამის შესრულების ერთ-ერთი შედეგია:

sorted.txt ფაილი	გამოსატანი ეკრანი
-7 -5 1 3 5 7 9	-7 -5 1 3 5 7 9 Enter insert number : -10 -10 -7 -5 1 3 5 7 9 Press any key to continue . . .

დავალება:

ა) შეასრულეთ პროგრამა კომპიუტერზე სხვადასხვა ჩასასმელი მთელი რიცხვისთვის და დარწმუნდით პროგრამის სისწორეში;

ბ) ახსენით ელემენტის ჩასმის ალგორითმი.

9. ამოცანის პირობაა: ვექტორში ნამდვილი რიცხვებია. წაშალეთ ვექტორის ბოლო ელემენტი. რიცხვები ვექტორში შეიტანეთ `reals.info` ფაილიდან.
გაარჩიეთ შესაბამისი C++ –ის პროგრამა:

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;
void printVector(vector<double> );
int main(){
  vector<double> D;
  double number;
  ifstream ifs("reals.info");
   while(ifs>>number)
     D.push_back(number);
  printVector(D);
  D.erase(D.end()-1);
  printVector(D);
  system("PAUSE");
  return 0;
}
```

```
void printVector(vector<double> x){
   for(int i=0; i<x.size(); i++)
       cout<<x[i]<<' ';
   cout<<endl;
}
```

პროგრამის შესრულების შედეგია:

reals.info ფაილი	გამოსატანი ეკრანი
0.05 -3.24 53.75 1 -0.94	0.05 -3.24 53.75 1 -0.94 0.05 -3.24 53.75 1 Press any key to continue . . .

დავალება:

ა) შეასრულეთ პროგრამა კომპიუტერზე და ახსენით შეტყობინება

```
D.erase(D.end()-1);
```

ბ) შეცვალეთ იგი შემდეგით:

```
D.pop_back();
```

და ისევ შეასრულეთ პროგრამა კომპიუტერზე. გააანალიზეთ მიღებული შედეგი.

10. შემოიტანეთ კლავიატურიდან 20 მთელი რიცხვი და ჩაწერეთ სათანადო სპეციფიკაციის ვექტორში. შემდეგ წაშალეთ ვექტორის რიგით მე-15 ელემენტი და დაბეჭდეთ ვექტორი.

11. ამოცანის პირობაა: reals.dat ფაილი შეიცავს ნამდვილ რიცხვებს. ჩაწერეთ ისინი სათანადო სპეციფიკაციის ვექტორში და შემდეგ წაშალეთ კლავიატურიდან შემოტანილი ნამდვილი რიცხვის ტოლი ვექტორის ელემენტი. თუ ასეთი ელემენტი ვექტორში არ არის, დაბეჭდეთ შესაბამისი გზავნილი.

გაარჩიეთ შესაბამისი C++ –ის პროგრამა:

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;
void printVector(vector<double> );
int main(){
  vector<double> v;
  double num;
  ifstream ifs("reals.dat");
   while(ifs>>num)
     v.push_back(num);
  printVector(v);
  double del;
  cout<<"Enter number to delete: ";
  cin>>del;
  int i;
  for(i=0; i<v.size(); i++)
     if(v[i] == del) break;
  if(i == v.size())
    cout<<"vector not contains this number\n";
  else{
    v.erase( v.begin()+ i );
    printVector(v);
  }
  return 0;
}
void printVector(vector<double> x){
   for(int i=0; i<x.size(); i++)
       cout<<x[i]<<' ';
```

```
    cout<<endl;
}
```

პროგრამის შესრულების ერთ-ერთი შედეგია:

reals.dat ფაილი	გამოსატანი ეკრანი
1.2 -0.03 2.5 0.056 12.45	1.2 -0.03 2.5 0.056 12.45 Enter deleted number : 1.2 -0.03 2.5 0.056 12.45 Press any key to continue . . .

დავალება:

ა) შეასრულეთ პროგრამა კომპიუტერზე სხვადასხვა წასაშლელი ნამდვილი რიცხვისთვის და დარწმუნდით პროგრამის სისწორეში;

ბ) ახსენით ელემენტის წაშლის ალგორითმი.

ლაბორატორიული მეცადინეობა № 8

ლაბორატორიული სამუშაოს აღწერა:

- ვექტორის გარკვეული თვისების ელემენტებს შორის უდიდესის და უმცირესის დადგენა

ლაბორატორიული სამუშაო:

ამოცანა 1: ვექტორში მთელი რიცხვებია, მათ შორის 3-ის ჯერადებიც. შექმენით ფუნქცია, რომელიც იპოვეთ ვექტორის 3-ის ჯერად ელემენტებს შორის უდიდესის ინდექსს. გამოიყენეთ ფუნქცია `main`-ში კლავიატურიდან შევსებული ვექტორისთვის და დაბეჭდეთ ნაპოვნი ინდექსი და შესაბამისი რიცხვი.

პროგრამის სახე შემდეგია:

```
//////////////////////////////////////////
// პროგრამა: ვექტორის 3-ის ჯერად ელემენტებს
//           შორის უდიდესის ინდექსის პოვნა
//////////////////////////////////////////
#include <iostream>
#include <vector>
using namespace std;
int maxIndex(vector<int> );
int main(){
  vector<int> A;
  int number;
  while(cin>>number)
     A.push_back(number);
  int index = maxIndex(A);
  cout<<index<<'\t'<<A[index]<<endl;
  return 0;
}
int maxIndex(vector<int> x){
  int i, index;
  for(i=0; i<x.size(); ++i)
    if(x[i]%3 == 0) break;
  index = i;
  for(i++; i<x.size(); ++i)
    if(x[i]%3 == 0 && x[i] > x[index])
       index = i;
  return index;
}
```

დავალება:

ა) შეასრულეთ პროგრამა რამდენჯერმე სხვადასხვა მთელ რიცხვთა მიმდევრობისთვის;
ბ) ახსენით `maxIndex` ფუნქციის ალგორითმი.

ამოცანა 2: ვექტორში ნამდვილი რიცხვებია, მათ შორის 200-ზე მეტი რიცხვებიც. შექმენით ფუნქცია, რომელიც იპოვეთ ვექტორის 200-ზე მეტ ელემენტებს შორის უმცირესის ინდექსს. გამოიყენეთ ფუნქცია `main`-ში კლავიატურიდან შევსებული ვექტორისთვის და დაბეჭდეთ ნაპოვნი ინდექსი და შესაბამისი რიცხვი.

პროგრამის სახე შემდეგია:

```
//////////////////////////////////////////
// პროგრამა: ვექტორის 200-ზე მეტ ელემენტებს
//           შორის უმცირესის ინდექსის პოვნა
//////////////////////////////////////////
#include <iostream>
```

```
#include <vector>
using namespace std;
bool property(double );
int minIndex(vector<double> );
int main(){
  vector<double> A;
  double number;
  while(cin>>number)
     A.push_back(number);
  int index = minIndex(A);
  cout<<index<<'\t'<<A[index]<<endl;
  return 0;
}
bool property(double d){
   return d > 200;
}
int minIndex(vector<double> x){
  int i, index;
  for(i=0; i<x.size(); ++i)
    if( property(x[i]) ) break;
  index = i;
  for(i++; i<x.size(); ++i)
    if(x[i] > 200 && x[i] < x[index])
       index = i;
  return index;
}
```

დავალება:

ა) შეასრულეთ პროგრამა რამდენჯერმე სხვადასხვა ნამდვილ რიცხვთა მიმდევრობისთვის;

ბ) ახსენით `minIndex` ფუნქციის ალგორითმი.

ამოცანა 3: `sentence.dat` ფაილი შეიცავს სტუდენტების სახელებს. ინფორმაცია ფაილიდან ჩაწერეთ სათანადო სპეციფიკაციის ვექტორში. დაწერეთ ფუნქცია, რომელის დაადგენს, არის თუ ვექტორში სახელი `Anna`. გამოიყენეთ ფუნქცია `main`-ში და დაბეჭდეთ შესაბამისი გზავნილი.

პროგრამის სახე შემდეგია:

```
//////////////////////////////////////////
// პროგრამა: სტრიქონების ვექტორში მოცემული
//                 სიტყვის ძებნა
//////////////////////////////////////////
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
using namespace std;
void printVector(vector<string> );
bool contains(vector<string> );
int main(){
  vector<string> S;
  string word;
  ifstream ifs("sentence.dat");
  while(ifs>>word)
     S.push_back(word);
  printVector(S);
  if ( contains(S) )
    cout<<"Name \"Anna\" is in file\n";
  else
    cout<<"File not contains name \"Anna\"\n";
```

```
  return 0;
}
void printVector(vector<string> x){
   for(int i=0; i<x.size(); i++)
       cout<<x[i]<<' ';
   cout<<endl;
}
bool contains(vector<string> x){
  for(int i=0; i<x.size(); ++i)
     if(x[i] == "Anna") return true;
  return false;
}
```

დავალება:

ა) შექმენით ფაილი `sentence.dat` და ჩაწერეთ მასში სახელები;
ბ) შეასრულეთ პროგრამა, როდესაც ფაილში სახელი `Anna` არის და როდესაც არ არის და დააკვირდით შედეგებს;
გ) ახსენით `contains` ფუნქციის ალგორითმი.

დამოუკიდებელი სამუშაო:

1. ვექტორში მთელი რიცხვებია, მათ შორის ლუწებიც. შექმენით ფუნქცია, რომელიც იპოვეთ ვექტორის ლუწ ელემენტებს შორის უდიდესის ინდექსს. გამოიყენეთ ფუნქცია `main`-ში კლავიატურიდან შევსებული ვექტორისთვის და დაბეჭდეთ ნაპოვნი ინდექსი და შესაბამისი რიცხვი.

2. ვექტორში მთელი რიცხვებია, მათ შორის 7-ის ჯერადებიც. შექმენით ფუნქცია, რომელიც იპოვეთ ვექტორის 7-ის ჯერად ელემენტებს შორის უმცირესის ინდექსს. გამოიყენეთ ფუნქცია `main`-ში კლავიატურიდან შევსებული ვექტორისთვის და დაბეჭდეთ ნაპოვნი ინდექსი და შესაბამისი რიცხვი.

3. ვექტორში მთელი რიცხვებია, მათ შორის კენტებიც. შექმენით ფუნქცია, რომელიც იპოვეთ ვექტორის კენტ ელემენტებს შორის უმცირესის ინდექსს. რიცხვები ვექტორში ჩაწერეთ `ints.dat` ფაილიდან. გამოიყენეთ ფუნქცია და დაბეჭდეთ ნაპოვნი ინდექსი და შესაბამისი რიცხვი.

4. ვექტორში ნამდვილი რიცხვებია, მათ შორის 25-ზე მეტი რიცხვებიც. შექმენით ფუნქცია, რომელიც იპოვეთ ვექტორის 25-ზე მეტ ელემენტებს შორის უმცირესის ინდექსს. რიცხვები ვექტორში ჩაწერეთ `data.in` ფაილიდან. გამოიყენეთ ფუნქცია და დაბეჭდეთ ნაპოვნი ინდექსი და შესაბამისი რიცხვი.

5. `words.txt` ფაილი შეიცავს სიტყვებს. ინფორმაცია ფაილიდან ჩაწერეთ სათანადო სპეციფიკაციის ვექტორში. დაწერეთ ფუნქცია, რომელის დაადგენს, არის თუ არა ვექტორში სიტყვა `student`. გამოიყენეთ ფუნქცია `main`-ში და დაბეჭდეთ შესაბამისი გზავნილი.

6. `words.in` ფაილი შეიცავს სიტყვებს. ინფორმაცია ფაილიდან ჩაწერეთ სათანადო სპეციფიკაციის ვექტორში. დაწერეთ ფუნქცია, რომელის წაშლის ვექტორიდან სიტყვას `war`. თუ ასეთი სიტყვა ვექტორში არ არის, დაბეჭდავს შესაბამის გზავნილს. გამოიყენეთ ფუნქცია `main`-ში და დაბეჭდეთ ვექტორი.

7. `Words.info` ფაილი შეიცავს ზრდადობით დალაგებულ სიტყვებს. ინფორმაცია ფაილიდან ჩაწერეთ სათანადო სპეციფიკაციის ვექტორში. დაწერეთ ფუნქცია, რომელის ჩასვამს ვექტორში სიტყვას `Peace` ისე, რომ სიტყვების დალაგება არ დაირღვეს. გამოიყენეთ ფუნქცია `main`-ში და დაბეჭდეთ ვექტორი.

8. ვექტორში მთელი რიცხვებია. შექმენით ფუნქცია, რომელიც იპოვეთ ვექტორის 3-ის ჯერად ელემენტებს შორის უდიდესის ინდექსს. თუ 3-ის ჯერადი რიცხვები ფაილში არ არის, ფუნქცია დააბრუნებს -1 -ს. გამოიყენეთ ფუნქცია main-ში კლავიატურიდან შევსებული ვექტორისთვის და დაბეჭდეთ ნაპოვნი ინდექსი და შესაბამისი რიცხვი, ან გზავნილი ფაილში 3-ის ჯერადი რიცხვების არარსებობის შესახებ.

გაარჩიეთ შათანადო C++ -პროგრამა:

```
#include <iostream>
#include <vector>
using namespace std;
int specialMax(vector<int> );
int main(){
  vector<int> A;
  int number;
  while(cin>>number)
     A.push_back(number);
  int index = specialMax(A);
  if(index != -1)
     cout<<index<<'\t'<<A[index]<<endl;
  else cout<<"VeqtorSi 3 -is jeradi ricxvebi"
             " ar aris\n";
  system("PAUSE");
  return 0;
}
int specialMax(vector<int> x){
  int i, index;
  for(i=0; i<x.size(); ++i)
    if(x[i]%3 == 0) break;
  if(i == x.size()) return -1;
  index = i;
  for(i++; i<x.size(); ++i)
    if(x[i]%3 == 0 && x[i] > x[index])
       index = i;
  return index;
}
```

9. ვექტორში ნამდვილი რიცხვებია. შექმენით ფუნქცია, რომელიც იპოვეთ ვექტორის 180-ზე ნაკლებ ელემენტებს შორის უდიდესის ინდექსს. თუ 180-ზე ნაკლები რიცხვები ფაილში არ არის, ფუნქცია დააბრუნებს -1 -ს. გამოიყენეთ ფუნქცია main-ში კლავიატურიდან შევსებული ვექტორისთვის და დაბეჭდეთ ნაპოვნი ინდექსი და შესაბამისი რიცხვი, ან სათანადო გზავნილი.

პრაქტიკული მეცადინეობა № 9

პრაქტიკული მეცადინეობის თემები:

- პოინტერი. მოქმედებები პოინტერზე
- ფუნქციის პარამეტრი/არგუმენტი - პოინტერი
- ფუნქცია `srand`

მასალა დამოუკიდებლად გაცნობისათვის:

1. პოინტერის გამოყენების მარტივი მაგალითები

№	პროგრამის ფრაგმენტი	შედეგი

1	`int n =179;`
	`// განაცხადი მთელი ტიპის p პოინტერზე`
	`int *p;  // იგივეა, რაც int* p;`
	`p = &n;`
	`cout<<*p<<endl;`
		`179`

2	`double a =12.5407;`
	`double *k;`
	`k = &a;`
	`cout<<*k <<endl;`
		`12.5407`

3	`char sym = '?';`
	`char* ptr = &sym;`
	`cout<<*ptr<<endl;`
		`?`

4	`int n = -48;`
	`/* განაცხადი მთელი ტიპის t პოინტერზე და მისი ინიციალიზება n ცვლადის მისამართით */`
	`int* t = &n;`
	`/* სინტაქსურად აგრეთვე სწორია`
	`int *t = &n; */`
	`cout<<*t<<endl;`
		`-48`

5	`float y = 4.25;`
	`float* p = &y;`
	`*p = 0.78;`
	`cout<<y<<endl;`
		`0.78`

6	`char s = 65;`
	`char* ptr = &s;`
	`cout<<*ptr<<' '<<(int)*ptr<<endl;`
		`A 65`

7	`int c = 100;`
	`int *k;`
	`k = &c;`
	`cout<<(int)k<<' '<<k`
	`<<' '<<*k<<endl;`
		`/* c -ს მისამართი დაიბეჭდა როგორც მთელი რიცხვი და როგორც თექვსმეტობითი რიცხვი, შემდეგ დაიბეჭდა c -ს მნიშვნელობა*/`
		`1310560  0013FF60  100`

8	`int b = 50;`
	`int* q = &b;`
	`cout<<*q<<' '<<*q+5<<endl;`
		`50 55`

9	`float t = 35.9;`
	`float *m;`
	`m = &t;`
	`cout<<(int)m<<endl<<*m-1<<endl;`
		`2293620`
		`34.9`

10	`char ch = 'E';`
	`char* k = &ch;`
	`cout<<(int)k<<' '<<*k<<endl;`
		`2293625 E`

11	`double d = 0.75E7;`
	`double* q = &d;`
	`cout<<*q/25<<endl<<(int)q<<endl;`
		300000
		2293616

12	`int k = -37;`
	`int* t = &k;`
	`cout<<*t+1<<' '<<int(t)`
	`<<' '<<int(t+1)<<endl;`
		/* დაიბეჭდა k-ს 1-თით გაზრდილი მნიშვნელობა, k-ს მისამართი და მისი მომდევნო მისამართი */
		-36 2293620 2293624

2. ფუნქცია `srand(x)` განსაზღვრავს `rand()`-ის მიერ გენერირებულ პირველ ფსევდოშემთხვევით რიცხვს. `x` არგუმენტი არის **`unsigned int`** ტიპის. ხშირად ეს ფუნქცია გამოიყენება იმისთვის, რომ სხვადასხვა ჯერზე გაშვების დროს პროგრამამ შეძლოს ფსევდოშემთხვევითი რიცხვების **სხვადასხვა** მიმდევრობის მიღება. ამისათვის ამ მიმდევრობის გენერაციამდე უნდა გამოვიძახოთ `srand(x)` ფუნქცია პარამეტრის სხვადასხვა მნიშვნელობით. ხშირად პარამეტრად გამოიყენება სისტემის კალენდარული დროის მიმდინარე მნიშვნელობა `time(NULL)`. `srand` ფუნქციაზე განაცხადი მოცემულია `stdlib.h` ფაილში.

№	პროგრამის ფრაგმენტი	შედეგი

1	`srand(0);`
	`cout<<rand()<<endl;`
	`cout<<rand()<<endl;`
	`cout<<rand()<<endl;`
		38
		7719
		21238

2	`srand(1);`
	`cout<<rand()<<endl;`
	`cout<<rand()<<endl;`
	`cout<<rand()<<endl;`
		41
		18467
		6334

3	`srand(157);`
	`cout<<rand()<<endl;`
	`cout<<rand()<<endl;`
	`cout<<rand()<<endl;`
		551
		24051
		7865

4	`srand(time(NULL));`
	`cout<<rand()<<endl;`
	`cout<<rand()<<endl;`
	`cout<<rand()<<endl;`
		30956
		13975
		5745

5	`// იგივე ფრაგმენტი შესრულდა 10 წუთის შემდეგ`
	`srand(time(NULL));`
	`cout<<rand()<<endl;`
	`cout<<rand()<<endl;`
	`cout<<rand()<<endl;`
		31263
		8517
		13809

სააუდიტორიო სამუშაო:

1. რა შეცდომას შეიცავს შემდეგი ფრაგმენტი?

```
int a, b =25;
int *p = &a;
*p = &b;
cout<<*p;
```

ფრაგმენტის მესამე სტრიქონი

```
*p = &b;
```

გამოიწვევს კომპილაციის შეცდომას: `cannot convert from 'int *' to 'int'`
გამოსახულება `*p = &b` კორექტულია მხოლოდ პოინტერის ინიციალიზების მომენტში, ანუ თუ დავწერთ

```
int *p = &b;
```

სხვა შემთხვევაში *p აღნიშნავს მეხსიერებაში p ცვლადში ჩაწერილ მისამართზე მოთავსებულ რიცხვს, ჩვენ შემთხვევაში 25 –ს.
ამიტომ, ინიციალიზების დროს სტილისტურად გამართლებულია ვარსკვლავის მოთავსება int –ის გვერდით, ასე

```
int* p = &b;
```

თუმცა, სინტაქსურად ასევე სწორია ჩანაწერი int *p = &b;

2. double *x, y, *z; განაცხადში რა ტიპისაა x, y და z ცვლადები?

x და z წარმოადგენენ double ტიპის პოინტერებს, ხოლო y – double ტიპის რიცხვს. ასეთ შემთხვევებში სტილისტურად გამართლებულია * –ის მიწერა ცვლადის სახელთან, რაც x –ის და z –ის ირიბ მნიშვნელობებზე მიუთითებს, ანუ double ტიპის იმ რიცხვებზე, რომელთა მისამართებიც ჩაიწერება x –ის და z –ის მეხსიერებაში.

