საქართველოს ტექნიკური უნივერსიტეტი

გ. ჯანელიძე

ალგორითმიზაციის საფუძვლები

დამტკიცებულია სტუ-ს
სასწავლო მეთოდური
საბჭოს მიერ

თბილისი
2007

უაკ 681.3.06

დამხმარე სახელმძღვანელოში წარმოდგენილია ალგორითმის დამუშავების მეთოდიკა, რომელიც წარმოადგენს კომპიუტერზე ამოცანის ამოხსნის ძირითად ეტაპს. პროგრამირების სწავლების დაწყებამდე მიზანშეწონილია, პროგრამირების რომელიმე ენისგან დამოუკიდებლად, შესწავლილ იქნას ალგორითმების დამუშავება ბლოკ-სქემების ან სხვა საშუალებების გამოყენებით. ასეთი მიდგომა უადვილებს სტუდენტებს პროგრამირების რომელიმე ენაზე გადასვლას.

დამხმარე სახელმძღვანელო განკუთვნილია ინფორმატიკის სფეროს I კურსის სტუდენტებისათვის.

რეცენზენტი: სრული პროფ. გ. სურგულაძე

საგამომცემლო სახლი „ტექნიკური უნივერსიტეტი“, 2007

ISBN 978-99940-951-1-7

შინაარსი

I თავი
ალგორითმიზაცია
1.1. ალგორითმის ცნება

პროგრამირების პროცესში უმთავრესია ამოცანის ამოხსნის ალგორითმის შემუშავება. ალგორითმიზაცია კომპიუტერზე ამოცანის ამოხსნის ეტაპებიდან ერთ-ერთ რთულ ეტაპს შეიძლება მივაკუთვნოთ.

ნებისმიერი ამოცანის გადაწყვეტა გულისხმობს გარკვეული მოქმედებების შესრულებას. მოქმედებათა სასრულ მიმდევრობას, რომლის ზუსტად შესრულების შემთხვევაში აღწევენ წინასწარ დასახულ მიზანს, ალგორითმი ეწოდება. ალგორითმის მკაცრი განმარტება არ არსებობს. შეიძლება ითქვას, რომ ეს ალგორითმის არსის ახსნის მხოლოდ მცდელობაა.

ადამიანს ყოველდღიურად უხდება რაღაც წესების მიმდევრობის დაცვა, შეასრულოს სხვადასხვა ინსტრუქციები ან მითითებები. მაგალითად, გადასასვლელზე როცა გადავდივართ შუქნიშნის გარეშე, ჯერ უნდა გავიხედოთ მარცხნივ, თუ მანქანები არ მოძრაობენ გადავკვეთოთ ნახევარი გზა, ხოლო თუ მანქანები მოძრაობენ დაველოდოთ გზის განთავისუფლებას და გადავკვეთოთ ქუჩა შუახაზამდე. შემდეგ გავიხედოთ მარჯვნივ, თუ მანქანები არ არის გადავიდეთ ბოლომდე, ხოლო თუ მანქანები მოძრაობენ, დაველოდოთ გზის განთავისუფლებას და გადავკვეთოთ ქუჩა ბოლომდე.

მათემატიკაში მსგავსი ამოცანებს ამოსახსნელად გამოიყენება განსაზღვრული წესები, რომლებიც აღწერენ მოქმედებათა მიმდევრობას. მაგალითად, წილადი რიცხვების შეკრების წესი, კვადრატული განტოლების ამოხსნის წესი და ა.შ. ნებისმიერი ინსტრუქცია და წესი მოიცავს მოქმედებებს, რომლებიც უნდა შესრულდეს განსაზღვრული მიმდევრობით. ამოცანის ამოსახსნელად უნდა ვიცოდეთ: რა არის მოცემული, რა უნდა მივიღოთ, რა მოქმედებები და რა მიმდევრობით უნდა შესრულდეს. მოქმედებათა მიმდევრობა, რომლის დანიშნულებაა საძიებო შედეგის მიღება, არის სწორედ ალგორითმი.

„ალგორითმი“, როგორც ცნება, წარმოიშვა უდიდესი შუააზიელი მათემატიკოსის სახელის ლათინური ფორმიდან: მუჰამედ იბნ მუსა ალ-ხორეზმი (Alhorithmi), რომელიც ცხოვრობდა 783-850 წლებში. თავის წიგნში მან აღწერა ნატურალური რიცხვების ჩაწერის წესები არაბული ციფრებით და მათზე მოქმედებები „ქვეშმიწერით“. XII საუკუნეში ეს წიგნი ითარგმნა ლათინურად და ფართოდ გავრცელდა ევროპაში.

ალგორითმის ცნება ითვლება ერთ-ერთ მნიშვნელოვან ცნებად არა მარტო მათემატიკაში, არამედ, თანამედროვე მეცნიერებაში. უფრო მეტიც, თანამედროვე ინფორმატიზაციის ერაში ალგორითმები გადაიქცა ცივილიზაციის ერთ-ერთ უმნიშვნელოვანეს ფაქტორად.

ალგორითმის შემსრულებელი - ეს არის რაღაც აბსტრაქ-

ტული ან რეალური (ტექნიკური, ბიოლოგიური ან ბიო-ტექნიკური) სისტემა, რომელსაც აქვს ალგორითმებით აღწერილი მოქმედებების შესრულების უნარი.

შემსრულებლებს ახასიათებთ:

- გარემო;
- ელემენტარული მოქმედებები;
- ბრძანებათა სისტემა;
- მტყუნებები.

1.2. ალგორითმის თვისებები

ალგორითმი ხასიათდება შემდეგი თვისებებით:

- ალგორითმი უნდა იყოს გარკვეული, ე.ი. ნებისმიერი შემსრულებლისათვის მისი ყოველი ბრძანება უნდა იყოს გასაგები და ადვილად შესრულებადი.

- ალგორითმი უნდა იყოს ბიჯებად (ეტაპებად) დაყოფილი. იგი ამოცანის ამოხსნის პროცესს უნდა წარმოადგენდეს, როგორც წინასწარ განსაზღვრულ მარტივ მოქმედებათა მიმდევრობას.

- ალგორითმის თითოეული წესი უნდა იყოს მკაფიოდ ცალსახად განსაზღვრული, რის წყალობითაც ალგორითმის შესრულებას აქვს მექანიკური ხასიათი და არ მოითხოვს არავითარ დამატებით მითითებას ამოსახსნელი ამოცანის შესახებ.

- ალგორითმი უნდა იყოს შედეგიანი, რაც ნიშნავს, რომ

გარკვეული ეტაპების შესრულების შემდეგ უნდა იძლეოდეს ამონახსნს, ან თუ ამონახსნი არა აქვს სასრული რაოდენობის ბიჯების შემდეგ გაჩერდეს და გამოსცეს შესაბამისი შეტყობინება, ან გაგრძელდეს რაღაც მოცემული დროის განმავლობაში და გამოიტანოს საშუალედო შედეგები.

- ალგორითმი უნდა იყოს მასობრივი (განზოგადოებული) ე.ი. მისი შესრულება უნდა იყოს შესაძლებელი არა მხოლოდ ერთი კონკრეტული ამოცანისათვის, არამედ ამოცანათა მთელი კლასისათვის, რომელთაც აქვთ სხვადასხვა საწყისი მონაცემები.

1.3. ალგორითმის ჩაწერის ფორმები

ალგორითმი შეიძლება ჩაიწეროს სხვადასხვა ფორმით:

- სიტყვიერი, რაც ნიშნავს ალგორითმის ჩაწერას ბუნებრივ სალაპარაკო ენაზე;

- გრაფიკული, ეს არის ალგორითმის ჩაწერა გრაფიკული გამოსახულებების სახით.