სწორია თუ არა შემდეგი მინიჭებები?

y = 0.09;	სწორია, y –ის მეხსიერებაში ჩაიწერება ნამდვილი რიცხვი
*x = &y;	შეცდომაა, უნდა იყოს x = &y;
z = y;	შეცდომაა, უნდა იყოს z = &y;
z = x;	სწორია, პოინტერების მინიჭება ნებადართულია.

3. დაწერეთ ფუნქცია, რომელიც გაუცვლის მნიშვნელობებს 2 მთელ რიცხვს. გამოიყენეთ ფუნქცია ძირითად პროგრამაში და შედეგი დაბეჭდეთ.

```
#include <iostream>
using namespace std;
void swap (int *, int *);
int main(){
  int a, b;
  cout<<"ShemoitaneT 2 mTeli ricxvi :  ";
  cin>>a>>b;
  cout<<"swap funqciis shesrulebamde :\n";
  cout<<"a = "<<a<<' '<<"b = "<<b<<endl;
  swap(&a, &b);
  cout<<"swap funqciis shesrulebis shemdeg :\n";
  cout<<"a = "<<a<<' '<<"b = "<<b<<endl;
  return 0;
}
void swap (int * x, int * y){
   int dam;
   dam = *x;
   *x = *y;
   *y = dam;
}
```

პროგრამის შესრულების შედეგია:

```
ShemoitaneT 2 mTeli ricxvi :  5 7
swap funqciis shesrulebamde :
a = 5 b = 7
swap funqciis shesrulebis shemdeg :
a = 7 b = 5
Press any key to continue . . .
```

4. ა) დაწერეთ ფუნქცია, რომელიც გაუცვლის მნიშვნელობებს 2 ნამდვილ რიცხვს. გამოიყენეთ ფუნქცია ძირითად პროგრამაში და დაალაგეთ ზრდადობით 3 მოცემული ნამდვილი რიცხვი. შედეგი დაბეჭდეთ.

```
#include <iostream>
```

```
using namespace std;
void swap (double* , double* );
int main(){
    double a, b, c;
    cout<<"ShemoitaneT 3 namdvili ricxvi :\n";
    cin>>a>>b>>c;
    if(a>b) swap(&a, &b);
    if(b>c) swap(&b, &c);
    if(a>b) swap(&a, &b);
    cout<<"ZrdadobiT dalagebuli 3 ricxvi :\n";
    cout<<a<<' '<<b<<' '<<c<<endl;
    return 0;
}
void swap (double* x, double* y){
   double t;
   t = *x; *x = *y; *y = t;
}
```

პროგრამის შესრულების შედეგია:

```
ShemoitaneT 3 namdvili ricxvi :
10.5 -3.7 22.25
ZrdadobiT dalagebuli 3 ricxvi :
-3.7 10.5 22.25
Press any key to continue . . .
```

ბ) მოიყვანეთ ისეთი სამი რიცხვის მაგალითი, რომ პროგრამის მესამე

```
if(a>b) swap(&a, &b);
```

შეტყობინების შესრულება აუცილებელი იყოს რიცხვების დალაგებისათვის.

დამოუკიდებელი სამუშაო:

1. შეასრულეთ და გააანალიზეთ შემდეგი პროგრამები

ა)
```
#include <iostream>
using namespace std;
int main(){
  int x = 15, y = 34;
    int *k, *p;
    k = &y;
    p = k;
    x = *p;
    cout<< x <<endl;
   return 0;
}
```

ბ)
```
#include <iostream>
using namespace std;
int main(){
  double x = 7.86, y = 3.37, *k, *p;
   k = &x;
   p = &y;
   y = *k;
   cout<<*p<<endl;
  return 0;
}
```

2. დაწერეთ ფუნქცია, რომელიც გაუცვლის მნიშვნელობებს 2 სიმბოლოს. გამოიყენეთ ფუნქცია ძირითად პროგრამაში და შედეგი დაბეჭდეთ.

3. დაწერეთ ფუნქცია, რომელიც გაუცვლის მნიშვნელობებს 2 სტრიქონს. გამოიყენეთ ფუნქცია ძირითად პროგრამაში და დაალაგეთ ზრდადობით 3 მოცემული სტრიქონი. შედეგი დაბეჭდეთ.
4. გადატვირთეთ ფუნქცია swap, რომელიც გაუცვლის მნიშვნელობებს ერთი და იგივე ტიპის 2 ცვლადს.
5. დაწერეთ ფუნქცია, რომელიც:
   ა) გამოითვლის ორი მთელი რიცხვის ჯამსა და ნამრავლს. ჯამი უნდა იყოს ფუნქციის დასაბრუნებელი მნიშვნელობა, ნამრავლი კი მან უნდა დააბრუნოს ცვლადი პარამეტრის საშუალებით. გამოიძახეთ ფუნქცია ძირითად პროგრამაში და დაბეჭდეთ მიღებული შედეგები;

   გაარჩიეთ შესაბამისი C++ -პროგრამა:

```
#include <iostream>
using namespace std;
int calculation(int, int, int* );
int main(){
  int a, b, sum, product;
  cout<<"SemoitaneT 2 mTeli ricxvi : ";
  cin>>a>>b;
  sum = calculation(a, b, &product);
  cout<<"Ricxvebis jami = "<<sum<<endl
      <<"Namravli = "<<product<<endl;
  return 0;
}
int calculation(int x, int y, int* p){
   *p = x * y;
   return x + y;
}
```

   პროგრამის შესრულების შედეგია:

```
SemoitaneT 2 mTeli ricxvi : 12 10
Ricxvebis jami = 22
Namravli = 120
Press any key to continue . . .
```

   ბ) გამოითვლის სამკუთხედის ფართობსა და პერიმეტრს სამი გვერდის მიხედვით (სამკუთხედის გვერდები ფუნქციის პარამეტრებია, პერიმეტრი – ფუნქციის ცვლადი პარამეტრი). გამოიყენეთ ფუნქცია ძირითად პროგრამაში და მოცემული სამკუთხედისათვის დაბეჭდეთ მისი ფართობი და პერიმეტრი;

   გ) გამოითვლის სამკუთხედის ფართობს გვერდისა და მასზე დაშვებული სიმაღლის მიხედვით. გამოიყენეთ ფუნქცია ძირითად პროგრამაში და მოცემული სამკუთხედისათვის დაბეჭდეთ მისი ფართობი.
6. ვექტორი შეიცავს N მთელ შემთხვევით რიცხვს. დაწერეთ პროგრამა, რომელიც იპოვის და დაბეჭდავს ვექტორის:
   ა) უდიდესი ელემენტის მნიშვნელობას და მის ინდექსს;

   გაარჩიეთ შესაბამისი C++ -პროგრამა:

```
#include <iostream>
#include <vector>
#include <ctime>
using namespace std;
void printVector(vector<int> );
int maxIndex(vector<int> );
int main(){
  const int N =15;
  vector<int> X;
```

```
  srand( time(NULL) );
  for(int count =1; count <=N; count++)
     X.push_back( rand() );
  printVector(X);
  int index = maxIndex(X);
  cout<<"Udidesi elementi = "<<X[index]<<endl
      <<"VeqtorSi misi indeqsia "<<index<<endl;
  return 0;
}
void printVector(vector<int> x){
   for(int i=0; i<x.size(); i++)
       cout<<x[i]<<' ';
   cout<<endl;
}
int maxIndex(vector<int> x){
   int ind = 0;
   for(int i=1; i<x.size(); i++)
      if(x[i] > x[ind]) ind = i;
   return ind;
}
```

პროგრამის შესრულების შედეგია:

```
30161  13246  26412  9137  20062  16277  8810  30701
22795 15790 12544 1996 29652 5732 4843
Udidesi elementi = 30701
VeqtorSi misi indeqsia 7
Press any key to continue . . .
```

ბ) ელემენტების ჯამს;

გ) ელემენტების საშუალო არითმეტიკულს.

7. ვექტორი შეიცავს N მთელ რიცხვს. დაწერეთ პროგრამა, რომელიც იპოვის და დაბეჭდავს მოცემული b რიცხვის ტოლ ვექტორის პირველივე ელემენტის ინდექსს, ან, თუ ასეთი ელემენტი არ აღმოჩნდა, შესაბამის გზავნილს. ვექტორი შეავსეთ შემთხვევითი რიცხვებით, b შემოიტანეთ კლავიატურიდან.

გაარჩიეთ შესაბამისი C++ -პროგრამა:

```
#include <iostream>
#include <vector>
#include <ctime>
using namespace std;
void printVector(vector<int> );
int get_index(vector<int> , int);
int main(){
  const int N =10;
  vector<int> A;
  srand( time(NULL) );
  for(int count =1; count <=N; count++)
     A.push_back( rand() );
  printVector(A);
  int b;
  cout<<"b = ";
  cin>>b;
  int index = get_index(A, b);
  if(index == -1)
    cout<<b<<"-s toli ricxvi veqtorSi ar aris\n";
  else
    cout<<b<<"-s toli ricxvis indeqsia "
        <<index<<endl;
```

```
  return 0;
}
void printVector(vector<int> x){
   for(int i=0; i<x.size(); i++)
       cout<<x[i]<<' ';
   cout<<endl;
}
int get_index(vector<int> x, int b){
   int ind = -1;
   for(int i=0; i<x.size(); i++)
      if(x[i] == b) return i;
   return ind;
}
```

დავალება:

შეასრულეთ პროგრამა `b` –ს სხვადასხვა მნიშვნელობისთვის და დააკვირდით შედეგს.

8. `words.in` ფაილი შეიცავს ჰარის სიმბოლოთი გამოყოფილ სიტყვებს. დაწერეთ ფუნქცია, რომელიც დაადგენს, არის თუ არა ფაილში სიტყვა `exam`. თუ ეს სიტყვა ფაილში არის, ფუნქციამ უნდა დააბრუნოს მისი რიგითი ნომერი, თუ კი არა – დააბრუნოს `-1`. გამოიყენეთ ფუნქცია პროგრამაში და დაბეჭდეთ სათანადო გზავნილი.

ლაბორატორიული მეცადინეობა № 9

ლაბორატორიული სამუშაოს აღწერა:

- პოინტერის გამოყენება ცვლადებთან
- პოინტერის გამოყენება ფუნქციის პარამეტრად

ლაბორატორიული სამუშაო:

ამოცანა 1: შეასრულეთ და გააანალიზეთ შემდეგი პროგრამა:

```
//////////////////////////////////////////
//  პროგრამა: პოინტერის გამოყენება ცვლადებთან
//////////////////////////////////////////
#include <iostream>
using namespace std;
int main(){
   int a =9, *p =&a;
   cout<<"a = "<<a<<endl;
   cout<<"a-s misamarti = "<< &a <<endl
       <<"p-s mnishvneloba = "<< p <<endl;
   cout<<"p misamartze mnishvneloba = "
       << *p <<endl;
   return 0;
}
```

ამოცანა 2: დაწერეთ ფუნქცია, რომელიც დაადგენს და დააბრუნებს მართკუთხედის ფართობსა და პერიმეტრს. მართკუთხედის გვერდები ნამდვილი რიცხვებია, რომელიც შეგვაქვს კლავიატურიდან. გამოთვლილი ფართობი დააბრუნეთ ფუნქციიდან **return** ოპერატორის, პერიმეტრი კი – პარამეტრის გამოყენებით.

შესაბამის C++ -პროგრამას აქვს სახე:

```
//////////////////////////////////////////////////////
//  პროგრამა: პოინტერის გამოყენება ფუნქციის პარამეტრად
//////////////////////////////////////////////////////
#include <iostream>
using namespace std;
double Rectangle(double, double, double* );
int main(){
  double a, b, area, perimeter;
  puts("SemoitaneT marTkuTxedis gverdebi :\n");
  cin>>a>>b;
  area = Rectangle(a, b, &perimeter);
  cout<<"FarTobi = "<<area
      <<"\nPerimetri = "<<perimeter<<endl;
   return 0;
}
double Rectangle(double x, double y, double* per){
    double area;
    area = x * y;
    *per = 2 * (x + y);
    return area;
}
```

დავალება:

ა). გაუშვით პროგრამა შესრულებაზე და შეიტანეთ გვერდების სიდიდეები 2.5 და 3.0. დარწმუნდით, რომ პროგრამამ დაბეჭდა

```
FarTobi = 7.5
Perimetri = 11
```

ბ). შეასრულეთ პროგრამა მართკუთხედის გვერდების სხვა მნიშვნელობებისთვის და დარწმუნდით პასუხის სისწორეში.

ამოცანა 3: ვექტორი შეიცავს სიტყვებს. დაწერეთ ფუნქცია, რომელიც იპოვის ყველაზე გრძელ სიტყვას და მის სიგრძეს. გამოიყენეთ ფუნქცია `main()`–ში და დაბეჭდეთ შედეგი. ვექტორში სიტყვები ჩაწერეთ `words.txt` ფაილიდან.

შესაბამის C++ -პროგრამას აქვს სახე:

```
//////////////////////////////////////////////////
//  პროგრამა: ფაილიდან სიტყვების წაკითხვა და ყველაზე
//            გრძელი სიტყვის და მისი სიგრძის პოვნა
//////////////////////////////////////////////////
#include <iostream>
#include <vector>
#include <fstream>
#include <string>
using namespace std;
string get_word(vector<string>, int* );
int main(){
  vector<string> v;
  string w;
  ifstream fin("words.txt");
  while(fin>>w)
    v.push_back(w);
  int wordLength;
  string k = get_word(v, &wordLength);
  cout<<"Yvelaze grZeli sityvaa '"
      <<k<<"' sigrZiT "<<wordLength<<endl;
   return 0;
}
string get_word(vector<string> x, int* len){
  string max; int l = 0;
  for(int i=0; i<x.size(); i++)
       if(x[i].length() > l){
          l = x[i].length();
          max = x[i];
       }
  *len = l;
  return max;
}
```

დავალება:

ა). შექმენით `words.txt` ფაილი, ჩაწერეთ მასში ჰარით გამოყოფილი სიტყვები და დაამატეთ ფაილი პროექტში. შეასრულეთ პროგრამა. მიიღეთ სწორი შედეგი?

ბ). ახსენით `get_word` ფუნქციის ალგორითმი;

გ). ახსენით `*len = l;` შეტყობინების არსი.

დამოუკიდებელი სამუშაო:

1. `input.txt` ფაილში წერია მთელი რიცხვები. შექმენით ფუნქცია, რომელიც შეიტანს ამ რიცხვებს ვექტორში და დააბრუნებს პირველივე დადებითის ინდექსს. თუ დადებითი რიცხვი მათ შორის არ არის, დააბრუნებს `-1`-ს. გამოიყენეთ ფუნქცია პროგრამაში და დაბეჭდეთ სათანადო გზავნილი.

გაარჩიეთ შესაბამისი C++ -პროგრამა:

```
//////////////////////////////////////////////////
// პროგრამა: ფაილიდან რიცხვების წაკითხვა და პირველივე
//           დადებითი რიცხვის ინდექსის პოვნა
//////////////////////////////////////////////////
```

```
#include <iostream>
#include <vector>
#include <fstream>
using namespace std;
int get_index();
int main(){
   int dad_ind = get_index();
   if(dad_ind == -1)
     cout<<"DadebiTi ricxvebi failSi ar iyo\n";
   else
     cout<<"Pirvelive dadebiTis indeqsia "<<dad_ind<<endl;
   return 0;
}
int get_index(){
  vector<int> v;
  int a, ind = -1;
  ifstream fin("input.txt");
  while(fin>>a)
    v.push_back(a);
  for(int i=0; i<v.size(); i++)
    if(v[i] > 0) return i;
  return ind;
}
```

დავალება:

ა). `input.txt` ფაილში ჩაწერეთ რამდენიმე მთელი რიცხვი ისე, რომ მათ შორის იყოს ერთი მაინც დადებითი, შეასრულეთ პროგრამა და გააანალიზეთ შედეგი.

ბ). `input.txt` ფაილში ჩაწერეთ რამდენიმე მთელი რიცხვი ისე, რომ მათ შორის არც ერთი დადებითი არ იყოს, შეასრულეთ პროგრამა და გააანალიზეთ შედეგი.

2. `data.in` ფაილში ნამდვილი რიცხვებია. დაწერეთ ფუნქცია, რომელიც წაიკითხავს რიცხვებს ფაილიდან და დააბრუნებს პირველივე უარყოფითი რიცხვის რიგით ნომერს. თუ უარყოფითი რიცხვი მათ შორის არ არის, დააბრუნებს `-1`-ს. გამოიყენეთ ფუნქცია პროგრამაში და დაბეჭდეთ სათანადო გზავნილი.

შესაბამისი C++ -პროგრამის სახეა:

```
//////////////////////////////////////////////////////
//  პროგრამა: ფაილიდან რიცხვების წაკითხვა და პირველივე
//            უარყოფითი რიცხვის რიგითი ნომრის პოვნა
//////////////////////////////////////////////////////
#include <iostream>
#include <fstream>
using namespace std;
int get_rn();
int main(){
   int negative_rn = get_rn();
   if(negative_rn == -1)
     cout<<"UaryofiTi ricxvebi failSi ar iyo\n";
   else
     cout<<"Pirvelive uaryofiTis rigiTi nomeria "
         <<negative_rn<<endl;
   return 0;
}
int get_rn(){
   double a;
   int count = 0;
   ifstream ifs("input.txt");
   while ( ifs>>a ){
```

```
      count++;
      if(a < 0)
        return count;
     }
   return -1;
}
```

დავალება: შეასრულეთ და გააანალიზეთ მოცემული პროგრამა. ახსენით `get_rn` ფუნქციის ალგორითმი.

3. მე-9 პრაქტიკული მეცადინეობის ყველა დავალებისთვის დაწერეთ, გამართეთ და შეასრულეთ შესაბამისი პროგრამები.

პრაქტიკული მეცადინეობა № 10

პრაქტიკული მეცადინეობის თემები:

- მითითება (reference). მოქმედებები მითითებაზე
- ფუნქციის პარამეტრი/არგუმენტი - მითითება

მასალა დამოუკიდებლად გაცნობისათვის:

1. მითითების გამოყენების მარტივი მაგალითები

№	პროგრამის ფრაგმენტი	შედეგი

1	`int x =-15;`
	`// განაცხადი მთელი ტიპის y მითითებაზე`
	`int& y = x;`
	`/* y მთელი რიცხვია, რომელსაც იგივე`
	`მისამართი აქვს, რაც x -ს */`
	`cout<<"x = "<<x<<endl`
	`<<"y = "<<y<<endl;`
		`x = -15`
		`y = -15`

2	`int a = 35;`
	`int& b =a;`
	`cout<<"a-s misamarTia "<<&a<<endl`
	`<<"b-s misamarTia "<<&b<<endl;`
	`cout<<"a = "<<a<<' '<<"b = "<<b<<endl;`
		`a-s misamarTia`
		`0012FF60`
		`b-s misamarTia`
		`0012FF60`
		`a = 35 b = 35`

3	`char ch = 'M';`
	`char& p = ch;`
	`cout<<"p = "<<p<<endl;`
	`++p;`
	`cout<<"p = "<<p<<' '<<"ch = "<<ch<<endl;`
		`p = M`
		`p= N ch= N`

4	`double x =0.067;`
	`double& y =x;`
	`x += 2;`
	`cout<<"x = "<<x<<' '<<"y = "<<y<<endl;`
	`cout<<(int)&x<<' '<<(int)&y<<endl;`
		`x = 2.067 y = 2.067`
		`1245020 1245020`

5	`string s ="TSU";`
	`string& q =s;`
	`cout<<"s : "<<s<<' '<<"q : "<<q<<endl;`
	`cout<<(int)&q<<' '<<(int)&s<<endl;`
		`s : TSU q : TSU`
		`1244980 1244980`

6	`float y = 4.25;`
	`float& p = y;`
	`p = 0.78;`
	`--y;`
	`cout<<"y = "<<y<<' '<<"p = "<<p<<endl;`
		`y = -0.22 p = -0.22`

7	`char s = 97;`
	`char& k = s;`
	`cout<<"k : "<<k<<" code = "<<(int)k<<endl;`
		`k : a code = 97`

8	`string s ="C++";`
	`string& k =s;`
	`// k სტრიქონის შებრუნება`
	`reverse(k.begin(), k.end());`
	`cout<<"s : "<<s<<endl`
	`<<"k : "<<k<<endl;`
		`s : ++C`
		`k : ++C`

9	`string s ="amocana";`
	`string& k =s;`
	`// s სტრიქონის დასაწყისში g ასოს ჩასმა`
	`s.insert(0,"g");`
	`cout<<"s : "<<s<<endl`
	`<<"k : "<<k<<endl;`
		`s : gamocana`
		`k : gamocana`

10	double t =0.067;
	double& u =t;
	u *= 2;
	t++;
	cout<<"t = "<<t<<' '<<"u = "<<u<<endl;
	cout<<(int)&t<<' '<<(int)&u<<endl;
		t = 1.134 u = 1.134
		1245020 1245020

სააუდიტორიო სამუშაო:

1. რა შეცდომას შეიცავს შემდეგი ფრაგმენტი?

```
int a = 25;
int& p;
p = &a;
cout<<p;
```

ფრაგმენტის მეორე სტრიქონი

```
int& p;
```

გამოიწვევს კომპილაციის შეცდომას

```
'p' : references must be initialized
```

რაც ნიშნავს, რომ მითითებაზე განაცხადის დროს **აუცილებელია** მითითების ინიციალიზება უკვე არსებული მნიშვნელობით, ჩვენ შემთხვევაში, მთელი a რიცხვის სიდიდით.