- ფსევდოკოდები, ალგორითმის ნახევრად ფორმალიზებული აღწერაა პირობით ალგორითმულ ენაზე, რომელიც შეიცავს როგორც პროგრამირების ენის ელემენტებს, ასევე ბუნებრივი ენის ფრაზებს, საზოგადოდ მიღებულ მათემატიკურ აღნიშვნებს და სხვა.

- პროგრამული, ეს არის ალგორითმის წარმოდგენა პროგრამირების ენაზე.

ალგორითმის ჩაწერის სიტყვიერი ხერხი წარმოადგენს

ამოცანის ამოხსნის ეტაპების მიმდევრობის აღწერას ბუნებრივ ენაზე.

მაგალითისათვის შევადგინოთ სიტყვიერი ალგორითმი, რომელიც იპოვის ორი ნატურალური რიცხვის უდიდეს საერთო გამყოფს. ალგორითმის მიმდევრობა შეიძლება ჩაიწეროს შემდეგი სახით:

1. განსაზღვრეთ ორი რიცხვი.

2. თუ რიცვები ტოლია, მაშინ აიღეთ ნებისმიერი მათგანი შედეგის სახით და გაჩერდით, წინააღმდეგ შემთხვევაში განაგრძეთ ალგორითმის შესრულება.

3. განსაზღვრეთ უდიდესი ამ რიცხვთაგან.

4. შეცვალეთ უდიდესი, უდიდესსა და უმცირეს შორის სხვაობით.

5. გაიმეორეთ ალგორითმი მე-2 ბიჯიდან.

აღწერილი ალგორითმი მართებულია ნებისმიერი ნატურალური რიცხვებისათვის. ალგორითმის აღწერის სიტყვიერი ხერხი არ არის ფართოდ გავრცელებული, რადგან ასეთი აღწერა:

- არ არის მკაცრად ფორმალიზებული;

- ხასიათდება ჩანაწერის მრავალსიტყვიანობით;

- ცალკეული ჩანაწერი ხშირად არაცალსახად არის განსაზღვრული.

გრაფიკული ხერხი მეტად კომპაქტურია და ვიზუალურად კარგად აღიქმება.

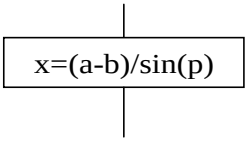
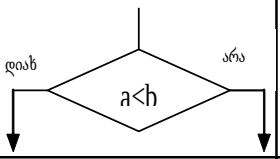
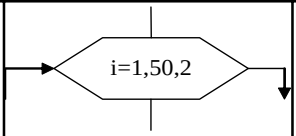
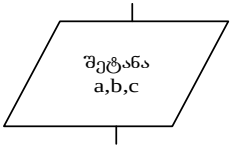
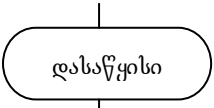
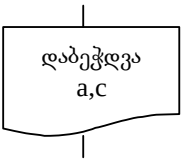
გრაფიკული წარმოდგენისას ალგორითმი გამოისახება ერთმანეთთან დაკავშირებული ფუნქციონალური ბლოკების მიმდევრობის სახით, რომელთაგან თითოეული შეესაბამება ერთ ან რამოდენიმე მოქმედების შესრულებას. ასეთ წარმოდგენას ეწოდება ალგორითმის სქემა ანუ ბლოკ-სქემა.

ბლოკ-სქემაში მოქმედების თითოეულ ტიპს (საწყისი მონაცემების შეტანას, გამოსახულების მნიშვნელობის გამოთვლას, პირობის შემოწმებას, მოქმედებათა განმეორების მართვას, დამუშავების დამთავრებას და ა.შ.) შეესაბამება გეომეტრიული ფიგურა, რომელიც წარმოდგენილია ბლოკური სიმბოლოს სახით. ბლოკური სიმბოლოები შეერთებულია გადასვლის ხაზებით, რომლებიც განსაზღვრავენ მოქმედებათა შესრულების რიგითობას. ნახ. 1.1.-ზე, ცხრილში მოყვანილია შედარებით ხშირად გამოყენებული სიმბოლოები.