ფრაგმენტის მესამე სტრიქონი

```
p = &a;
```

გამოიწვევს კომპილაციის შეცდომას

```
cannot convert from 'int *' to 'int'
```

ეს ნიშნავს, რომ მესამე სტრიქონში ჩვენ ვცდილობთ მთელ p რიცხვს მივანიჭოთ a –ს მისამართი.

სწორი ფრაგმენტი ასე გამოიყურება:

```
int a = 25;
int& p = a;
cout<<p;
```

2. double x, y; double & z = x ; განაცხადებში რა ტიპისაა x, y და z ცვლადები?
x, y და z წარმოადგენენ double ტიპის რიცხვებს, ამასთან z და x ნამდვილი რიცხვები ერთ მისამართზე არიან განთავსებული. ე.ი. z არის x რიცხვის სხვა სახელი (ფსევდონიმი)

სწორია თუ არა შემდეგი მინიჭებები?

```
y = 0.09;    სწორია, y –ის მეხსიერებაში ჩაიწერება ნამდვილი რისხვი 0.09
x = y;       სწორია, x –ის მეხსიერებაში ჩაიწერება ნამდვილი რისხვი 0.09
z = &y;      შეცდომაა, უნდა იყოს  z = y;
z = x + y;   სწორია, x, y და z – ნამდვილი რიცხვებია.
```

3. დაწერეთ ფუნქცია, რომელიც გაუცვლის მნიშვნელობებს 2 სტრიქონს. გამოიყენეთ ფუნქცია ძირითად პროგრამაში კლავიატურიდან შემოტანილი ორი სტრიქონისთვის და შედეგი დაბეჭდეთ.

```
#include <iostream>
#include <string>
using namespace std;
void swap(string & , string &);
```

```
int main(){
  string a, b;
  cout<<"ShemoitaneT 2 striqoni :  ";
  cin>>a>>b;
  cout<<"swap funqciis shesrulebamde :\n";
  cout<<"a = "<<a<<' '<<"b = "<<b<<endl;
  swap(a, b);
  cout<<"swap funqciis shesrulebis shemdeg :\n";
  cout<<"a = "<<a<<' '<<"b = "<<b<<endl;
  return 0;
}
void swap (string& x, string& y){
   string dam = x;
   x = y;
   y = dam;
}
```

პროგრამის შესრულების შედეგია:

```
ShemoitaneT 2 striqoni :  ananasi banani
swap funqciis shesrulebamde :
a = ananasi b = banani
swap funqciis shesrulebis shemdeg :
a = banani b = ananasi
Press any key to continue . . .
```

4. დაწერეთ ფუნქცია, რომელიც მთელ ვექტორში დაითვლის პირველი p ელემენტის ჯამს. ვექტორში რიცხვები ჩაწერეთ ints.txt ფაილიდან. ძირითად პროგრამაში გამოიძახეთ ფუნქცია პირველი:

ა) 10 რიცხვის ჯამის დასადგენად;

ბ) 15 რიცხვის ჯამის დასათვლელად;

გ) კლავიატურიდან შემოტანილი რაოდენობის რიცხვის ჯამის დასათვლელად.

რიცხვების ვექტორში ჩასაწერად და ვექტორის ელემენტების დასაბეჭდად ასევე შექმენით ფუნქციები.

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;
void fillVector(vector<int>& );
void printVector(vector<int> );
int Sum(vector<int> , int =10);
int main(){
   vector<int> A;
   fillVector(A);
   cout<<"Vectori :\n";
   printVector(A);
   cout<<"Pirveli 10 ricxvis jami tolia : "
       <<Sum(A)<<endl;
   cout<<"Pirveli 15 ricxvis jami tolia : "
       <<Sum(A, 15)<<endl;
   cout<<"Ramdeni ricxvis jami vipovoT ? ";
   int quantity;
   cin>>quantity;
   cout<<"Pirveli "<<quantity<<" ricxvis jami tolia : "
       <<Sum(A, quantity)<<endl;
   return 0;
}
```

```
void fillVector(vector<int>& a){
   int number;
   ifstream fin("ints.txt");
   while( fin>>number )
      a.push_back(number);
}
void printVector(vector<int> a){
   for(int i=0; i<a.size(); i++)
       cout<<a[i]<<' ';
   cout<<endl;
}
int Sum(vector<int> a, int p){
   int s = 0;
   for(int k=0; k<p; k++)
      s +=a[k];
   return s;
}
```

პროგრამის შესრულების შედეგია:

```
Vectori :
5 -7 100 3 78 123 -124 0 90 -10 -100 120 9 -21 0 4 -5 65 43 20
Pirveli 10 ricxvis jami tolia : 258
Pirveli 15 ricxvis jami tolia : 266
Ramdeni ricxvis jami vipovoT ? 5
Pirveli 5 ricxvis jami tolia : 179
Press any key to continue . . .
```

დამოუკიდებელი სამუშაო:

1. შეასრულეთ და გააანალიზეთ შემდეგი პროგრამები

ა)

```
#include <iostream>
using namespace std;
int main(){
   int j, k = 121;
   int &i = j;
   j = 10;
   cout<<j<<" "<<i<<endl;
   i = k;
   cout<<j<<" "<<i<<endl;
   i++;
   cout<<j<<" "<<i<<endl;
   return 0;
}
```

ბ)

```
#include <iostream>
using namespace std;
int main(){
   double a, b = 10.05;
   double &c = b;
   a = c;
   cout<<c<<" "<<a<<endl;
   c = -3.75;
   cout<<c<<" "<<a<<endl;
   b -= 0.5;
   cout<<c<<" "<<b<<endl;
   return 0;
}
```

2. დაწერეთ ფუნქცია, რომელიც გაუცვლის მნიშვნელობებს 2 სიმბოლოს. სიმბოლოები ფუნქციის ცვლადი პარამეტრები – მითითებებია. გამოიყენეთ ფუნქცია ძირითად პროგრამაში და შედეგი დაბეჭდეთ.
3. დაწერეთ ფუნქცია, რომელიც გაუცვლის მნიშვნელობებს 2 ნამდვილ რიცხვს. გამოიყენეთ ფუნქციის ცვლადი პარამეტრები – მითითებები. გამოიძახეთ ფუნქცია ძირითად პროგრამაში და შედეგი დაბეჭდეთ.

   გაარჩიეთ შესაბამილი C++-პროგრამა:
4. დაწერეთ ფუნქცია, რომელიც გაუცვლის მნიშვნელობებს 2 მთელ რიცხვს. გამოიყენეთ ფუნქციის ცვლადი პარამეტრები – მითითებები. გამოიძახეთ ფუნქცია ძირითად პროგრამაში და დაალაგეთ კლებადობით 3 მოცემული მთელი რიცხვი. შედეგი დაბეჭდეთ.

   გაარჩიეთ შესაბამილი C++-პროგრამა:

```
#include <iostream>
using namespace std;
void swap (int & , int & );
int main(){
    int a, b, c;
    cout<<"ShemoitaneT 3 mTeli ricxvi :\n";
    cin>>a>>b>>c;
    if(a<b) swap(a, b);
    if(b<c) swap(b, c);
    if(a<b) swap(a, b);
    cout<<"KlebadobiT dalagebuli 3 ricxvi :\n";
    cout<<a<<' '<<b<<' '<<c<<endl;
    return 0;
}
void swap (int& x, int& y){
   int temp;
   temp = x;
   x = y;
   y = temp;
}
```

პროგრამის შესრულების შედეგია:

```
ShemoitaneT 3 mTeli ricxvi :
5 7 9
KlebadobiT dalagebuli 3 ricxvi :
9 7 5
Press any key to continue . . .
```

5. დაწერეთ ფუნქცია, რომელიც გამოთვლის წრის ფართობსა და წრეწირის სიგრძეს. წრის რადიუსი ნამდვილი რიცხვია და ფუნქციის პარამეტრს წარმოადგენს. გამოთვლილი ფართობი დააბრუნეთ ფუნქციიდან **return** ოპერატორის, წრეწირის სიგრძე კი – პარამეტრის გამოყენებით. გამოიძახეთ ფუნქცია ძირითად პროგრამაში კონკრეტული წრისთვის, რომლის რადიუსი შეგვაქვს კლავიატურიდან.

   გაარჩიეთ შესაბამილი C++-პროგრამა:

```
#include <iostream>
using namespace std;
const double pi =3.14159;
double Circle(double, double & );
int main(){
   double R;
   double area,length;
   cout<<"Enter radius : ";
   cin>>R;
```

```
    area = Circle(R, length);
    cout<<"Area = "<<area<<endl;
    cout<<"Length = "<<length<<endl;
    return 0;
}
double Circle(double r, double & len){
    len = 2*pi*r;
    return  pi*r*r;
}
```

პროგრამის შესრულების შედეგია:

```
Enter radius : 5
Area = 78.5397
Length = 31.4159
Press any key to continue . . .
```

5. დაწერეთ ფუნქცია, რომელიც:
   ა) გამოითვლის მართკუთხედის ფართობს და პერიმეტრს. პერიმეტრი უნდა იყოს ფუნქციის დასაბრუნებელი მნიშვნელობა, ფართობი კი ფუნქციამ უნდა დააბრუნოს ცვლადი პარამეტრის საშუალებით. გამოიძახეთ ფუნქცია ძირითად პროგრამაში და დაბეჭდეთ მიღებული შედეგები;
   ბ) გამოითვლის მართკუთხა სამკუთხედის ფართობსა და პერიმეტრს კათეტების მიხედვით (სამკუთხედის კათეტები ფუნქციის პარამეტრებია, პერიმეტრი – ფუნქციის ცვლადი პარამეტრი). გამოიყენეთ ფუნქცია ძირითად პროგრამაში და მოცემული მართკუთხა სამკუთხედისათვის დაბეჭდეთ მისი ფართობი და პერიმეტრი;
6. ვექტორი შეიცავს N მთელ შემთხვევით რიცხვს [-20,80] შუალედიდან. დაწერეთ პროგრამა, რომელიც იპოვის და დაბეჭდავს ვექტორის:
   ა) უმცირესი ელემენტის მნიშვნელობას და მის ინდექსს;
   ბ) ელემენტების ჯამის მეხუთედს;
   გ) ელემენტების ნამრავლს;
   დ) კენტი ელემენტების საშუალო არითმეტიკულს:

   გაარჩიეთ შესაბამისი C++ -პროგრამა:

```
#include <iostream>
#include <vector>
#include <ctime>
using namespace std;
const int N =20;
void fillVector(vector<int> & );
void printVector(vector<int> );
bool isOdd(int );
double averageOdds(vector<int> );
int main(){
  vector<int> V;
  fillVector(V);
  printVector(V);
  double average = averageOdds(V);
  cout<<"Kentebis saSualo ariTmetikuli = "
      <<average<<endl;
  return 0;
}
void fillVector(vector<int> & x){
   srand( time(NULL) );
   for(int count =1; count <=N; count++)
      x.push_back( rand()%101 -20 );
}
```

```
void printVector(vector<int> x){
   for(int i=0; i<x.size(); i++)
       cout<<x[i]<<' ';
   cout<<endl;
}
bool isOdd(int k){
   return k%2 != 0;
}
double averageOdds(vector<int> x){
   double sum =0; int count =0;
   for(int i=0; i<x.size(); i++)
      if( isOdd(x[i]) ){
         sum += x[i];
         ++count;
      }
   return sum/count;
}
```

პროგრამის შესრულების შედეგია:

```
49 22 26 3 47 -19 40 22 5 -4 16 -3 79 68 19 78 25 41 44 -16
Kentebis saSualo ariTmetikuli = 24.6
Press any key to continue . . .
```

7. ვექტორში ზრდადობით დალაგებული მთელი რიცხვებია. დაწერეთ ფუნქცია პროტოტიპით `int BinarySearch(vector<int> a, int c);` რომელიც `a` ვექტორში მოძებნის `c` –ს ტოლ რიცხვს. ფუნქცია დააბრუნებს მის ინდექსს ან –1 -ს, თუ ასეთი რიცხვი ვექტორში არ არის. გამოიყენეთ ორობითი ძიების ალგორითმი.

   პროგრამაში ისარგებლეთ `BinarySearch` ფუნქციით ვექტორისთვის, რომელიც შევსებულია `sorted_numbers.txt` ფაილიდან. საძიებელი რიცხვი შემოიტანეთ კლავიატურიდან.

გაარჩიეთ შესაბამისი C++ -პროგრამა:

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;
void fillVector(vector<int> & );
void printVector(vector<int> );
int BinarySearch(vector<int> ,int );
int main(){
  vector<int> sorted;
  fillVector(sorted);
  printVector(sorted);
  int check;
  cout<<"\nEnter checked number : ";
  cin>>check;
  int index = BinarySearch(sorted, check);
  if(index == -1)
     cout<<"\nVeqtor does not contain "
         <<check<<endl;
  else cout<<"\nVeqtor contains "<<check
           <<", index is "<<index<<endl;
  return 0;
}
void fillVector(vector<int> & a){
   int number;
   ifstream ifs("sorted_numbers.txt");
   while( ifs>>number )
```

```
      a.push_back(number);
}
int BinarySearch(vector<int> a, int c){
   int l =0, r =a.size()-1, m;
   while( l <= r ){
     m = (l+r)/2;
     if(c == a[m]) return m;
     if(c > a[m]) l =m+1;
     else r =m-1;
   }
   return -1;
}
void printVector(vector<int> x){
   cout<<"Vector : \n\n";
   for(int i=0; i<x.size(); i++)
       cout<<x[i]<<' ';
   cout<<endl;
}
```

პროგრამის შესრულების შედეგია:

sorted_numbers.txt ფაილი	გამოსატანი ეკრანი
1 3 5 7 9 13 24 55 67	Vector : 1 3 5 7 9 13 24 55 67 Enter checked number : 55 Veqtor contains 55, index is 7 Press any key to continue . . .

8. ვექტორში N ნამდვილი განსხვავებული რიცხვია. დაწერეთ ფუნქცია, რომელიც დაადგენს და დააბრუნებს ვექტორის უმცირესი და უდიდესი ელემენტების ინდექსებს. ძირითად პროგრამაში შეავსეთ ვექტორი კლავიატურიდან, გამოიძახეთ ფუნქცია და დაბეჭდეთ ვექტორის უმცირესი და უდიდესი ელემენტების ჯამი.

გაარჩიეთ შესაბამისი C++ -პროგრამა:

```
#include <iostream>
#include <vector>
using namespace std;
int Indexes(vector<double> , int& );
void fillVector(vector<double>& );
int main(){
  vector<double> D;
  fillVector(D);
  int max_ind, min_ind;
  max_ind = Indexes(D, min_ind);
  double sum = D[max_ind] + D[min_ind];
  cout<<D[max_ind]<<" + "<<D[min_ind]
      <<" = "<<sum<<endl;
  return 0;
}
int Indexes(vector<double> x, int& min_i){
   int max_i;
   max_i = min_i =0;
   for(int i=1; i<x.size(); i++)
      if(x[i] > x[max_i]) max_i = i;
      else
        if( x[i] < x[min_i] ) min_i = i;
   return max_i;
}
```

```
void fillVector(vector<double>& x){
  double real;
  cout<<"SemoitaneT namdvili ricxvebi, "
        "bolos - simbolo\n";
  while( cin>>real )
     x.push_back(real);
}
```

პროგრამა დაბეჭდავს:

```
SemoitaneT namdvili ricxvebi, bolos - simbolo
1.2  0  75.6  -65.2  47.9  -34.75  100.95 *
100.95 + -65.2 = 35.75
Press any key to continue . . .
```

ლაბორატორიული სამუშაოს აღწერა:

- მითითების გამოყენება ცვლადებთან
- მითითების გამოყენება ფუნქციის პარამეტრად

ლაბორატორიული სამუშაო:

ამოცანა 1: შეასრულეთ და გააანალიზეთ შემდეგი პროგრამა:

```
//////////////////////////////////////////////
//  ავტორი:
//  პროგრამა: მითითების გამოყენება ცვლადებთან
//////////////////////////////////////////////
#include <iostream>
using namespace std;
int main(){
   int a =9;
   int& p =a;
   cout<<"a = "<<a<<endl;
   cout<<"a-s misamarti = "<< &a <<endl
       <<"p-s mnishvneloba = "<< p <<endl
       <<"p-s misamarti = "<< &p <<endl;
   a+=5;
   cout<<"a = "<< a <<endl
       <<"p = "<< p <<endl;
   p-=10;
   cout<<"a = "<< a <<endl
       <<"p = "<< p <<endl;
   return 0;
}
```

ამოცანა 2: დაწერეთ ფუნქცია, რომელიც დაითვლის კვადრატის ფართობსა და დიაგონალის სიგრძეს. კვადრატის გვერდი ნამდვილი რიცხვია, რომელიც შეგვაქვს კლავიატურიდან. კვადრატის ფართობი დააბრუნეთ ფუნქციიდან **return** ოპერატორის, დიაგონალის სიგრძე კი – პარამეტრი-მითითების გამოყენებით.

შესაბამის C++ -პროგრამას აქვს სახე:

```
//////////////////////////////////////////////////////
// პროგრამა: მითითების გამოყენება ფუნქციის პარამეტრად
//////////////////////////////////////////////////////
#include <iostream>
using namespace std;
double Square(double, double & );
int main(){
   double G;
   double area,diagonal;
   cout<<"Enter side : ";
   cin>>G;
   area = Square(G, diagonal);
   cout<<"Area = "<<area<<endl;
   cout<<"Diagonal = "<<diagonal<<endl;
   return 0;
}
double Square (double s, double& d){
   d = s * sqrt(2.0);
   return  s * s;
}
```

დავალება:

ა). გაუშვით პროგრამა შესრულებაზე, შეიტანეთ გვერდის სიდიდე 2.0 და დარწმუნდით, რომ პროგრამა დაბეჭდავს

```
Area = 4
Diagonal = 2.82843
```

ბ). შეასრულეთ პროგრამა კვადრატის გვერდის სხვა მნიშვნელობებისთვის და დარწმუნდით პასუხების სისწორეში.

ამოცანა 3: ჩაწერეთ ვექტორში 150 შემთხვევითი რიცხვი [100, 200] შუალედიდან, შემდეგ დაბეჭდეთ ვექტორის ელემენტები 7 სვეტად. ვექტორის შევსება და ბეჭდვა გააფორმეთ ფუნქციების სახით.

შესაბამის C++ -პროგრამას აქვს სახე:

```
//////////////////////////////////////////////////////
// ავტორი:
// პროგრამა: ვექტორის ელემენტების ბეჭდვა ცხრილის სახით
//////////////////////////////////////////////////////
#include <iostream>
#include <vector>
#include <iomanip>
#include <ctime>
using namespace std;
const int N =150;
void fillVector(vector<int>& );
void printVector(vector<int> );
int main(){
   vector<int> A;
   fillVector(A);
   printVector(A);
   return 0;
}
void fillVector(vector<int>& x){
   srand( time(NULL) );
   for(int count =1; count <=N; count++)
      x.push_back( rand()%101+100 );
}
void printVector(vector<int> x){
   for(int i=0; i<x.size(); i++){
      cout<<setw(5)<<x[i];
       if( (i+1)%7 == 0)cout<<endl;
   }
   cout<<endl;
}
```

დავალება:

ა). შეასრულეთ პროგრამა რამდენიმეჯერ, ახსენით `setw` მანიპულატორის დანიშნულება;

ბ). ახსენით

```
if( (i+1)%7 == 0)cout<<endl;
```

შეტყობინების მოქმედება;

გ). გადაწერეთ ფუნქცია ისე, რომ ვექტორის ელემენტები დაიბეჭდის: 4 სვეტად; 5 სვეტად შებრუნებული რიგით.

დამოუკიდებელი სამუშაო:

1. ვექტორში ჩაწერილია ზრდადობით დალაგებული სახელები. დაწერეთ ფუნქცია, რომელიც ჩაამატებს ვექტორში კლავიატურიდან შემოტანილ სახელს ისე, რომ სახელების დალაგება არ დაირღვეს. გამოიყენეთ შექმნილი ფუნქცია და დაბეჭდეთ ვექტორი. ვექტორში სახელები ჩაწერეთ `names.txt` ფაილიდან. **შენიშვნა:** ფუნქცია დაწერეთ `sort` ალგორითმის გამოყენების გარეშე.

გაარჩიეთ შესაბამისი C++ -პროგრამა:

```
//////////////////////////////////////////////////
// პროგრამა: ფაილიდან სტრიქონების ჩაწერა ვექტორში,
//           სტრიქონის ჩასმა  ვექტორში სათანადო ადგილას
//////////////////////////////////////////////////
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
using namespace std;
void fillVector(vector<string>& );
void printVector(vector<string> );
void insertName(vector<string>& , string );
int main(){
   vector<string> N;
   fillVector(N);
   printVector(N);
   cout<<"Enter name to insert : ";
   string name;
   cin>>name;
   insertName(N, name);
   printVector(N);
   return 0;
}
void printVector(vector<string> x){
   for(int i=0; i<x.size(); i++)
       cout<<x[i]<<' ';
   cout<<endl;
}
void fillVector(vector<string>& x){
   string name;
   ifstream fin("names.txt");
   while( fin>>name )
     x.push_back(name);
}
void insertName(vector<string>& x, string n){
   if( n > x[x.size()-1] ) x.push_back(n);
   else {
     int i =0;
     while( n > x[i] )i++;
     x.insert(x.begin()+i,n);
   }
}
```

დავალება:

ა). შეასრულეთ პროგრამა კლავიატურიდან შემოტანილი სხვადასხვა სახელებისთვის. ახსენით ფუნქციის ალგორითმი.

ბ). შეცვალეთ პროგრამის `insertName` ფუნქცია გამარტივებული ვერსიით: გამოიყენეთ `sort` ალგორითმი. კვლავ შეასრულეთ პროგრამა კლავიატურიდან შემოტანილი სხვადასხვა სახელებისთვის.

2. `A` და `B` ვექტორებში კლებადობით დალაგებული მთელი რიცხვებია. დაწერეთ ფუნქცია, რომელიც `A` და `B` ვექტორების საფუძველზე შექმნის ასევე კლებადობით დალაგებულ `C` ვექტორს. გამოიყენეთ ფუნქცია და დაბეჭდეთ `C` ვექტორი `C.txt` ფაილში.. `A` და `B` ვექტორებში რიცხვები შეიტანეთ `A.txt` და `B.txt` ფაილებიდან შესაბამისად. ვექტორის შევსება და ბეჭდვა გააფორმეთ ფუნქციების სახით. ამოცანა ამოხსენით ორი გზით: 1) `sort` ალგორითმის გამოყენების გარეშე; 2) `sort` ალგორითმის გამოყენებით.

1) გაარჩიეთ შესაბამისი C++ -პროგრამა:

```
//////////////////////////////////////////////////
// პროგრამა: დალაგებული მასივების შერწყმა დალაგების
//             შენარჩუნებით
//////////////////////////////////////////////////
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;
void fillVector(vector<int>& ,string );
void printVector(vector<int> );
void merge(vector<int> ,vector<int> ,vector<int>& );
int main(){
  vector<int> A, B, C;
  fillVector(A,"A.txt");
  fillVector(B,"B.txt");
  merge(A, B, C);
  printVector(C);
  return 0;
}
void printVector(vector<int> x){
   ofstream fout("C.txt");
   for(int i=0; i<x.size(); i++)
       fout<<x[i]<<' ';
}
void fillVector(vector<int>& x, string fname){
   ifstream fin(fname);
   int n;
   while( fin >> n ){
     x.push_back(n);
   }
}
void merge(vector<int> a,vector<int> b,vector<int>& c){
   int i, j, k;
   i =j =k =0;
   while( i <a.size() && j < b.size() )
      if( a[i] > b[j] ) c.push_back(a[i++]);
      else  c.push_back(b[j++]);
   if(i == a.size())
      for(j; j < b.size(); j++) c.push_back( b[j] );
   else  for(i; i < a.size(); i++) c.push_back( a[i] );
}
```

დავალება:

ა). შექმენით `A.txt` და `B.txt` ფაილები და ჩაწერეთ მათში კლებადობით დალაგებული მთელი რიცხვები. შეასრულეთ პროგრამა კომპიუტერზე და დარწმუნდით, რომ მიღებულ ფაილში რიცხვები დალაგებულია კლებადობით.

ბ). ახსენით `merge` ფუნქციის ალგორითმი.

2) **მითითება:** `X` ვექტორში მთელი რიცხვების კლებადობით დალაგებას უზრუნველყოფს `sort` ალგორითმი შემდეგი სახით: `sort(X.rbegin(), X.rend());`

ან `sort(X.begin(), X.end(), greater<int>());`

3. მე-10 პრაქტიკული მეცადინეობის ყველა დავალებისთვის დაწერეთ, გამართეთ და შეასრულეთ შესაბამისი პროგრამები.

პრაქტიკული მეცადინეობა № 11

პრაქტიკული მეცადინეობის თემები:

- ვექტორის ელემენტი - ვექტორი
- ორგანზომილებიანი ვექტორის გადაცემა ფუნქციაში

მასალა დამოუკიდებლად გაცნობისათვის:

1. ორგანზომილებიან ვექტორზე განაცხადი, მისთვის მეხსიერების გამოყოფა, ვექტორის ინიციალიზება და ელემენტების ბეჭდვა

№	პროგრამის ფრაგმენტი	შედეგი

1	/* განაცხადი ორგანზომილებიან Matrix ვექტორზე, რომელიც შეიცავს 3 ელემენტს – Row ვექტორს, თითოეულ Row ვექტორში 4 მთელი რიცხვია */
	vector<int> Row(4);
	vector<vector <int> > Matrix(3, Row);
		მეხსიერებაში გამოიყოფა უწყვეტი უბანი 12 (3 * 4) მთელი რიცხვისთვის

2	/* ორგანზომილებიან Matrix ვექტორში რიცხვების ჩაწერა */
	vector<int> Row(4);
	vector<vector <int> > Matrix(3, Row);
	for(int i=0; i<3; i++)
	for (int j=0; j<4; j++)
	Matrix[i][j] = i+j;
		განაწილებულ მეხსიერებაში ჩაიწერა რიცხვები
		0 1 2 3 1 2 3 4 2 3 4 5

3	/* ორგანზომილებიან Matrix ვექტორში ჩაწერილი რიცხვების ბეჭდვა */
	for(int i=0; i<3; i++)
	{
	for (int j=0; j<4; j++)
	cout <<Matrix[i][j]<<'\t';
	cout<<endl;
	}
		0 1 2 3
		1 2 3 4
		2 3 4 5

4	/* ორგანზომილებიან ნამდვილ რიცხვთა 2 x 3 ვექტორში კლავიატურიდან შემოტანილი რიცხვების ჩაწერა და შემდეგ ბეჭდვა */
	vector<double> row(3);
	vector<vector<double>> matrix(2, row);
	for(int i=0; i<2; ++i)
	for(int j=0; j<3; ++j)
	cin>>matrix[i][j];
	for(int i=0; i<2; ++i) {
	for(int j=0; j<3; ++j)
	cout<<setw(9)<<matrix[i][j];
	cout<<endl;
	}
	cout<<endl;
		// შეტანა
		23.5 -456.75 0.0987
		1.065 0.0 -64.25
		// გამოტანა
		23.5 -456.75 0.0987
		1.065 0 -64.25

5	/* ორგანზომილებიან 5 x 9 ვექტორში ჩავწეროთ სიმბოლოები symbols.dat ფაილიდან და შემდეგ დავბეჭდოთ ცხრილის სახით table.out ფაილში */ vector<char> row(9);	// symbols.dat ფაილი asdfashgdfhgfxcbvnxmc vbiupoiupoiuoiujk;lkj cxvbvcbvcxbfgshfjayiqywo

	vector<vector<char>> matrix(5, row);
	ifstream ifs("symbols.dat");
	for(int i=0; i<5; ++i)
	  for(int j=0; j<9; ++j)
	  ifs>>matrix[i][j];
	ofstream ofs("table.out");
	for(int i=0; i<5; ++i) {
	  for(int j=0; j<9; ++j)
	  ofs<<matrix[i][j];
	  ofs<<endl;
	}
		// table.out ფაილი
		
		asdfashgd
		fhgfxcbvn
		xmcvbiupo
		iupoiuoiu
		jk;lkjcxv

6	/* ორგანზომილებიან მთელ რიცხვთა MxN ვექტორში ჩავწეროთ შემთხვევითი რიცხვები [-50,100] შუალედიდან, შემდეგ ვიპოვოთ მათი ჯამი და დავბეჭდოთ */
	const int M = 5, N = 4;
	vector<int> row(N);
	vector<vector<int>> matrix(M, row);
	for(int i=0; i<M; ++i)
	  for(int j=0; j<N; ++j)
	  matrix[i][j] = rand()%151 -50;
	int sum = 0;
	for(int i=0; i<M; ++i)
	  for(int j=0; j<N; ++j)
	  sum += matrix[i][j];
	cout<<"sum = "<<sum<<endl;
		// კომპიუტერმა დააგენერირა
		// რიცხვები
		
		-9 -5 93 25
		93 -30 -48 14
		34 -48 68 9
		-23 16 96 -12
		76 -37 96 -50
		
		// შედეგი
		
		sum = 358

7	/* ორგანზომილებიან მთელ რიცხვთა MxN ვექტორში შემთხვევითი რიცხვებია. ვიპოვოთ მათ შორის კენტების ჯამი და დავბეჭდოთ */
	const int M = 3, N = 4;
	vector<int> row(N);
	vector<vector<int>> matrix(M, row);
	for(int i=0; i<M; ++i)
	  for(int j=0; j<N; ++j)
	  matrix[i][j] = rand();
	int sum = 0;
	for(int i=0; i<M; ++i)
	  for(int j=0; j<N; ++j)
	  if(matrix[i][j]%2 == 1)
	  sum += matrix[i][j];
	cout<<"sum of odd numbers = "
	  <<sum<<endl;
		// კომპიუტერმა დააგენერირა
		// რიცხვები
		
		41 18467 6334 26500
		19169 15724 11478 29358
		26962 24464 5705 28145
		
		// შედეგი
		
		sum of odd numbers = 71527

2. ორგანზომილებიან ვექტორთან მუშაობის ფუნქციების მაგალითები

№	ფუნქციის განსაზღვრა	ფუნქციის გამოძახება

1	/* ორგანზომილებიანი მთელ რიცხვთა 3x4 ვექტორის კლავიატურიდან შემოტანილი რიცხვებით შევსების ფუნქცია */
	void fillMatrix(vector<vector<int>>& A)
	{
	  for(int i=0; i<A.size(); i++)
	  for(int j=0; j< A[i].size(); j++)
	  cin >> A[i][j];
	}
		int main(){
		 vector<int> row(4);
		vector< vector<int> > M(3,row);
		
		 fillMatrix(M);
		  . . .
		}

2	/* იგივე ფუნქცია: ორგანზომილებიანი ვექტორისთვის შემოვიღეთ ტიპი matrix */
	typedef vector<vector <int> > matrix;
	void fillMatrix(matrix & A){
	for(int i=0; i<3; i++)
	for(int j=0; j<4; j++)
	cin >> A[i][j];
	}
		int main(){
		vector<int> row(4);
		matrix M(3, row);
		fillMatrix(M);
		. . .
		}

3	/* ორგანზომილებიანი მთელ რიცხვთა MxN ვექტორის შემთხვევითი რიცხვებით შევსების ფუნქცია: სტრიქონისთვის შემოვიღეთ ტიპი row, ხოლო ორგანზომილებიანი ვექტორისთვის - ტიპი matrix */
	const int M =3, N =4;
	typedef vector<int> Row;
	typedef vector<Row> Matrix;
	void fillMatrix(Matrix & a){
	for(int i=0; i<M; i++)
	for (int j=0; j<N; j++)
	a[i][j] = rand();
	}
		int main(){
		Row R(N);
		Matrix A (M, R);
		fillMatrix(A);
		. . .
		}

	/* ორგანზომილებიანი მთელ რიცხვთა MxN ვექტორის ბეჭდვის ფუნქცია */
	void printMatrix(vector<vector<int>> A){
	for(int i=0; i<M; i++)
	{
	for(int j=0; j<N; j++)
	cout<<A[i][j]<<'\t';
	cout<<endl;
	}
	cout<<endl;
	}
		int main(){
		vector<int> row(N);
		vector< vector<int> > B(M,row);
		fillMatrix(B);
		printMatrix(B);
		. . .
		}

3	/* იგივე ფუნქცია: ორგანზომილებიანი ვექტორისთვის შემოვიღეთ ტიპი matrix */
	typedef vector<vector <int> > matrix;
	void printMatrix(matrix A){
	for(int i=0; i<M; i++)
	{
	for (int j=0; j<N; j++)
	cout<<A[i][j]<<'\t';
	cout<<endl;
	}
	cout<<endl;
	}
		int main(){
		vector<int> Row(N);
		matrix B(M, Row);
		fillMatrix(B);
		printMatrix(B);
		. . .
		}

4	/* ორგანზომილებიანი ნამდვილ რიცხვთა 2x3 ვექტორის კლავიატურიდან შევსების ფუნქცია */
	typedef vector<vector<double>> matrix;
	void fillMatrix(matrix & X){
	for(int i=0; i<2; i++)
	for (int j=0; j<3; j++){
	cin >>X[i][j];
	}
	}
		int main(){
		vector<double> row(3);
		matrix M(2, row);
		fillMatrix(M);
		. . .
		}

5	/* ორგანზომილებიანი ნამდვილ რიცხვთა K x P ვექტორის ბეჭდვის ფუნქცია */
	const int K =2, P =3;
	typedef vector<vector<double>> matrix;
	void printMatrix(matrix X){
	for(int i=0; i<K; i++)
	{
	for (int j=0; j<P; j++)
	cout <<X[i][j]<<'\t';
	cout<<endl;
	}
	cout<<endl;
	}
		int main(){
		vector<double> row(P);
		matrix M(K, row);
		fillMatrix(M);
		printMatrix(M);
		. . .
		}

6	/* ფუნქცია პოულობს და აბრუნებს ნამდვილ რიცხვთა M x N ვექტორის ელემენტების ჯამს */
	typedef vector<double> Row;
	typedef vector< Row > Matrix;
	double Sum(Matrix x){
	double s =0;
	for(int i=0; i<M; i++)
	for(int j=0; j<N; j++)
	s +=x[i][j];
	return s;
	}
		int main(){
		Row r(N);
		Matrix A(M, r);
		// ვექტორის შევსება
		. . .
		double s = Sum(A);
		cout<<"sum = "<<s
		<<endl;
		. . .
		}

7	/* ფუნქცია ითვლის მთელ რიცხვთა კვადრატული ვექტორის მთავარი დიაგონალის ელემენტების ჯამს */
	const int M = 3;
	typedef vector<int> Row;
	typedef vector< Row > Matrix;
	int Diagonal(Matrix x){
	int s =0;
	for(int i=0; i<M; ++i)
	s +=x[i][i];
	return s;
	}
		int main(){
		Row r(M);
		Matrix B(M, r);
		// ვექტორის შევსება
		. . .
		int s = Diagonal(B);
		cout<<"sum of diagonal"
		" elements = "<<
		s<<endl;
		. . .
		}

სააუდიტორიო სამუშაო:

1. დაწერეთ პროგრამა, რომელიც შეავსებს 3x4 ორგანზომილებიან ვექტორს შემთხვევითი მთელი რიცხვებით [0,50] შუალედიდან და შემდეგ დაბეჭდავს ვექტორს სტრიქონ-სტრიქონ.

```
#include <iostream>
#include <vector>
using namespace std;
int main(){
   vector<int> Row(4);
   vector<vector<int>> Matrix(3, Row);
   for(int i=0; i<3; i++)
       for(int j=0; j<4; j++){
           Matrix[i][j] = rand()%51;
   }
   for(int i=0; i<3; i++)
   {
       for(int j=0; j<4; j++)
```

```
            cout<<Matrix[i][j]<<'\t';
        cout<<endl;
    }
    cout<<endl;
    return 0;
}
```

პროგრამის შესრულების შედეგია:

```
41      5       10      31
44      16      3       33
34      35      44      44
Press any key to continue . . .
```

2. დაწერეთ პროგრამა, რომელიც შეავსებს $K \times P$ ორგანზომილებიან ვექტორს შემთხვევითი მთელი რიცხვებით [0,50] შუალედიდან და შემდეგ დაბეჭდავს ვექტორს სტრიქონ-სტრიქონ. K და P – შესაბამისად ორგანზომილებიანი ვექტორის სტრიქონებისა და სვეტების რაოდენობა – შემოიღეთ მუდმივების სახით.

```
#include <iostream>
#include <vector>
using namespace std;
const int K = 3, P = 4;
int main(){
    vector<int> Row(P);
    vector<vector<int>> Matrix(K, Row);
    for(int i=0; i<K; i++)
        for(int j=0; j<P; j++){
            Matrix[i][j] = rand()%51;
     }
    for(int i=0; i<K; i++)
     {
        for(int j=0; j<P; j++)
            cout<<Matrix[i][j]<<'\t';
        cout<<endl;
     }
     cout<<endl;
    return 0;
}
```

3. იგივე პირობისთვის ორგანზომილებიანი ვექტორის შემთხვევითი მთელი რიცხვებით შევსება და მისი სტრიქონ-სტრიქონ ბეჭდვა გააფორმეთ ფუნქციების სახით. გამოიძახეთ ორივე ფუნქცია ძირითად პროგრამაში.

ა).

```
#include <iostream>
#include <vector>
#include <ctime>
using namespace std;
const int K = 3, P = 4;
void fillMatrix(vector< vector<int> > & a);
void printMatrix(vector< vector<int> > a);
int main(){
    vector<int> R(P);
    vector< vector<int> > M(K, R);
    fillMatrix(M);
    printMatrix(M);
    return 0;
}
```

```
void fillMatrix(vector< vector<int> > & a){
   srand(time(NULL));
   for(int i=0; i<K; i++)
      for (int j=0; j<P; j++)
         a[i][j] = rand()%51;
}
void printMatrix(vector< vector<int> > a){
    for(int i=0; i<K; i++)
    {
       for(int j=0; j<P; j++)
          cout <<a[i][j]<<'\t';
       cout<<endl;
    }
    cout<<endl;
}
```

ბ). წინასწარ შემოვიღოთ ორგანზომილებიანი ვექტორის ტიპი matrix და მისი სტრიქონის ტიპი row.

```
#include <iostream>
#include <vector>
#include <ctime>
using namespace std;
typedef vector<int> row;
typedef vector< row > matrix;
void fillMatrix(matrix & a);
void printMatrix(matrix a);
int main(){
   int K, P;
   cout<<"SemoitaneT striqonSi elementebis raodenoba : ";
   cin >> P;
   cout<<"SemoitaneT striqonebis raodenoba : ";
   cin >> K;
    row R(P);
    matrix M(K, R);
    fillMatrix(M);
    printMatrix(M);
    return 0;
}
void fillMatrix(matrix & a){
   srand(time(NULL));
   for(int i=0; i<a.size(); i++)
      for (int j=0; j<a[i].size(); j++)
         a[i][j] = rand()%51;
}
void printMatrix(matrix a){
    for(int i=0; i<a.size(); i++)
    {
       for(int j=0; j<a[i].size(); j++)
          cout <<a[i][j]<<'\t';
       cout<<endl;
    }
    cout<<endl;
}
```

პროგრამის შესრულების ერთ-ერთი შედეგია:

```
29      48      31      17
38      36      42      13
0       32      50      13
```

```
Press any key to continue . . .
```

4. მონაცემების ფაილი წარმოადგენს M × N მართკუთხა ფორმის ნაკვეთის გეგმას წვიმის შემდეგ, სადაც სიმბოლო '+'-ით აღნიშნულია მშრალი ნაწილები, გუბეები კი – სიმბოლო '-'-ით. დაწერეთ ფუნქცია, რომელიც წაიკითხავს ინფორმაციას ფაილიდან, ჩაწერს მას სიმბოლური ტიპის ორგანზომილებიანი ვექტორის სახით და შემდეგ დაითვლის და დააბრუნებს გუბეების რაოდენობას ნაკვეთზე.