ბლოკი - „პროცესი“ გამოიყენება მოქმედების ან მოქმედებათა მიმდევრობის აღსანიშნავად, რომლებიც იცვლიან მნიშვნელობას, წარმოდგენის ფორმას ან მონაცემთა განთავსებას. სქემის ვიზუალობის გასაუმჯობესებლად დამუშავების რამოდენიმე ცალკეული ბლოკი შეიძლება გაერთიანდეს ერთ ბლოკად. ცალკეული ოპერაციების წარმოდგენა საკმაოდ შეუზღუდავია.

ბლოკი - „გადაწყვეტა“ გამოიყენება პირობის მიხედვით მართვაზე გადასვლის აღსანიშნავად. თითოეულ ასეთ ბლოკში უნდა იყოს ნაჩვენები კითხვა, პირობა ან შედარება, რომლის

მიხედვითაც ხდება გადაწყვეტილების მიღება.

ბლოკის დასახელება	აღნიშვნა, შევსება	განმარტება
პროცესი		მოქმედებების მიმდევრობა
გადაწყვეტა		პირობის შემოწმება
მოდიფიკაცია		ციკლის დასაწყისი
წინასწარგანსაზღვრული პროცესი პარამეტრების გამოთვლა		გამოთვლები ქვეპროგრამებში
შეტანა-გამოტანა		შეტანა-გამოტანა ზოგადი სახით
დასაწყისი-დასასრული		ალგორითმის დასაწისი, დასასრული. ქვეპროგ-რამაში შესასვლელი, გამოსასვლელი
დოკუმენტი		შედეგების გამოტანა საბეჭდ მოწყობი-ლობაზე

ნახ. 1.1.

ბლოკი - „მოდიფიკაცია“ გამოიყენება ციკლური კონსტრუქციის ორგანიზებისათვის. (მოდიფიკაცია ნიშნავს სახის შეცვლას, გარდაქმნას). ბლოკის შიგნით ჩაიწერება ციკლის პარამეტრი, რომლისთვისაც მიეთითება საწყისი მნიშვნელობა, საბოლოო მნიშვნელობა და პარამეტრის ცვლილების ბიჯი თითოეული განმეორებისათვის.

ბლოკი - „წინასწარ განსაზღვრული პროცესი“ გამოიყენება დამხმარე ალგორითმებზე, ქვეპროგრამებზე მიმართვის გამოსახვისათვის.

ფსევდოკოდი წარმოადგენს აღნიშვნებისა და წესების სისტემას, რომელთა დანიშნულებაა ალგორითმის ერთგვაროვანი ჩაწერა.

ფსევდოკოდს შუალედური ადგილი უჭირავს ბუნებრივ და ფორმალურ ენებს შორის. ერთის მხრივ, იგი ახლოს არის ჩვეულებრივ ბუნებრივ ენასთან, ამიტომ ალგორითმი ამ სახით ჩაიწერება და წაიკითხება როგორც ჩვეულებრივი ტექსტი. მეორეს მხრივ, ფსევდოკოდში გამოიყენება ზოგიერთი ფორმალური კონსტრუქცია და მათემატიკური სიმბოლიკა, რომელიც ალგორითმის ჩაწერას აახლოვებს საზოგადოდ მიღებულ მათემატიკურ ჩანაწერთან.

ფსევდოკოდებში არ არის მიღებული მკაცრი სინტაქსური წესები ბრძანებების ჩასაწერად. თუმცა ფსევდოკოდში არის რამოდენიმე კონსტრუქცია, ფორმალური ენის მსგავსად, რაც აიოლებს ფსევდოკოდებიდან ალგორითმის გადატანას

ფორმალურ ენაზე.