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
using namespace std;
const int M = 4, N = 5;
int Gubeebi();
int main (){
    int k;
    k =Gubeebi();
    cout<<"nakveTze "<<k<<" gubea\n ";
    return 0;
}
int Gubeebi(){
    vector< string > gegma;
    int counter =0;
     string row;
    ifstream fin("nakveTi.txt");
    while(fin>>row)
        gegma.push_back(row);
    for(int i=0; i<gegma.size(); i++)
      for(int j=0; j<gegma[i].size(); ++j)
         if(gegma[i][j] == '-') counter++;
    return counter;
}
```

პროგრამის მუშაობის შედეგია:

nakveTi.txt ფაილი	გამოტანის ეკრანი
-+--+ +-++- ++-++ ---++	nakveTze 9 gubea

დამოუკიდებელი სამუშაო:

1. დაწერეთ ფუნქცია, რომლის პარამეტრიც არის მთელ რიცხვთა ორგანზომილებიანი ვექტორი M × N. ვექტორის განზომილებები M და N წარმოადგენს კონსტანტებს. ფუნქციამ უნდა დაითვალოს ვექტორის 5-ის ჯერადი რიცხვების ჯამი. გამოიყენეთ ფუნქცია პროგრამაში კლავიატურიდან შევსებული ორგანზომილებიანი ვექტორისთვის. რიცხვების ჩაწერა ვექტორში ასევე გააფორმეთ ფუნქციის სახით.

2. დაწერეთ ფუნქცია, რომლის პარამეტრი არის ნამდვილ რიცხვთა ორგანზომილებიანი ვექტორი M × N. ვექტორის განზომილებები M და N წარმოადგენს კონსტანტებს. ფუნქციამ უნდა დაითვალოს და დააბრუნოს ვექტორის:
   ა) არაუარყოფითი ელემენტების რაოდენობა;
   ბ) იმ ელემენტების ჯამი, რომლებიც ეკუთვნიან [-10, 15] შუალედს;
   გ) იმ ელემენტების რაოდენობა, რომლებიც არ ეკუთვნიან [2, 30] შუალედს.

   გამოიყენეთ ფუნქცია პროგრამაში კლავიატურიდან შევსებული ორგანზომილებიანი ვექტორისთვის და დაბეჭდეთ შედეგი.

3. დაწერეთ პროგრამა, რომლებიც ნამდვილ რიცხვთა M x N ვექტორს შეავსებს კლავიატურიდან შემოტანილი რიცხვებით, დაითვლის რიგით მეორე სტრიქონის ელემენტების ჯამს, რიგით მესამე სვეტის ელემენტების ჯამს და დაბეჭდავს შედეგს. ვექტორთან ყველა მოქმედებისთვის დაწერეთ და პროგრამაში გამოიყენეთ სათანადო ფუნქციები.

გაარჩიეთ შესაბამისი პროგრამა :

```
#include <iostream>
#include <iomanip>
#include <vector>
using namespace std;
const int M = 3, N = 4;
typedef vector<double> row;
typedef vector< row > matrix;
void fillMatrix(matrix & x);
void printMatrix(matrix x);
int Row2(matrix x);
int Column3(matrix x);
int main(){
   row r(N);
   matrix vec(M, r);
   fillMatrix(vec);
   printMatrix(vec);
   int sum = Row2(vec);
   cout<<"Sum of second row's elements = "<<sum<<endl;
   sum = Column3(vec);
   cout<<"Sum of third column's elements = "<<sum<<endl;
   return 0;
}
void fillMatrix(matrix & x){
    for(int i=0; i<x.size(); i++)
       for(int j=0; j<x[i].size(); j++)
          cin >> x[i][j];
}
void printMatrix(matrix x){
   for(int i=0; i< x.size(); i++){
      for(int j=0; j< x[i].size(); j++)
         cout <<setw(7)<<x[i][j];
      cout<<endl;
   }
   cout<<endl;
}
int Row2(matrix x){
    int s =0;
    for(int k=0; k<x[1].size(); ++k)
        s += x[1][k];
    return s;
}
int Column3(matrix x){
    int s =0;
    for(int k=0; k<x.size(); ++k)
        s += x[k][2];
    return s;
}
```

პროგრამის შესრულების შედეგია:

```
1 4 6 8 0 3 5 12 -7 -21 45 10
      1       4       6       8
```

```
      0       3       5      12
     -7     -21      45      10
Sum of second row's elements = 20
Sum of third column's elements = 56
Press any key to continue . . .
```

4. დაწერეთ ფუნქცია, რომლის პარამეტრი არის მთელ რიცხვთა ორგანზომილებიანი ვექტორი $A(M\times K)$. ვექტორის განზომილებები M და K წარმოადგენს კონსტანტებს. ფუნქციამ უნდა დაადგინოს და დაბეჭდოს ვექტორის უდიდესი ელემენტის ინდექსები. გამოიყენეთ ფუნქცია ძირითად პროგრამაში კონკრეტული ორგანზომილებიანი ვექტორისთვის.

5. მონაცემების ფაილი წარმოადგენს $M\times N$ მართკუთხა ფორმის ნაკვეთის გეგმას წვიმის შემდეგ, სადაც სიმბოლო `'.'`-ით აღნიშნულია მშრალი ნაწილები, გუბეები კი – სიმბოლო `'0'`-ით. დაწერეთ ფუნქცია, რომელიც წაიკითხავს ინფორმაციას ფაილიდან და შემდეგ დაითვლის და დააბრუნებს:
   ა) ნაკვეთის მშრალი უბნების რაოდენობას;
   ბ)* იმ გუბეების რაოდენობას, რომლებიც ნაკვეთის ნაპირებთან მდებარეობს.

   ფუნქცია შექმენით სამი ხერხით: 1). მონაცემების განთავსებით ორგანზომილებიან სიმბოლური ტიპის ვექტორში; 2). მონაცემების განთავსებით სტრიქონების ვექტორში; 3). ვექტორის შემოღების გარეშე.

6. დაწერეთ ფუნქცია, რომელიც დაითვლის მთელ რიცხვთა $A(N\times N)$, $N=4$ ვექტორის რიგით პირველი და მესამე სტრიქონის ელემენტების ჯამს. გამოიყენეთ ფუნქცია ძირითად პროგრამაში კონკრეტული ორგანზომილებიანი ვექტორისთვის.

7. დაწერეთ ფუნქცია, რომელიც დაბეჭდავს ნამდვილ რიცხვთა $B(N\times M)$, $N=3$, $M=4$ ვექტორის:
   ა) მაქსიმალური ელემენტის სიდიდეს;
   ბ) მინიმალური ელემენტის ინდექსებს;
   გ) უდიდესი ელემენტის სიდიდესა და მის ინდექსებს;
   დ) უდიდესი და უმცირესი ელემენტების ჯამს.

   გამოიყენეთ ფუნქცია ძირითად პროგრამაში ორგანზომილებიანი ვექტორისთვის, რომლის ელემენტებიც წაიკითხეთ `reals.txt` ფაილიდან.

8. დაწერეთ ფუნქცია, რომელიც $A(N\times M)$ ვექტორში, სადაც N და M მთელი კონსტანტებია
   ა) მოძებნის K რიცხვის ტოლ ელემენტს და დააბრუნებს **`true`**-ს, თუ ასეთი ელემენტი ვექტორში არის. წინააღმდეგ შემთხვევაში დააბრუნებს **`false`**-ს. K–ს მნიშვნელობა შემოიტანეთ კლავიატურიდან;
   ბ) მოძებნის [-5, 100] დიაპაზონიდან ერთ-ერთ რიცხვს;
   გ) დაითვლის მინიმალური ელემენტის ტოლი ელემენტების რაოდენობას;
   დ) დაითვლის მაქსიმალური ელემენტისაგან განსხვავებული ელემენტების რაოდენობას.

9. დაწერეთ ფუნქცია, რომელიც დაითვლის მთელ რიცხვთა $P(M\times N)$, $M=4$, $N=3$ ვექტორის:
   ა) 17-ის ჯერადი ელემენტების ჯამს;
   ბ) დადებითი რიცხვების რაოდენობას;
   გ) კენტი რიცხვების საშუალო არითმეტიკულს;
   დ) კენტინდექსიანი სტრიქონების ელემენტების ჯამს.

10. მთელ რიცხვთა `A(NxN)`(მაგალითად, `N=5`) ორგანზომილებიანი ვექტორი შევსებულია კლავიატურიდან შემოტანილი რიცხვებით. შეამოწმეთ, წარმოადგენს თუ არა იგი

“მაგიურ კვადრატს”. კვადრატულ ვექტორს უწოდებენ “მაგიურ კვადრატს”, თუ მისი ყველა სტრიქონის, ყველა სვეტის და ორივე დიაგონალის ელემენტების ჯამი ტოლია.

11. დაწერეთ ფუნქცია, რომლის პარამეტრებია მთელ რიცხვთა ორგანზომილებიანი ვექტორი და მთელი X რიცხვი და რომელიც გაამრავლებს ვექტორს X -ზე.

12*. დაწერეთ ფუნქცია, რომელიც პარამეტრებად მიიღებს ნამდვილ რიცხვთა სამ ორგანზომილებიან ვექტორს და პირველი ორის ჯამს ჩაწერს მესამეში.

13*. დაწერეთ ფუნქცია, რომელიც პარამეტრებად მიიღებს მთელ რიცხვთა სამ ორგანზომილებიან ვექტორს და პირველი ორის ნამრავლს ჩაწერს მესამეში.

ლაბორატორიული სამუშაოს აღწერა:

- ორგანზომილებიან ვექტორთან მუშაობა

ლაბორატორიული სამუშაო:

ამოცანა 1: ჩაწერეთ ორგანზომილებიან `B(M x N),(M=5; N=3)` ვექტორში 15 შემთხვევითი რიცხვი [-10, 30] შუალედიდან, შემდეგ იპოვეთ ვექტორის ლუწი ელემენტების ჯამი და დაბეჭდეთ. ვექტორის შევსება და ლუწების ჯამის გამოთვლა გააფორმეთ ფუნქციების სახით.

შესაბამის C++ -პროგრამას აქვს სახე:

```
//////////////////////////////////////////////////////////
// პროგრამა: ორგანზომილებიანი ვექტორის შემთხვევითი რიცხვებით
//           შევსება და ლუწი ელემენტების ჯამის პოვნა
//////////////////////////////////////////////////////////
#include <iostream>
#include <iomanip>
#include <ctime>
#include <vector>
using namespace std;
const int M = 5, N = 3;
void fillMatrix(vector< vector<int> > & a);
void printMatrix(vector< vector<int> > a);
bool Even(int x);
int sumEven(vector< vector<int> > a);
int main(){
   vector<int> row(N);
   vector< vector<int> > B(M, row);
   fillMatrix(B);
   printMatrix(B);
   int S = sumEven(B);
   cout<<"Sum of even elements = "<<S<<endl;
   return 0;
}
void fillMatrix(vector< vector<int> > & a){
     srand(time(NULL));
     for(int i=0; i<M; i++)
       for(int j=0; j<N; j++)
          a[i][j] = rand()%41 - 10;
}
bool Even(int x){
   return x%2 == 0;
}
int sumEven(vector< vector<int> > a){
   int sum =0;
   for(int i=0; i<M; i++)
      for(int j=0; j<N; j++)
         if( Even(a[i][j]) )
            sum += a[i][j];
   return sum;
}
void printMatrix(vector< vector<int> > a){
   for(int i=0; i<M; i++)
   {
      for(int j=0; j<N; j++)
         cout <<setw(9)<<a[i][j];
```

```
      cout<<endl;
   }
   cout<<endl;
}
```

დავალება:

ა). შეასრულეთ პროგრამა რამდენიმეჯერ, ახსენით `setw` მანიპულატორის დანიშნულება;

ბ). ახსენით `srand(time(NULL));` ფუნქციის მოქმედება.

ამოცანა 2: `V(MxN)` ორგანზომილებიან ვექტორში ნამდვილი რიცხვებია. დაწერეთ ფუნქცია, რომელიც დააბრუნებს ვექტორის უდიდეს ელემენტს. რიცხვები ვექტორში ჩაწერეთ `reals.in` ფაილიდან. ამისათვის ასევე დაწერეთ ფუნქცია. პროგრამაში გამოიყენეთ ორივე ფუნქცია და დაბეჭდეთ უდიდესი ელემენტი.

```
//////////////////////////////////////
//  პროგრამა: ორგანზომილებიან ვექტორში
//            უდიდესი ელემენტის პოვნა
//////////////////////////////////////
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
using namespace std;
const int M = 3, N = 4;
typedef vector<double> row;
typedef vector< row > matrix;
void fillMatrix(matrix & x);
void printMatrix(matrix x);
double maxElem(matrix x);
int main(){
   row R(N);
   matrix V(M, R);
   fillMatrix(V);
   printMatrix(V);
   double Max = maxElem(V);
   cout<<"Udidesia "<<Max<<endl;
   return 0;
}
void fillMatrix(matrix & x){
    ifstream fin("reals.in");
     for(int i=0; i<x.size(); i++)
       for(int j=0; j<x[i].size(); j++)
          fin >> x[i][j];
}
void printMatrix(matrix x){
   for(int i=0; i< x.size(); i++)
   {
      for(int j=0; j< x[i].size(); j++)
         cout <<setw(10)<<x[i][j];
      cout<<endl;
   }
   cout<<endl;
}
double maxElem(matrix x){
   double max = x[0][0];
   for(int i=0; i<x.size(); i++)
      for(int j=0; j<x[i].size(); j++)
         if(x[i][j] > max)
            max = x[i][j];
```

```
    return max;
}
```

დავალება:

ა). შექმენით `reals.in` ფაილი და ჩაწერეთ მასში 12 ნამდვილი რიცხვი

ბ). შეასრულეთ პროგრამა და შეამოწმეთ მიღებული შედეგი.

ამოცანა 3: `A(MxN)` ორგანზომილებიან ვექტორში მთელი რიცხვებია. დაწერეთ ფუნქცია, რომელიც გაცვლის ვექტორის უმცირესი ელემენტის მქონე სტრიქონს და ბოლო სტრიქონს. რიცხვები ვექტორში ჩაწერეთ კლავიატურიდან.. პროგრამაში დაბეჭდეთ გარდაქმნილი ვექტორი.

შესაბამის C++ -პროგრამას აქვს სახე:

```
//////////////////////////////////////
//  პროგრამა: ორგანზომილებიან ვექტორში
//           სტრიქონების გაცვლა
//////////////////////////////////////
#include <iostream>
#include <iomanip>
#include <vector>
using namespace std;
const int M = 3, N = 4;
typedef vector<int> row;
typedef vector< row > matrix;
void fillMatrix(matrix & x);
void printMatrix(matrix x);
void Transform(matrix& x);
int main(){
   row R(N);
   matrix A(M, R);
   fillMatrix(A);
   Transform(A);
   printMatrix(A);
   return 0;
}
void Transform(matrix& x){
   int min = x[0][0], rowIndex =0;
   for(int i=0; i<x.size(); i++)
      for(int j=0; j<x[i].size(); j++)
         if(x[i][j] < min){
            min = x[i][j];
            rowIndex = i;
         }
   row temp;
   temp = x[rowIndex]; x[rowIndex] = x[M-1];
   x[M-1] = temp;
}
void fillMatrix(matrix & x){
    for(int i=0; i<x.size(); i++)
       for(int j=0; j<x[i].size(); j++)
          cin >> x[i][j];
}
void printMatrix(matrix x){
   for(int i=0; i<x.size();  i++){
      for(int j=0; j<x[i].size();  j++)
         cout <<setw(9)<<x[i][j];
      cout<<endl;
   }
   cout<<endl;
```

}

დავალება:

ა). გაუშვით პროგრამა შესრულებაზე, შეიტანეთ 12 მთელი რიცხვი და დააკვირდით მიღებულ შედეგს;

ბ). ახსენით `Transform` ფუნქციის ალგორითმი.

დამოუკიდებელი სამუშაო:

1. `V(M x N)` ორგანზომილებიამ ვექტორში ნამდვილი რიცხვებია. დაწერეთ ფუნქცია, რომელიც დააბრუნებს ვექტორის უმცირესი ელემენტის ინდექსებს. რიცხვები ვექტორში ჩაწერეთ `reals.in` ფაილიდან. ამისათვის ასევე დაწერეთ ფუნქცია. პროგრამაში გამოიყენეთ ორივე ფუნქცია და დაბეჭდეთ უმცირესი ელემენტი და მისი ინდექსები.

გაარჩიეთ შესაბამისი `C++` -პროგრამა:

```
////////////////////////////////////////////////
//  პროგრამა: ორგანზომილებიან ვექტორში უმცირესი
//            ელემენტის ინდექსების პოვნა
////////////////////////////////////////////////
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
using namespace std;
const int M = 4, N = 3;
typedef vector<double> row;
typedef vector< row > matrix;
void fillMatrix(matrix & x);
void printMatrix(matrix x);
int Indexes(matrix x, int& rowInd);
int main(){
   row R(N);
   matrix V(M, R);
   fillMatrix(V);
   printMatrix(V);
   int minRInd, minCInd;
   minCInd = Indexes(V, minRInd);
   cout<<"Umsiresia "<<V[minRInd][minCInd]<<endl
        <<minRInd<<" striqonSi, "<<minCInd<<" svetSi\n";
   system("PAUSE");
   return 0;
}
void fillMatrix(matrix & x){
    ifstream fin("reals.in");
     for(int i=0; i<M; i++)
       for(int j=0; j<N; j++)
          fin >> x[i][j];
}
void printMatrix(matrix x){
  for(int i=0; i<M; i++){
     for(int j=0; j<N; j++)
        cout <<setw(11)<<x[i][j];
     cout<<endl;
  }
  cout<<endl;
}
int Indexes(matrix x, int& rowInd){
   int columnInd;
```

```
    rowInd = columnInd =0;
    for(int i=0; i<M; i++)
       for(int j=0; j<N; j++)
           if(x[i][j] < x[rowInd][columnInd]){
                rowInd = i;
                columnInd = j;
           }
     return columnInd;
}
```

დავალება:

ა). შექმენით ფაილი `reals.in` და ჩაწერეთ მასში `M x N` ნამდვილი რიცხვი, მაგალითად:

```
0.25 -12.5 6.45 0 10.89 1.125 -5.7 19.65 -31.5 -3.67 9.75 0.05
```

ბ). შეასრულეთ პროგრამა. დარწმუნდით, რომ მიიღეთ შედეგი

```
   0.25   -12.5    6.45
      0   10.89   1.125
   -5.7   19.65   -31.5
  -3.67    9.75    0.05

Umsiresia -31.5
2 striqonSi, 2 svetSi
Press any key to continue . . .
```

2. დაწერეთ პროგრამა, რომელიც განსხვავებულ ნამდვილ რიცხვთა კვადრატული ვექტორისთვის გამოითვლის და დაბეჭდავს უდიდესი ელემენტის მქონე სტრიქონის და უმცირესი ელემენტის მქონე სვეტის სკალარულ ნამრავლს. სტრიქონის და სვეტის სკალარული ნამრავლი არის მათი რიგით შესაბამისი კომპონენტების ნამრავლების ჯამი. ვექტორის განზომილება შემოიღეთ მუდმივის სახით, ელემენტები შემოიტანეთ კლავიატურიდან.

3. მთელ რიცხვთა `C(MxN)` ორგანზომილებიანი ვექტორი შევსებულია კლავიატურიდან შემოტანილი რიცხვებით. დაბეჭდეთ ვექტორის ყველა "უნაგირა წერტილის" ინდექსები. ვექტორის ელემენტს უწოდებენ "უნაგირა წერტილს", თუ იგი უდიდესია თავის სტრიქონში და ამავე დროს უმცირესია თავის სვეტში.

4. მე-11 პრაქტიკული მეცადინეობის ყველა დავალებისთვის დაწერეთ, გამართეთ და შეასრულეთ შესაბამისი პროგრამები.

პრაქტიკული მეცადინეობა № 12

პრაქტიკული მეცადინეობის თემები:

- ფუნქციის არგუმენტების მნიშვნელობები გულისხმობის პრინციპით
- მომხმარებლის მიერ განსაზღვრული ტიპის შექმნა `class` –ის საშუალებით

მასალა დამოუკიდებლად გაცნობისათვის:

1. ფუნქციის პარამეტრების ინიციალიზება (არგუმენტების მნიშვნელობები გულისხმობის პრინციპით)

№	ფუნქცია	ფუნქციის გამოძახება - main()-ის ფრაგმენტი

1	/* გამოძახების შესაბამისად ფუნქცია ითვლის ან ორი მთელი რიცხვის, ან სამი მთელი რიცხვის ჯამს */
	int Sum(int a, int b, int c = 0){
	return a + b + c;
	}
		int main(){
		cout<<Sum(-12,49)<<endl;
		cout<<Sum(15,-5,10)
		<<endl;
		. . .
		}

2	// იგივე ფუნქცია
	
	int Sum(int a, int b, int c = 0){
	return a + b + c;
	}
		int main(){
		int x =5, y =9, u =14;
		cout<<x<<" + "<<y<<" = "
		<<Sum(x,y)<<endl;
		cout<<x<<" + "<<y<<" + "
		<<u<<" = "
		<<Sum(x,y,u)<<endl;
		. . .
		}

3	/* გამოძახების შესაბამისად ფუნქცია ითვლის ან ორი ნამდვილი რიცხვის, ან სამი ნამდვილი რიცხვის ნამრავლს */
	double Product(double a, double b,
	double c = 1)
	{
	return a * b * c;
	}
		int main(){
		cout<<Product(2,3)
		<<endl;
		cout<<Product(2,3,5)
		<<endl;
		. . .
		}

4	// იგივე ფუნქცია
	
	double Product(double a, double b,
	double c = 1)
	{
	return a * b * c;
	}
		int main(){
		double x =5, y =7, z =3;
		cout<<x<<" * "<<y<<" = "
		<<Product(x,y)<<endl;
		cout<<x<<" * "<<y<<" * "
		<<z<<" = "
		<<Product(x,y,z)<<endl;
		. . .
		}

5	/* ფუნქცია ითვლის ორი ნამდვილი რიცხვის ჯამს */
	float Add(float a =2, float b=3){
	return a + b;
	}
		int main(){
		cout<<Add()<<endl;
		cout<<Add(3)<<endl;
		cout<<Add(-5,9)<<endl;
		. . .
		}

6	/* ფუნქცია ითვლის სამი მთელი რიცხვის ნამრავლს */
	int Mult(int a =2, int b=3, int c =4){
	return a * b * c;
	}
		int main(){
		cout<<Mult()<<endl;
		cout<<Mult(5)<<endl;
		cout<<Mult(7,4)<<endl;
		cout<<Mult(3,5,6)<<endl;
		. . .

		}

7	/* გამოძახების მიხედვით ფუნქცია ბეჭდავს შესაბამის სახელს */
	void Name(string s = "Alex"){
	cout<<s<<endl;
	}
		int main(){
		Name();
		Name("George");
		Name("Mari");
		. . .
		}

2. ახალი ტიპის შემოღება ღია (public) წევრებიანი class –ის საშუალებით:

№	ტიპის განსაზღვრა	პროგრამის ფრაგმენტი

1	// კლასი აღწერს წრეს:
	class Circle
	{
	public:
	double radius;
	double Area(void)
	{
	return 3.14* radius* radius;
	}
	double Perimeter(void)
	{
	return 2*3.14* radius;
	}
	};
		int main ()
		{
		Circle x;
		x.radius = 2.01;
		cout << "radius= "
		<<x.radius<<endl;
		cout << "Area= "
		<<x.Area()<<endl;
		cout << "perimeter= "
		<<x.Perimeter()<<endl;
		return 0;
		}

2	// მოცემული კლასის ერთი ობიექტისთვის მეორეს მინიჭება
	class Circle
	{
	public:
	double radius;
	double Area(void)
	{
	return 3.14* radius* radius;
	}
	double Perimeter(void)
	{
	return 2*3.14* radius;
	}
	};
		int main () {
		Circle x,y;
		y.radius = 2.01;
		x = y;
		cout << "radius="
		<<x.radius<<endl;
		cout << "Area= "
		<<x.Area()<<endl;
		cout << "perimeter= "
		<<x.Perimeter()<<endl;
		return 0;
		}

3	/* კლასი აღწერს წრეს ცხადი კონსტრუქტორით ახალი ობიექტის კონსტრუირებისთვის */
	class Circle
	{
	public:
	double radius;
	Circle(double value) {
	radius = value;
	}
	double Area() {
	return 3.14* radius* radius;
	}
	double Perimeter() {
	return 2*3.14* radius;
	}
	};
		/* ეს საშუალებას გვაძლევს, რომ კლასის ობიექტები განვსაზღვროთ ისევე ადვილად, როგორც მარტივი ტიპის ცვლადები: */
		int main ()
		{
		Circle x(2.1);
		cout << "radius= "
		<<x.radius<<endl;
		cout << "Area= "
		<<x.Area()<<endl;
		cout << "perimeter= "
		<<x.Perimeter()<<endl;
		return 0;
		}

4	/* იგივე კლასი */
	class Circle
	{
	  public:
	  double radius;
	  Circle(double value) {
	  radius = value;
	  }
	  double Area() {
	  return 3.14* radius* radius;
	  }
	  double Perimeter() {
	  return 2*3.14* radius;
	  }
	};
		/* მაგრამ, ამ შემთხვევაში ვკარგავთ წინა მაგალითში მოყვანილ ობიექტის ინიციალიზაციის გზას: */
		int main ()
		{
		  Circle x; // შეცდომაა !
		  x.radius = 2.1;
		  cout << "radius= "
		  <<x.radius<<endl;
		  cout << "Area= "
		  <<x.Area()<<endl;
		  cout << "perimeter= "
		  <<x.Perimeter()<<endl;
		  return 0;
		}

5	/* გამოსავალს წარმოადგენს წრის კლასზე განაცხადის გაკეთების დროს ორი, არაცხადი და ცხადი კონსტრუქტორის გათვალისწინება ახალი ობიექტის კონსტრუირებისთვის */
	class Circle
	{
	  public:
	  double radius;
	  Circle(double value)
	  {
	  radius = value;
	  }
	  Circle()
	  {
	  radius = 0;
	  }
	  double Area()
	  {
	  return 3.14* radius* radius;
	  }
	  double Perimeter()
	  {
	  return 2*3.14* radius;
	  }
	};
		int main ()
		{
		  Circle x;
		  Circle y(1.1);
		  cout << "x.radius= "
		  << x.radius<<endl;
		  cout << "x.Area= "
		  << x.Area()<<endl;
		  cout << "x.perimeter= "
		  << x.Perimeter()<<endl;
		  cout << "y.radius= "
		  << y.radius<<endl;
		  cout << "y.Area= "
		  << y.Area()<<endl;
		  cout << "y.perimeter= "
		  << y.Perimeter()<<endl;
		  return 0;
		}

სააუდიტორიო სამუშაო:

1. დაწერეთ ფუნქცია Volume, რომელიც საზოგადოდ გამოითვლის მართკუთხა პარალელეპიპედის მოცულობას სამი პარამეტრის - სიმაღლე, ფუძის სიგრძე და სიგანე – მიხედვით. იგულისხმეთ, რომ ძალიან ხშირად გვხვდება ისეთი ფიგურები, რომლისთვისაც სიგრძე არის 3-ის, სიგანე 4-ის, ხოლო სიმაღლე 1-ის ტოლი. გამოიძახეთ ფუნქცია ძირითად პროგრამაში და დაბეჭდეთ: პარალელეპიპედის ფუძის ფართობი, სხვადასხვა პარალელეპიპედის მოცულობა.

```
#include <iostream>
using namespace std;
int Volume(int h =1, int x =3, int y =4);
// h - პარალელეპიპედის სიმაღლე
```

```
// x - პარალელეპიპედის ფუძის სიგრძე,
// y - პარალელეპიპედის ფუძის სიგანე,
int main()
{
   cout<< Volume()<<endl  // ფუძის ფართობი
       << Volume(2)<<endl
       << Volume(3,5)<<endl
       << Volume(4,5,8)<<endl;

   return 0;
}
int Volume(int h, int x, int y)
{
   return x * y * h;
}
```

პროგრამის შესრულების შედეგია:

```
12
24
60
160
Press any key to continue . . .
```

2. `class` –ის საშუალებით შემოიღეთ ტიპი, რომელიც აღწერს მართკუთხედს. მართკუთხედის კლასის ღია (**public**) ველებია: სიგრძე და სიგანე. კლასის ფუნქციებია: ცხადი კონსტრუქტორი, მონაცემების (მისი სიგრძე, სიგანე და ფართობი) ბეჭდვის ფუნქცია, მართკუთხედის ფართობის გამოთვლის ფუნქცია. პროგრამაში გააკეთეთ განაცხადი ობიექტზე – მართკუთხედზე და დაბეჭდეთ მისი მონაცემები. შემდეგ გააორმაგეთ ობიექტის ველების მნიშვნელობები და დაბეჭდეთ მიღებული მართკუთხედის მონაცემები. შემდეგ მიანიჭეთ ველებს განსხვავებული მნიშვნელობები და კიდევ ერთხელ დაბეჭდეთ მიღებული მართკუთხედის მონაცემები.

```
#include <iostream>
using namespace std;
class Rectangle
{
  public:
     double length;
      double width;
      Rectangle(double lengtValue, double widthValue)
      {
          length = lengtValue;
          width = widthValue;
      }
      double Area()
      {
         return length * width;
      }
      void printInfo()
      {
         cout<<"Length is equal to  "<< length <<endl;
         cout<<"width is equal to  "<< width <<endl;
         cout<<"area is equal to  "<< Area() <<endl;
      }
};
int main()
{
   Rectangle rec(3.0,2.1);
   rec.printInfo();
```

```
    rec.length *=2;
    rec.width *=2;
    cout<< endl <<"new data:"<<endl<<endl;
    rec.printInfo();

    rec.length = 5.1;
     rec.width = 4.5;
    cout<< endl <<"new data:"<<endl<<endl;
    rec.printInfo();
    return 0;
}
```

პროგრამის შესრულების ერთ-ერთი შედეგია:

```
Length is equal to  3
width is equal to  2.1
area is equal to  6.3

new data:

Length is equal to  6
width is equal to  4.2
area is equal to  25.2

new data:

Length is equal to  5.1
width is equal to  4.5
area is equal to  22.95
Press any key to continue . . .
```

3. შექმენით კლასი Sea ღია (**public**) მონაცემებით: ზღვის დასახელება და სიღრმე. კლასში იქონიეთ პარამეტრებიანი კონსტრუქტორი და ფუნქცია, რომელიც ბეჭდავს კლასის ობიექტის მონაცემებს. Seas.txt ფაილი შეიცავს ინფორმაციას ორი ზღვის შესახებ. პროგრამაში შექმენით კლასის ორ ობიექტი – Seas.txt ფაილის ინფორმაციის საფუძველზე და შემდეგ დაბეჭდეთ უდიდესი სიღრმის მქონე ზღვის მონაცემები.

```
#include <iostream>
#include <string>
#include <fstream>
using namespace std;
class Sea
{
  public:
    string name;
    int depth;
    Sea(string n, int d) : name(n),depth(d){ }
    void printInfo(){
      cout<<name<<" Sea - "
          <<depth<<endl;
   }

};

int main()
{
   string N; int D;
   ifstream fin("Seas.txt");
   fin>>N>>D;
```

```
    Sea A(N, D);
    fin>>N>>D;
    Sea B(N, D);
    cout<<"Maximum depth have\n";
    if(A.depth > B.depth)
        A.printInfo();
    else B.printInfo();
    return 0;
}
```

პროგრამის შესრულების შედეგია:

Seas.txt ფაილი	გამოტანის ეკრანი
Black 2938 Norwegian 3860	Maximum depth have Norwegian Sea - 3860 Press any key to continue . . .

4. ფაილში "Volcanoes.txt" წერია რამდენიმე ვულკანის მონაცემი – სახელი, სიმაღლე და ადგილმდებარეობა. მაგალითად :

```
etna                 3340     italy
hekla                1491     island
vezuv                1277     italy
kluchevskaia_sopka   4750     kamchatka
fuziama              3776     japan
krakatau             813      azia
kamerun              4070     kamerun
strombol             926      italy
hipokatepetl         5452     mexico
```

შექმენით კლასი ვულკანისთვის, რომლის ღია ველები იქნება სახელი, სიმაღლე და ადგილმდებარეობა, შემდეგ რიგრიგობით წაიკითხეთ ვულკანების მონაცემები ამ ფაილიდან და დაბეჭდეთ ყველაზე მაღალი ვულკანის მონაცემები.

```
#include <iostream>
#include <string>
#include <fstream>
using namespace std;
class Volcano
{
  public:
    string name;
    int height;
    string location;
     Volcano(string n, int h, string l) :
                name(n),height(h),location(l){ }
     void printVolcano(){
         cout<< name <<", its height is " << height <<" meters"<<endl
             <<"It is located in "<< location<<endl;
     }
};

int main()
{
    string N, Loc; int D;
    ifstream fin("Volcanoes.txt");
    fin>>N>>D>>Loc;
    Volcano maxVolcano(N, D, Loc);
    while(fin>>N>>D>>Loc)
    {
        Volcano nextxVolcano(N, D, Loc);
```

```
        if(maxVolcano.height < nextxVolcano.height)
            maxVolcano = nextxVolcano;
    }
    cout<<"Maximum height has volcano\n";
    maxVolcano.printVolcano();
    return 0;
}
```

პროგრამა დაბეჭდავს:

```
Maximum height has volcano
hipokatepetl, its height is 5452 meters
It is located in mexico
Press any key to continue . . .
```

დამოუკიდებელი სამუშაო:

1. დაწერეთ ფუნქცია **int** Sum(vector<**int**> a, **int** A =2, **int** B =10); რომელიც მთელ რიცხვთა a ვექტორისთვის შეკრებს მის ელემენტებს, დაწყებული A –ურიდან და დამთავრებული B –ურით (ჩათვლით), შემდეგ დააბრუნებს ამ ჯამს. რიცხვები ვექტორში შეიტანეთ კლავიატურიდან, არანაკლებ 15 მთელი რიცხვისა. ვექტორის შევსება გააფორმეთ ფუნქციის სახით. ძირითად პროგრამაში გამოიძახეთ Sum ფუნქცია ვექტორის:
   ა) რიგით მე-5-დან მე-11-მდე რიცხვის ჯამის დასადგენად;
   ბ) რიგით მე-7-დან მე-15-მდე რიცხვის ჯამის დასათვლელად;
   გ) რიგით მე-2-დან მე-10-მდე რიცხვის ჯამის დასათვლელად;
   დ) რიგით მე-3-დან მე-9-მდე რიცხვის ჯამის დასათვლელად;

   გაარჩიეთ გ) და დ) –ს შესაბამისი C++ -პროგრამა:

```
#include <iostream>
#include <vector>
using namespace std;
const int M =15;
void fillVector(vector<int>& a);
int Sum(vector<int> a, int A =2, int B =10);
int main()
{
   vector<int> V, X;
   fillVector(V);
   cout<<"RrigiT me-2-dan me-10-mde (maTi chaTvliT)"
         "\n elementebis jami = "
        <<Sum(V)<<endl;
   fillVector(X);
   cout<<"RrigiT me-3-dan me-9-mde (maTi chaTvliT)"
         "\n elementebis jami = "
        <<Sum(X, 3, 9)<<endl;
   return 0;
}
void fillVector(vector<int>& a){
  int element;
  cout<<"\nEnter "<<M<<" ints\n";
  for(int i=0; i<M; i++)
  {
     cin>>element;
     a.push_back(element);
  }
}
int Sum(vector< int > a, int A, int B){
```

```
    int s =0;
    for(int i=A-1; i<=B-1; i++)
       s += a[i];
    return s;
}
```

პროგრამის შესრულების შედეგია:

```
Enter 15 ints
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
RrigiT me-2-dan me-10-mde (maTi chaTvliT)
 elementebis jami = 54

Enter 15 ints
10 5 7 -5 15 25 33 -10 20 0 2 8 -13 38 1
RrigiT me-3-dan me-9-mde (maTi chaTvliT)
 elementebis jami = 85
Press any key to continue . . .
```

2. დაწერეთ ფუნქცია **double** sum(vector<**double**> x, **int** k =10); რომელიც ნამდვილ რიცხვთა x ვექტორისთვის დაითვლის ბოლო k ელემენტის ჯამს. ვექტორში რიცხვები შეიტანეთ reals.txt ფაილიდან, რომელშიც ჩაწერილია არანაკლებ 20 ნამდვილი რიცხვისა. ძირითად პროგრამაში გამოიძახეთ ფუნქცია ვექტორის ბოლო:

   ა) 10 რიცხვის ჯამის დასადგენად;

   ბ) 15 რიცხვის ჯამის დასათვლელად;

   გ) კლავიატურიდან შემოტანილი N (N<=20) რაოდენობის რიცხვის ჯამის დასათვლელად.

   რიცხვების ვექტორში ჩასაწერად ასევე შექმენით ფუნქცია.

   გაარჩიეთ გ) –ს შესაბამისი C++ -პროგრამა:

```
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;
void fillVector(vector<double>& );
double Sum(vector<double> , int k =10);
int main()
{
   vector<double> V;
   fillVector(V);
   cout<<"Ramdeni ricxvis jami vipovoT ? ";
   int N;
   cin>>N;
   cout<<"Veqtoris bolo "<<N<<" ricxvis jami = "
       <<Sum(V, N)<<endl;
   return 0;
}
void fillVector(vector<double>& a){
   ifstream input("reals.txt");
   double element;
   while( input>>element )
       a.push_back(element);
}
double Sum(vector< double > a, int k){
   double s =0;
   for(int i=a.size()-1; i>=a.size()-k; i--)
      s +=a[i];
   return s;
}
```

პროგრამის შესრულების შედეგია:

reals.txt ფაილი	გამოტანის ეკრანი
0.5 -1.5 2.56 18.35 -22.45 40.0 12.75 -50.55 0.25 -11.25 27.33 112.5 0.45 -0.05 24.75 -100.65 95.0 1.125 -34.75 60.5	Ramdeni ricxvis jami vipovoT ? 7 Veqtoris bolo 7 ricxvis jami = 45.925 Press any key to continue . . .

3. კლასი აღწერს ქვეყანას ღია მონაცემებით: ქვეყნის დასახელება, მოსახლეობის რაოდენობა (მლნ.), დაკავებული ტერიტორიის ფართობი (კვ. კმ); კლასის ფუნქციებია: კონსტრუქტორი და ინფორმაციის ბეჭდვის ფუნქცია. პროგრამაში გააკეთეთ განაცხადი კლასის ობიექტზე, დაბეჭდეთ ქვეყნის სახელი, მოსახლეობის რაოდენობა და დაკავებული ტერიტორიის ფართობი (გამოიძახეთ ბეჭდვის ფუნქცია).

4. შექმენით კლასი `Student`, რომელიც ახასიათებს სტუდენტს ღია მონაცემებით: გვარი, სახელი, ჯგუფის ნომერი და ქულა. კლასში იქონიეთ ფუნქციები – კონსტრუქტორი და ფუნქცია, რომელიც დაბეჭდავს ინფორმაციას სტუდენტის შესახებ. `Students.txt` ფაილი შეიცავს ინფორმაციას ორი სტუდენტის შესახებ. პროგრამაში გააკეთეთ განაცხადი კლასის ორ ობიექტზე – სტუდენტზე და დაბეჭდეთ იმ სტუდენტის მონაცემები, რომლის ქულაც მეტია.

5. შემოიღეთ კლასი რომელიც შეინახავს კომპიუტერის შესახებ შემდეგ ინფორმაციას: მის სახელს, მეხსიერების მოცულობას და ღირებულებას. კლასში განსაზღვრეთ რამდენიმე საჭირო ფუნქცია.

   ა) ძირითად პროგრამაში გააკეთეთ განაცხადი ამ ტიპის სამ `C1, C2` და `C3` ობიექტზე და მოიყვანეთ პროგრამული კოდის ფრაგმენტი, რომლის საშუალებითაც ამ ობიექტების ველებში მნიშვნელობების შეტანა მოხდება კლავიატურიდან;

   ბ) მოიყვანეთ პროგრამული კოდის ფრაგმენტი, რომელიც დაადგენს თუ რომელია უფრო ძვირად ღირებული კომპიუტერი და დაბეჭდავს მის მონაცემებს;

   გ) მოიყვანეთ პროგრამული კოდის ფრაგმენტი, რომელიც გაარკვევს თუ ამ სამ კომპიუტერს შორის რომლის სახელია უფრო მოკლე და დაბეჭდავს მისი მეხსიერების მოცულობას.

6. შემოიღეთ კლასი, რომელიც შეინახავს ბიბლიოთეკის წიგნის შესახებ შემდეგ ინფორმაციას: ავტორის გვარს და მკითხველების მიერ ამ წიგნის გატანის რაოდენობას. კლასში განსაზღვრეთ რამდენიმე საჭირო ფუნქცია.

   ა) ძირითად პროგრამაში გააკეთეთ განაცხადი ამ ტიპის სამ `B1, B2` და `B3` ობიექტზე და მიანიჭეთ თითოეულის ველებს რაიმე მნიშვნელობები;

   ბ) მოიყვანეთ პროგრამული კოდის ფრაგმენტი, რომელიც დაადგენს ნაკლებად პოპულარულ წიგნს და დაბეჭდავს მისი ავტორის გვარს;

   გ) მოიყვანეთ პროგრამული კოდის ფრაგმენტი, რომელიც გაარკვევს თუ ამ სამ წიგნს შორის რომლის დასახელებაა უფრო გრძელი და დაბეჭდავს ინფორმაციას ამ წიგნის შესახებ.

7. შექმენით კლასი `bankAccount`, რომელიც აღწერს ბანკის ანგარიშს. კლასის ღია მონაცემებია: ბანკის დასახელება, კლიენტის სახელი და გვარი, ანგარიშის ნომერი და ანგარიშზე არსებული თანხა. კლასში გაითვალისწინეთ კონსტრუქტორი და მონაცემების ბეჭდვის გუნქცია. ფაილში `"account.txt"` წერია რამდენიმე ანგარიშის შესახებ ინფორმაცია. პროგრამაში რიგრიგობით წაიკითხეთ ანგარიშების მონაცემები ამ ფაილიდან და დაბეჭდეთ ყველა იმ ანგარიშის მონაცემები, სადაც თანხა 500 ლარზე მეტია.

8. შექმენით კლასი, რომელიც შეინახავს მწვერვალის შესახებ შემდეგ ინფორმაციას: მის სახელს და სიმაღლეს (ეს არის კლასის ღია მონაცემები). კლასში იქონიეთ კონსტრუქტორი და მწვერვალის მონაცემების ბეჭდვის ფუნქცია. `"info.txt"` ფაილში წერია რამდენიმე მწვერვალის შესახებ ინფორმაცია. პროგრამაში რიგრიგობით წაიკითხეთ მწვერვალების მონაცემები ამ ფაილიდან და დაბეჭდეთ ყველაზე მაღალი მწვერვალის მონაცემები.

9. `"data.in"` ფაილში წერია მდინარეების შესახებ ინფორმაცია. მაგალითად :