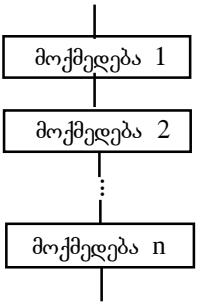
ალგორითმის სიტყვიერი ფორმით, ბლოკ-სქემის სახით ან ფსევდოკოდებით ჩაწერისას დასაშვებია განსაზღვრული თავისუფლება ბრძანებათა გამოსახვისათვის. აქ მნიშვნელოვანია ჩაწერის როგორი სტილი იქნება უფრო ზუსტი, რაც ადამიანს გაუადვილებს საქმის არსში ჩაწვდომას და ალგორითმის შესრულებას. რადგან პრაქტიკულად ალგორითმის შემსრულებელი არის კომპიუტერი, ამდენად ალგორითმი, რომელიც გათვალისწინებულია კომპიუტერისათვის უნდა იყოს დაწერილი მისთვის გასაგებ ენაზე. ამ დროს წინა პლანზე გადადის ბრძანებათა ზუსტი ჩაწერა. შესაბამისად, ალგორითმების ჩაწერის ენა უნდა იყოს ფორმალიზებული. ასეთ ენას ეწოდება - პროგრამირების ენა, ხოლო ამ ენაზე დაწერილ ალგორითმს - პროგრამა კომპიუტერისათვის.

1.4. საბაზო ალგორითმული სტრუქტურები

ალგორითმები შეიძლება წარმოვადგინოთ როგორც ცალკეული საბაზო (ე.ი. ძირითადი) ელემენტებისაგან შედგენილი სტრუქტურები. ბუნებრივია, ალგორითმების კონსტრუირების ძირითადი პრინციპების შესწავლა აუცილებელია დაიწყოს ამ საბაზო ელემენტების შესწავლით. ნებისმიერი ალგორითმის ლოგიკური სტრუქტურა შეიძლება იქნას წარმოდგენილი სამი საბაზო სტრუქტურის კომბინაციით: მიმდევრობა, განშტოება, ციკლი.

საბაზო სტრუქტურებისათვის დამახასიათებელია მათში ერთი შესასვლელისა და ერთი გამოსასვლელის არსებობა.

1. საბაზო სტრუქტურა „მიმდევრობა“ ხორციელდება ერთმანეთის მომდევნო მოქმედებათა მიმდევრობით:



ნახ. 1.2.

2. საბაზო სტრუქტურა „განშტოება“ პირობის შემოწმების შედეგის მიხედვით („დიახ“ ან „არა“) უზრუნველყოფს ალგორითმის მუშაობის ერთ-ერთი ალტერნატიული გზის არჩევას. თითოეულ გზას მივყავართ საერთო გამოსასვლელთან, ასე რომ ალგორითმის მუშაობა გაგრძელდება იმის მიუხედავად, თუ რომელი გზა იქნა არჩეული. განშტოების სტრუქტურა გადმოიცემა ოთხი ძირითადი ვარიანტის სახით:

1) თუ-მაშინ;

2) თუ-მაშინ-წინააღმდეგ შემთხვევაში;

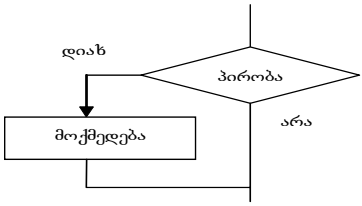
3) ამორჩევა;

4) ამორჩევა-წინააღმდეგ შემთხვევაში;

განვიხილოთ თითოეული ვარიანტი.

თუ - მაშინ

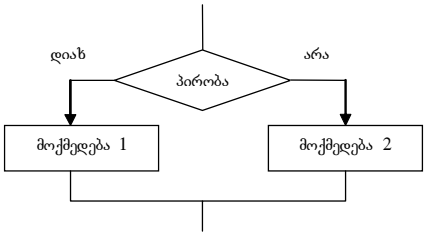
თუ პირობა
 მაშინ მოქმედება
დასრულება



ნახ. 1.3.