```
alazani    407
rioni      333
enguri     206
xrami      220
mtkvari    1515
iori       351
sufsa      117
aragvi     112
```

შემოიღეთ კლასი `River`, რომელიც აღწერს მდინარეს შემდეგი ღია მონაცემებით: დასახელება და სიგრძე. კლასში იქონიეთ კონსტრუქტორი და მონაცემების ბეჭდვის ფუნქცია. დაწერეთ პროგრამა, რომელიც რიგრიგობით წაიკითხეთ მდინარეების მონაცემებს ფაილიდან და

ა) `"data.out"` ფაილში დაბეჭდავს ყველა იმ მდინარის მონაცემებს, რომლის სახელი იწყება `'a'` –ზე ;

ბ) ეკრანზე გამოიტანს ყველაზე მოკლე მდინარის მონაცემებს.

ლაბორატორიული მეცადინეობა № 12

ლაბორატორიული სამუშაოს აღწერა:

- ფუნქციის პარამეტრების ინიციალიზება
- მარტივი კლასის მაგალითი

ლაბორატორიული სამუშაო:

ამოცანა 1: დაწერეთ ფუნქცია, პროტოტიპით

```
void fullName(string ="John", string ="Doe");
```

რომელიც ბეჭდავს მის ორ პარამეტრს – სახელსა და გვარს. ძირითად პროგრამაში მოახდინეთ ფუნქციის ტესტირება.

შესაბამის C++ -პროგრამას აქვს სახე:

```
//////////////////////////////////////////////
// პროგრამა: ფუნქციის არგუმენტების მნიშვნელობები
//           გულისხმობის პრინციპით
//////////////////////////////////////////////
#include <iostream>
#include <string>
using namespace std;
void fullName(string ="John", string ="Doe");
int main(){
   fullName();
   string first_name;
   cout<<"First Name ? ";
   cin>>first_name;
   fullName(first_name);
   string last_name;
   cout<<"Last Name ? ";
   cin>>last_name;
   fullName(first_name, last_name);
   return 0;
}
void fullName(string first, string last){
    cout<<first<<' '<<last<<endl;
}
```

დავალება:

ა). შეასრულეთ პროგრამა, ახსენით რა რიგით ხდება პარამეტრების ჩანაცვლება არგუმენტებით;

ბ). დაამატეთ ფუნქციას სტრიქონის ტიპის მესამე პარამეტრი საწყისი მნიშვნელობით `boss`, რომელიც აღნიშნავს თანამდებობას. მოახდინეთ ახალი ფუნქციის ტესტირება.

მიაქციეთ ყურადღება ფუნქციის პროტოტიპის სინტაქსს და განსაზღვრის დროს ფუნქციის სათაურის სინტაქსს.

ამოცანა 2: შექმენით კლასი `Point2D`, რომელიც აღწერს წერტილს. წერტილის ღია მონაცემებია მისი ორი კოორდინატი. კლასის ფუნქციებია - უპარამეტრო კონსტრუქტორი, კოორდინატების ბეჭდვის ფუნქცია. `main()`–ში გააკეთეთ განაცხადი კლასის ობიექტზე – წერტილზე, დაბეჭდეთ მისი მონაცემები, შემდეგ შეცვალეთ წერტილის კოორდინატები და ისევ დაბეჭდეთ მისი მონაცემები.

შესაბამისი C++ -პროგრამა შემდეგი სახისაა:

```
//////////////////////////////////////////////
//  პროგრამა: Point2D კლასის შექმნა და ტესტირება
//////////////////////////////////////////////
```

```
#include <iostream>
using namespace std;
class Point2D
{
  public:
    int x, y;
    Point2D(){
        x = 0, y = 0;
    }
    void showPoint2D(){
       cout<<"( "<<x<<" , "
          <<y<<" )"<<endl;
    }
};

int main()
{
   Point2D P;
   P.showPoint2D();
   P.x = 10;
   P.showPoint2D();
   P.y = 20;
   P.showPoint2D();
   return 0;
}
```

დავალება:

ა). გაუშვით პროგრამა შესრულებაზე და დააკვირდით მიღებულ შედეგს;

ბ). შეცვალეთ წერტილის კოორდინატების მნიშვნელობები და ისევ შეასრულეთ პროგრამა;

ამოცანა 3: შექმენით კლასი `Student`, რომელიც ახასიათებს სტუდენტს ღია მონაცემებით გვარი, სახელი და ქულა. კლასში იქონიეთ ფუნქციები – უპარამეტრო კონსტრუქტორი და მონაცემების ბეჭდვის ფუნქცია. პროგრამაში გააკეთეთ განაცხადი კლასის ობიექტზე – სტუდენტზე და დაბეჭდეთ ობიექტის მონაცემები.

შესაბამის C++ -პროგრამას აქვს სახე:

```
////////////////////////////////////
//  პროგრამა: Student კლასის შექმნა
//            და ტესტირება
////////////////////////////////////
#include <iostream>
#include <string>
using namespace std;
class Student
{
  public:
      string firstName;
      string lastName;
      int qula;

      Student(){
         firstName = "Maia";
         lastName = "LomiZe";
         qula = 98;
      }
      void printInfo(){
          cout<<firstName<<' '
              <<lastName<<endl
```

```
            <<qula<<endl;
    }
};

int main()
{
   Student X;
   X.printInfo();
   return 0;
}
```

დავალება:

ა). რა ინფორმაციას დაბეჭდავს `printInfo` ფუნქცია, თუ პროგრამაში აღვწერთ კიდევ ერთ `Student` კლასის ობიექტს, და რატომ?

ბ). გადატვირთეთ კონსტრუქტორი: დაამატეთ კლასში პარამეტრებიანი კონსტრუქტორი

```
Student(string first, string last, int q = 100)
{
    firstName = first;
    lastName = last;
    qula = q;
}
```

`main` –ში გააკეთეთ განაცხადი კლასის ობიექტზე და მოახდინეთ მისი ინიციალიზება, მაგალითად ასე:

```
Student Y("Irakli","Beridze");
```

შემდეგ დაბეჭდეთ `Y` ობიექტის ყველა მონაცემი.

გ). დაამატეთ `main` –ში კოდის ფრაგმენტი, რომელიც:
შექმენის კლასის ახალ ობიექტს და მასზე განაცხადისთანავე მოახდინეთ მის ინიციალიზებას კლავიატურიდან შემოტანილი სტუდენტის გვარით, სახელით და ქულით, ხოლო შემდეგ დაბეჭდავს ამ სტუდენტის შესახებ ინფორმაციას.

დამოუკიდებელი სამუშაო:

1. შექმენით კლასი `Box` შემდეგი მონაცემებით: სიგრძე, სიგანე და სიმაღლე. კლასში იქონიეთ ფუნქციები – კონსტრუქტორი, მონაცემების ბეჭდვის ფუნქცია და ყუთის მოცულობის გამოთვლის ფუნქცია. `main` –ში გააკეთეთ განაცხადი `Box` კლასის ობიექტზე, დაბეჭდეთ მისი მონაცემები და მოცულობა.
2. მე -12 პრაქტიკული მეცადინეობის ყველა დავალებისთვის დაწერეთ, გამართეთ და შეასრულეთ შესაბამისი პროგრამები.

პრაქტიკული მეცადინეობა № 13

პრაქტიკული მეცადინეობის თემები:

- კლასის პარამეტრებიანი კონსტრუქტორი
- კლასის კონსტრუქტორი, რომლის პარამეტრებს მინიჭებული აქვთ საწყისი მნიშვნელობები
- კონსტრუქტორის გადატვირთვა

მასალა დამოუკიდებლად გაცნობისათვის:

1. `class` –ში სხვადასხვა ტიპის კონსტრუქტორების გამოყენება

№	კლასის განსაზღვრა	კლასის ობიექტის შექმნა და მასთან მუშაობა

1	`// განაცხადი კვადრატის კლასზე`
	`class Square`
	`{`
	`  public:`
	`  double side;`
	`  // პარამეტრიანი კონსტრუქტორი`
	`  Square(double );`
	`  void printSquare();`
	`};`
	`// კლასის ფუნქციების განსაზღვრა`
	`Square::Square(double G){`
	`  side = G;`
	`}`
	`void Square::printSquare(){`
	`  cout<<"side = "<<side<<endl;`
	`}`
		`int main()`
		`{`
		`  Square C(5);`
		`  C.printSquare();`
		`  . . .`
		`}`

2	`// კვადრატის კლასში პარამეტრიანი`
	`// კონსტრუქტორის განსაზღვრა სხვა სახით`
	`class Square`
	`{`
	`  public:`
	`  double side;`
	`  Square(double );`
	`  void printSquare();`
	`};`
	`Square::Square(double G): side(G){ }`
	`void Square::printSquare(){`
	`  cout<<"side = "<<side<<endl;`
	`}`
		`int main()`
		`{`
		`  Square A1(5), A2(2);`
		`  A1.printSquare();`
		`  A2.printSquare();`
		`  . . .`
		`}`

3	`// კონსტრუქტორის გადატვირთვა - კვადრატის`
	`// კლასი ორი კონსტრუქტორით`
	`class Square`
	`{`
	`  public:`
	`  double side;`
	`  // კონსტრუქტორები`
	`  Square( );  // უპარამეტრო`
	`  Square(double );  // პარამეტრიანი`
	`  void printSquare();`
	`};`
	`// კლასის რეალიზების ნაწილი - ფუნქციების`
		`int main(){`
		`  Square A;`
		`  A.printSquare();`
		`  Square B(7);`
		`  B.printSquare();`
		`  . . .`
		`}`

	// განსაზღვრა
	Square::Square( ){
	  side = 10;
	}
	Square::Square(double G){
	  side = G;
	}
	void Square::printSquare(){
	  cout<<"side = "<<side<<endl;
	}	

4	// კვადრატის კლასში კონსტრუქტორების
	// განსაზღვრა სხვა სახით
	class Square
	{
	  public:
	  double side;
	  Square( ):side(10){ }
	  Square(double G):side(G){ }
	  void printSquare();
	};
	void Square::printSquare()
	{
	  cout<<"side = "<<side<<endl;
	}
		int main()
		{
		  Square A;
		  A.printSquare();
		  Square B(7);
		  B.printSquare();
		  . . .
		}

5	/* კვადრატის კლასში კონსტრუქტორის
	პარამეტრის ინიციალიზება. ასეთი კონსტრუქტორი
	ცვლის ორივე – უპარამეტრო და პარამეტრიან –
	კონსტრუქტორს */
	class Square
	{
	  public:
	  double side;
	  Square(double = 3);
	  void printSquare();
	};
	// კლასის ფუნქციების იმპლემენტაცია
	Square::Square(double G)
	{
	  side = G;
	}
	void Square::printSquare()
	{
	  cout<<"side = "<<side<<endl;
	}
		int main()
		{
		  Square A;
		  A.printSquare();
		  Square B(7);
		  B.printSquare();
		  Square C(20), D;
		  C.printSquare();
		  D.printSquare();
		  . . .
		}

6	// კონსტრუქტორის, რომლის პარამეტრსაც აქვს
	// გულისხმობით მნიშვნელობა, განსაზღვრა სხვა
	// სახით
	class Square
	{
	  public:
	  double side;
	  Square(double G = 3) : side(G){ }
	  void printSquare();
	};
	// კლასის ფუნქციის იმპლემენტაცია
		int main()
		{
		  Square A;
		  A.printSquare();
		  Square B(7);
		  B.printSquare();
		  Square C(20), D;
		  C.printSquare();
		  D.printSquare();
		  . . .
		}

	`void Square::printSquare()`
	`{`
	`  cout<<"side = "<<side<<endl;`
	`}`	

7	`// მწვერვალის კლასის აღწერა`
	`class Peak`
	`{`
	`  public:`
	`  string name;`
	`  int height;`
	`  Peak(string = "Ushba", int = 4695);`
	`};`
	`// კლასის რეალიზების ნაწილი`
	`Peak::Peak(string n, int h)`
	`{`
	`  name = n; height = h;`
	`}`
		`int main(){`
		`  Peak P1;`
		`  cout<<"Name : "`
		`  <<P1.name <<endl`
		`  <<"Height : "`
		`  <<P1.height`
		`  <<endl;`
		`  Peak P2("Sxara",5068);`
		`  cout<<"Name : "`
		`  <<P2.name`
		`  <<endl`
		`  <<"Height : "`
		`  <<P2.height`
		`  <<endl;`
		`  . . .`
		`}`

სააუდიტორიო სამუშაო:

1. შექმენით მართკუთხედის კლასი ღია მონაცემებით: მართკუთხედის სიგრძე და სიგანე; კლასის ფუნქციებია: უპარამეტრო კონსტრუქტორი, პარამეტრებიანი კონსტრუქტორი, ფართობის და პერიმეტრის გამოთვლის ფუნქციები. პროგრამაში შექმენით კლასის ორი ობიექტი-მართკუთხედი, პირველი – უპარამეტრო კონსტრუქტორის, ხოლო მეორე – პარამეტრებიანი კონსტრუქტორის საშუალებით. შემდეგ დაბეჭდეთ ორივე მართკუთხედის გვერდები, პერიმეტრი და ფართობი.

```
#include <iostream>
using namespace std;
class Rectangle
{
   public:
     double length, width;
     Rectangle();
     Rectangle(double, double );
     double Area();
    double Perimeter();
    void printRectangle();
};
// კლასის ფუნქციების იმპლემენტაცია
Rectangle::Rectangle()
{
   length =3, width =4;
}
Rectangle::Rectangle(double a, double b)
{
   length =a, width =b;
}
double Rectangle::Area()
{
   return  length * width;
}
double Rectangle::Perimeter()
```

```
{
   return 2 * (length + width);
}
void Rectangle::printRectangle()
{
   cout<<"marTkuTxedis gverdebia ";
   cout<< length <<" "<< width <<endl;
   cout<<"farTobi = "<<Area()<<endl;
   cout<<"perimetri = "
       <<Perimeter()<<endl;
}
int main(){
   Rectangle A;
   cout<<"A ";
   A.printRectangle();
   Rectangle B(5,6);
   cout<<"B ";
   B.printRectangle();
   return 0;
}
```

პროგრამის შესრულების შედეგია:

```
A marTkuTxedis gverdebia 3 4
farTobi = 12
perimetri = 14
B marTkuTxedis gverdebia 5 6
farTobi = 30
perimetri = 22
Press any key to continue . . .
```

2. Rectangle კლასის აღწერაში ორივე კონსტრუქტორი შეცვალეთ ერთით – კონსტრუქტორით, რომლის პარამეტრებსაც მინიჭებული აქვთ მნიშვნელობები. პროგრამაში გააკეთეთ განაცხადი კლასის ორ ობიექტზე – მართკუთხედზე, დაბეჭდეთ მათი გვერდები, ფართობები და პერიმეტრი. შემდეგ შეცვალეთ პირველი მართკუთხედის სიგანე და მეორე მართკუთხედის ორივე გვერდი და კვლავ დაბეჭდეთ მართკუთხედების ფართობები და პერიმეტრები.

```
#include <iostream>
using namespace std;
class Rectangle
{
   public:
     double length, width;
     Rectangle(double =3, double =4);
     double Area();
     double Perimeter();
     void printRectangle();
};
// კლასის ფუნქციების იმპლემენტაცია
Rectangle::Rectangle(double a, double b)
{
   length =a, width =b;
}
double Rectangle::Area()
{
   return  length * width;
}
double Rectangle::Perimeter()
```

```
{
   return 2 * (length + width);
}
void Rectangle::printRectangle(){
  cout<<"marTkuTxedis gverdebia ";
  cout<< length <<" "<< width <<endl;
  cout<<"farTobi = "<<Area()<<endl;
  cout<<"perimetri = "
      <<Perimeter()<<endl;
}
int main()
{
   Rectangle A, B(5,7);
   cout<<"A ";
   A.printRectangle();
   cout<<"B ";
   B.printRectangle();
   A.width = 3.5;
   cout<<"A marTkuTxedis farTobi = "
       <<A.Area()<<endl
       <<"perimetri = "<<A.Perimeter()<<endl;
   B.length = 9;  B.width = 5.5;
   cout<<"B marTkuTxedis farTobi = "
       <<B.Area()<<endl
       <<"perimetri = "<<B.Perimeter()<<endl;
   return 0;
}
```

პროგრამის შესრულების შედეგია:

```
A marTkuTxedis gverdebia 3 4
farTobi = 12
perimetri = 14
B marTkuTxedis gverdebia 5 7
farTobi = 35
perimetri = 24
A marTkuTxedis farTobi = 10.5
perimetri = 13
B marTkuTxedis farTobi = 49.5
perimetri = 29
Press any key to continue . . .
```

3. შექმენით კლასი Sea მონაცემებით: ზღვის დასახელება და სიღრმე. კლასში გაითვალისწინეთ ფუნქციები: პარამეტრებიანი კონსტრუქტორი, ზღვის მონაცემების ბეჭდვის ფუნქცია. Seas.txt ფაილი შეიცავს ინფორმაციას სამი ზღვის შესახებ. პროგრამაში შექმენით კლასის სამი ობიექტი – Seas.txt ფაილის ინფორმაციის საფუძველზე და შემდეგ დაბეჭდეთ უდიდესი სიღრმის მქონე ზღვის მონაცემები.

```
#include <iostream>
#include <string>
#include <fstream>
using namespace std;
class Sea
{
  public:
    string name;
    int depth;
    Sea(string n, int d) : name(n),depth(d){ }
    void printSea();
};
```

```
void Sea::printSea()
{
    cout<<name<<" Sea - "
        <<depth<<endl;
}
int main()
{
   string N; int D;
   ifstream fin("Seas.txt");
   fin>>N>>D;
   Sea A(N, D);
   fin>>N>>D;
   Sea B(N, D);
   fin>>N>>D;
   Sea C(N, D);
   cout<<"Maximum depth have\n";
   Sea max = A;
   if(B.depth > max.depth)
      max = B;
   if(C.depth > max.depth)
      max = C;
   max.printSea();
   cout<<endl;
   return 0;
}
```

პროგრამის შესრულების შედეგია:

Seas.txt ფაილი	გამოტანის ეკრანი
Black 2938 Grenlandiis 4846 Norwegian 3860	Maximum depth have Grenlandiis Sea - 4846 Press any key to continue . . .

4. იგივე ამოცანაში უდიდესი სიღრმის მქონე ზღვის არჩევა მოახდინეთ კლასის ტიპზე პოინტერის გამოყენებით.

მითითება: კლასის ტიპზე პოინტერს უნდა მივანიჭოთ ობიექტის მისამართი, ხოლო მიმთითებლის გამოყენებით კლასის მონაცემებზე და ფუნქციებზე წვდომისატვის უნდა ვისარგებლოთ **->** ოპერატორით.

```
#include <iostream>
#include <string>
#include <fstream>
using namespace std;
class Sea
{
  public:
     string name;
     int depth;
     Sea(string n, int d) : name(n),depth(d){ }
      void printSea();
};
void Sea::printSea()
{
    cout<<name<<" Sea - "
      <<depth<<endl;
}
int main()
{
```

```
    string N; int D;
    ifstream fin("Seas.txt");
    fin>>N>>D;
    Sea A(N, D);
    fin>>N>>D;
    Sea B(N, D);
    fin>>N>>D;
    Sea C(N, D);

    cout<<"Maximum depth have\n";
    Sea * max = &A;
    if(B.depth > max->depth)
        max = &B;
    if(C.depth > max->depth)
        max = &C;
    max->printSea();
    cout<<endl;
    return 0;
}
```

დამოუკიდებელი სამუშაო:

1. კლასის საშუალებით შემოიღეთ მონაცემთა ტიპი, რომელიც აღწერს ქალაქს. კლასში გაითვალისწინეთ: მონაცემები – ქალაქის სახელი, მოსახლეობის რაოდენობა, დაკავებული ტერიტორიის ფართობი; კლასის ფუნქციები – პარამეტრებიანი კონსტრუქტორი, ქალაქის მონაცემების ბეჭდვის ფუნქცია. პროგრამაში გააკეთეთ განაცხადი კლასის ორ ობიექტზე – ქალაქზე, შემდეგ დაბეჭდეთ მეტი მოსახლეობის მქონე ქალაქის ინფორმაცია.

2. შექმენით კლასი `Triangle`, რომლის მონაცემებია სამკუთხედის სამი გვერდი. კლასში იქონიეთ ფუნქციები – უპარამეტრო კონსტრუქტორი; პარამეტრებიანი კონსტრუქტორი; მონაცემების ბეჭდვის ფუნქცია; სამკუთხედის ფართობის გამოთვლის ფუნქცია. პროგრამაში გააკეთეთ განაცხადი კლასის ორ ობიექტზე – სამკუთხედზე, დაბეჭდეთ მათი მონაცემები და შემდეგ შეადარეთ მათი ფართობები (დაბეჭდეთ სათანადო გზავნილი).

3. შექმენით კლასი `Circle`, რომლის მონაცემებია წრის ცენტრის კოორდინატები და რადიუსი. კლასში იქონიეთ ფუნქციები – უპარამეტრო კონსტრუქტორი; პარამეტრებიანი კონსტრუქტორი; მონაცემების ბეჭდვის ფუნქცია; წრეწირის სიგრძის გამოთვლის ფუნქცია. პროგრამაში გააკეთეთ განაცხადი კლასის ორ ობიექტზე – წრეზე ისე, რომ პირველს ჰქონდეს გულიცხმობით მინიჭებული მონაცემები, მეორეს კი – თქვენ მიერ კლავიატურიდან შემოტანილი მონაცემები. დაბეჭდეთ წრეების მონაცემები და შემდეგ შეადარეთ მათი წრეწირების სიგრძეები (დაბეჭდეთ სათანადო გზავნილი).

4. შექმენით კლასი `Square`. კლასში გაითვალისწინეთ: მონაცემი – კვადრატის გვერდი; კლასის ფუნქციები – კონსტრუქტორი, რომლის პარამეტრსაც მინიჭებულია აქვს საწყისი მნისვნელობა; კვადრატის ფართობის გამოთვლის ფუნქცია; კვადრატის პერიმეტრის გამოთვლის ფუნქცია; კვადრატის დიაგონალის გამოთვლის ფუნქცია. პროგრამაში აღწერეთ კლასის ორი ობიექტი ისე, რომ პირველს ჰქონდეს გულიცხმობით მინიჭებული მონაცემი, მეორეს კი – კლავიატურიდან შემოტანილი მონაცემი. დაბეჭდეთ ორივე კვადრატის მონაცემი, ფართობი და პერიმეტრი. შემდეგ შეადარეთ კვადრატების დიაგონალები და დაბეჭდეთ სათანადო გზავნილი.

5. შექმენით კლასი `Student` შემდეგი მონაცემებით – სტუდენტის სახელი, გვარი და რეიტინგის აღმნიშვნელი ასო. კლასის ფუნქციებია – კონსტრუქტორი, რომლის პარამეტრებსაც მინიჭებულია აქვთ საწყისი მნისვნელობები; ინფორმაციის ბეჭდვის ფუნქცია. პროგრამაში აღწერეთ კლასის ორი ობიექტი ისე, რომ პირველს ჰქონდეს ნაგულისხმევი მონაცემები, მეორეს კი – თქვენ მიერ კლავიატურიდან შემოტანილი

მონაცემები. დაბეჭდეთ ორივე სტუდენტის მონაცემები. შეუცვალეთ პირველ სტუდენტს რეიტინგის მონაცემი და შემდეგ დაბეჭდეთ მის შესახებ ინფორმაცია.

6. შექმენით კლასი `Student` მონაცემებით: სტუდენტის სახელი, გვარი, ასაკი და სქესი. კლასში გაითვალისწინეთ თქვენი აზრით საჭირო ფუნქციები. დაწერეთ პროგრამა, რომელშიც 3 სტუდენტის მონაცემების შეტანა ხდება კლავიატურიდან და შემდეგ იბეჭდება:
   ა) სტუდენტთა შორის რამდენია გოგონა;
   ბ) ყველაზე უფროსი სტუდენტის გვარი-სახელი;
   გ) ყველაზე ახალგაზრდა ვაჟის ინფორმაცია. თუ სტუდენტებს შორის ვაჟი არ არის, დაიბეჭდოს სათანადო გზავნილი.

7. `rectangles.txt` ფაილში ჩაწერილია სამი მართკუთხედის მონაცემები – თითოეულის სიგრძე და სიგანე. დაწერეთ პროგრამა, რომელიც დაბეჭდავს უმცირესი ფართობის მქონე მართკუთხედის მონაცემებს და თვით ფართობს. ამისათვის შექმენით მართკუთხედის კლასი. კლასში გაითვალისწინეთ თქვენი აზრით საჭირო ფუნქციები. უმცირესი ფართობის მქონე მართკუთხედის დასადგენად გამოიყენეთ კლასის ტიპზე პოინტერი.

8. `data.txt` ფაილში ჩაწერილია სამი სახლის მონაცემები – თითოეულის სართულების რაოდენობა, სადარბაზოების რიცხვი და ყოველ სართულზე ბინების რაოდენობა. დაწერეთ პროგრამა, რომელიც `answer.txt` ფაილში დაბეჭდავს იმ სახლის სრულ ინფორმაციას, რომელშიც ყველაზე მეტი ბინაა. ამისათვის შექმენით სახლის კლასი, რომელშიც იქონიეთ თქვენი აზრით საჭირო ფუნქციები. ამოცანის ამოხსნისას გამოიყენეთ კლასის ტიპზე პოინტერი.

ლაბორატორიული მეცადინეობა № 13

ლაბორატორიული სამუშაოს აღწერა:

- მრავალფაილიანი პროექტი (multifile project) კლასის მონაწილეობით

ლაბორატორიული სამუშაო:

ამოცანა 1. შექმენით მართკუთხედის კლასი ღია წევრებით. ველები: მართკუთხედის სიგრძე და სიგანე; მეთოდები: უპარამეტრო კონსტრუქტორი, პარამეტრებიანი კონსტრუქტორი, ფართობის გამოთვლის ფუნქცია, პერიმეტრის გამოთვლის ფუნქცია.

პროგრამაში შექმენით კლასის ორი ობიექტი-მართკუთხედი, პირველი – უპარამეტრო კონსტრუქტორის, ხოლო მეორე – პარამეტრებიანი კონსტრუქტორის საშუალებით. შემდეგ დაბეჭდეთ ორივე მართკუთხედის მონაცემები.

ამოხსნა. პრაქტიკულ მეცადინეობებზე ჩვენ ვხსნიდით ასეთ ამოცანებს მარტივი ერთფაილიანი პროექტის სახით, ვათავსებდით რა განაცხადს კლასზე `main()`-ის წინ, ხოლო კლასის მეთოდების იმპლემენტაციას - კლასის განაცხადში ან კლასის გარეთ. ჩვენი ამოცანისთვის პროექტის ერთადერთ ფაილს ექნება ასეთი სახე:

```
#include <iostream>
using namespace std;
class Rectangle{
   public:
     double length, width;
     Rectangle();
     Rectangle(double, double );
     double Area();
     double Perimeter();
     void printRectangle();
 };
 // კლასის რეალიზების ნაწილი
 Rectangle::Rectangle() {
    length =3, width =4;
 }
 Rectangle::Rectangle(double a, double b) {
    length =a, width =b;
 }
 double Rectangle:: Perimeter() {
    return  2*(length + width);
 }
 double Rectangle::Area() {
    return  length * width;
 }
 void Rectangle::printRectangle(){
   cout<<"marTkuTxedis gverdebia ";
   cout<< length <<" "<< width <<endl;
   cout<<"farTobi = "<<Area()<<endl;
   cout<<"perimetri = "<< Perimeter()<<endl;
 }
int main(){
   Rectangle A;
   cout<<"A ";
   A.printRectangle();
   Rectangle B(5,6);
   cout<<"B ";
   B.printRectangle();
   return 0;
}
```

ასეთ მიდგომას აქვს რამდენიმე სერიოზული ნაკლი. აღვნიშნოთ ზოგიერთი:

- კოდი ნაკლებად დაცულია: კლასის სახელი, ცვლადების და ფუნქციების სახელები ნაკლებად კონტროლდება კომპილერის მიერ. მაგალითად, როდესაც ქვემოთ გავაკეთებთ კლასს Visual Studio-ს საშუალებებით, კომპილერი არ მოგვცემს `Rectangle` სახელის დარქმევის საშუალებას, რადგან ეს სახელი რეზერვირებულია (ასეთი კლასი უკვე არსებობს).
- როდესაც პრაქტიკული ღირებულების მქონე ამოცანაში იქმნება კლასი, მისი ზომები და სირთულე ბევრად აღემატება აქ განხილული მაგალითებისას. ამ შემთხვევებში Visual Studio-ს გარემოს დახმარება შეუცვლელია. მით უმეტეს, რომ იგი გვეხმარება ცვლადებისა და ფუნქციების შექმნაში, თვითონ გაანაწილებს სხვადასხვა საქაღალდეში header და cpp ფაილებს და სხვა.

შევქმნათ ამ ერთფაილიანი პროექტის ეკვივალენტური მრავალფაილიანი პროექტი. ამისთვის განვახორციელოთ შემდეგი მოქმედებები:

ვთქვათ უკვე შექმნილი გვაქვს ცარიელი პროექტი სახელით A და საქაღალდეში Source Code გახსნილი გვაქვს .cpp ფაილი (მაგალითად) სახელით main. ამ ფაილის შექმნა ყველაზე ადვილია, საკმარისია

```
#include <iostream>
```

სტრიქონის შემდეგ დავამატოთ

```
#include “Rect.h”
```

რაც ნიშნავს, რომ ვამატებთ ჩვენს მიერ შექმნილი კლასის ინტერფეისს, რომელიც სხვა ფაილშია და იმიტომ ჩართვა სჭირდება.

ახლა შევქმნათ კლასი და მისი იმპლემენტაცია. ამაში ჩვენ მნიშვნელოვნად ვისარგებლებთ იმ ყალიბებით, რაც უკვე გამზადებულია Visual Studio-ს გარემოში პროგრამისტების დასახმარებლად.

მივიყვანოთ მაუსი პროექტის სახელთან “A” (იგივე Solution Exsplorer ფანჯარაში), დავაჭიროთ მაუსის მარჯვენა ღილაკს ხელი და გადავიდეთ პუნქტზე Add, შემდეგ ავირჩიოთ სტრიქონი Class. გაიხსნება ფანჯარა Add Class – A. Installed Templates სვეტში მოვნიშნოთ C++ სტრიქონი და დავაჭიროთ Add-ს (მარჯვენა ქვედა კუთხეში). გაიხსნება Generic C++ Class Wizard – A. ტექსტურ ველში სახელით Class Name ჩაწერეთ თქვენი კლასის სახელი Rect (შეგიძლიათ სცადოთ სრული სახელის Rectangle –ის ჩაწერა, რასაც ვერ შევძლებთ, რადგან ეს სახელი რეზერვირებულია). ტექსტურ ველებში სახელებით .h-file და .cpp-file ავტომატურად ჩაიწერება Rect.h და Rect.cpp. Access ჩამონათვალში დავტოვოთ გულისხმობის პრინციპით ჩაწერილი public და დავაჭიროთ Finish ღილაკს.

შედეგად, ჩვენს პროექტს დაემატა ორი ფაილი: Rect.h და Rect.cpp. პირველი შეიცავს Rect კლასზე განაცხადის ესკიზს, მომზადებული კონსტრუქტორის და დესტრუქტორის პროტოტიპებით, ხოლო მეორე ფაილი წარმოადგენს კლასის იმპლემენტაციას და აქაც არის კონსტრუქტორის და დესტრუქტორის იმპლემენტაციის ჩანასახები.

ახლა, კლასს დავამატოთ წევრები: ველები და მეთოდები (ანუ ცვლადები და ფუნქციები). Class View ფანჯარაში (რომელიც, თუ უკვე არაა გახსნილი, იხსნება View მენიუდან), მივიყვანოთ მაუსი პროექტის სახელზე “A”, ჩამოვშალოთ კლასის სახელი Rect. ახლა მივიყვანოთ მაუსი კლასის სახელზე Rect, დავაჭიროთ მარჯვენა ღილაკს და ავირჩიოთ Add-> Add Variable. იხსნება Add Member Variable Wizard - A ფანჯარა. წვდომის (Access) ველში დავტოვოთ public. Variable Type ველში ავირჩიოთ მონაცემის ტიპი. იმ კლასისთვის, რომელსაც ჩვენ ვქმნით, გვჭირდება ორი ველი – ორი ნამდვილი ცვლადი, სიგრძე და სიგანე. შევქმნათ პირველი მათგანი. ჩამოვშალოთ ველი სახელით Variable Type და ავირჩიოთ ტიპი

double. ტექსტურ ველში, რომელიც გამოყოფილია ცვლადის სახელისთვის, ჩავწეროთ length. შეგვიძლია ამ ცვლადს დავურთოთ კომენტარიც, ეს სავალდებულო არაა (როგორც გნებავთ). დავაჭიროთ დამთავრების (Finish) ღილაკს და დავათვალიეროთ ფაილებში Rect.h და Rect.cpp მომხდარი ცვლილებები.

ანალოგიური მოქმედებებით შევქმნათ ცვლადი

```
double width;
```

კვლავ დავათვალიეროთ ფაილებში Rect.h და Rect.cpp მომხდარი ცვლილებები. ყურადღება მივაქციოთ უპარამეტრო კონსტრუქტორის ცვლილებას.

ახლა შევქმნათ კლასის მეთოდები.

Class View ფანჯარაში მარჯვენა ღილაკი დავაჭიროთ კლასის სახელს, ავირჩიოთ Add-> Add Function. იხსნება Add Member Function Wizard - A ფანჯარა, რომელიც ინტუიციურად ადვილად გამოსაყენებელია. დავიწყოთ

```
double Area();
```

ფუნქციის შექმნით. დასაბრუნებელ მნიშვნელობად (Return type), ცხადია ავიღოთ double, ფუნქციის სახელად Area, არგუმენტები ამ ფუნქციას არ სჭირდება და ამიტომ დავაჭერთ დამთავრების ღილაკს.

თუ ახლა კვლავ დავათვალიერებთ Rect.h და Rect.cpp ფაილებს, ვნახავთ რომ კლასში შეიქმნა ფუნქციის პროტოტიპი, ხოლო იმპლემენტაციის ფაილში – იმპლემენტაციის ესკიზი. დავასრულოთ ეს ესკიზი. ამისთვის, ფუნქციის ტანი შევცვალოთ ისე, რომ ფუნქციამ მიიღოს სახე

```
double Rect::Area()
{
    return  length * width;
}
```

და დავიმახსოვროთ ცვლილებები.

ანალოგიურად გავაკეთოთ Perimeter ფუნქცია.

განსაკუთრებით საინტერესოა ცხადი (პარამეტრებიანი) კონსტრუქტორის შექმნა, რადგან მას არა აქვს დასაბრუნებელ;ი მნიშვნელობა, მაგრამ აქვს არგუმენტები.

დავიწყოთ ისევ Add-> Add Function-ის არჩევით. Add Member Function Wizard - A ფანჯრის საშუალებით შევქმნათ

```
Rect::Rect(double a, double b)
{
    length =a, width =b;
}
```

ფუნქცია. დასაბრუნებელ მნიშვნელობის ველი (Return type) დავტოვოთ ცარიელი (ანუ წავშალოთ რაც იქ წერია). ფუნქციის სახელად ავიღოთ კლასის სახელი Rect. შევადგინოთ არგუმენტები (პარამეტრების) სია. დავიწყოთ პირველი პარამეტრიდან. პარამეტრის ტიპად ავიღოთ double, პარამეტრის სახელად a. ამ მომენტში გააქტიურდება Add ღილაკი, რადგან პირველი პარამეტრი უკვე მზადაა დასამატებლად. დავაჭიროთ ამ ღილაკს და ვნახავთ, რომ პირველი პარამეტრი გადავა მარჯვენა ფანჯარაში, რომელსაც ჰქვია პარამეტრების სია. ეს აგრეთვე ნიშნავს, რომ ტიპის და სახელის ველების გამოყენება უკვე შეიძლება მეორე პარამეტრისთვის. პარამეტრის ტიპად ავიღოთ double, პარამეტრის სახელად b.

თუ ახლაც დავათვალიეროთ ფაილებში Rect.h და Rect.cpp მომხდარი ცვლილებებს, ვნახავთ, რომ კლასში შეიქმნა კონსტრუქტორის პროტოტიპი, ხოლო იმპლემენტაციის ფაილში –

კონსტრუქტორის იმპლემენტაციის ესკიზი. დავასრულოთ ეს ესკიზი. ამისთვის, ფუნქციის ტანი შევცვალოთ ისე, რომ მივიღოთ

```
Rect::Rect (double a, double b)
{
    length =a, width =b;
}
```

printRectangle ფუნქვია იქმნება ანალოგიურად.
ამით კლასების შექმნა დასრულებულია. ახლა საჭიროა მთავარი main() ფუნქციის დაწერა. საბოლოოდ ჩვენი მრავალფაილიანი პროექტის ფაილები მიიღებს სახეს:

```
// Rect.h (header) ფაილი
#pragma once
class Rect
{
   public:
      Rect(void);
      ~Rect(void);
      double length;
      double width;
      double Area(void);
      double Perimeter(void);
      Rect(double a, double b);
      void printRectangle(void);
};
```

```
// Rect.cpp კლასის იმპლემენტაციის ფაილი
#include "Rect.h"
#include <iostream>
using namespace std;

Rect::Rect(void)
      : length(0)
      , width(0)
{
}

Rect::~Rect(void)
{
}

double Rect::Area(void)
{
      return length * width;
}

double Rect::Perimeter(void)
{
      return 2 * (length + width);
}

Rect::Rect(double a, double b)
{
      length =a, width =b;
}
```

```
void Rect::printRectangle(void)
{
      cout << "martkutxedis gverdebia ";
      cout << length << " " << width << endl;
      cout << "fartibi = " << Area() <<endl;
      cout << "perimetri = " << Perimeter() <<endl;
}
```

```
// main.cpp ფაილი
#include <iostream>
#include "Rect.h"
using namespace std;

int main()
{
      Rect A;
      cout << "A: ";
      A.printRectangle();

      Rect B(5,6);
      cout << "B: ";
      B.printRectangle();

      return 0;
}
```

დამოუკიდებელი სამუშაო:

მე -13 პრაქტიკული მეცადინეობის ყველა დავალებისთვის დაწერეთ, გამართეთ და შეასრულეთ შესაბამისი პროგრამები.