თუ - მაშინ - წინააღმდეგ შემთხვევაში

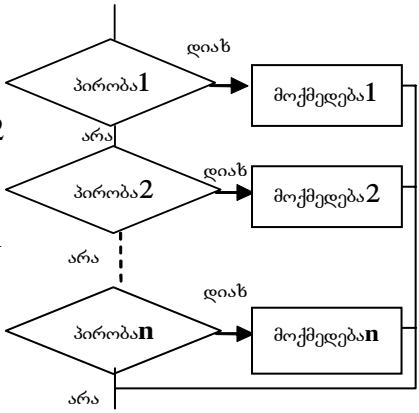
თუ პირობა
 მაშინ მოქმედება 1
  წ.შ. მოქმედება 2
დასრულება



ნახ. 1.4.

ამორჩევა

ამორჩევა
 პირობა 1 : მოქმედება 1
 პირობა 2 : მოქმედება 2
 . . . . . .
 პირობა n : მოქმედება n
დასრულება



ნახ. 1.5.

ამორჩევა-წინააღმდეგ შემთხვევაში

ამორჩევა

პირობა 1 : მოქმედება 1

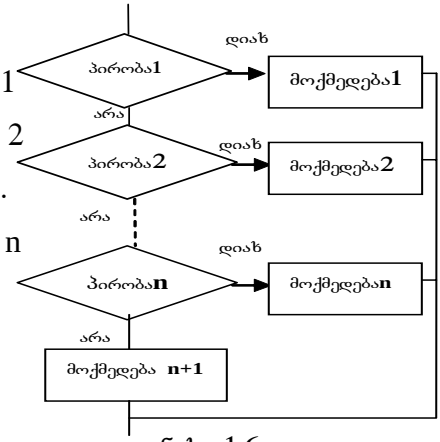
პირობა 2 : მოქმედება 2

. . . . . .

პირობა n : მოქმედება n

წ.შ. მოქმედება n+1

დასრულება



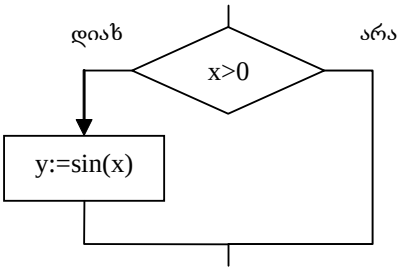
ნახ. 1.6.

განშტოების მაგალითები.

თუ x>0

მაშინ y:=sin(x)

დასრულება



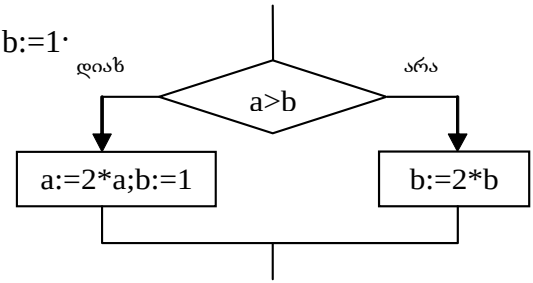
ნახ. 1.7.

თუ a>b

მაშინ a:=2*a;b:=1·

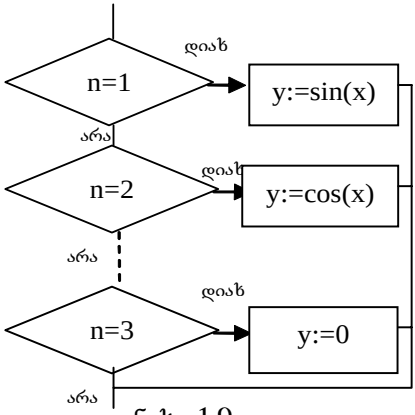
წ.შ. b:=2*b;

დასრულება



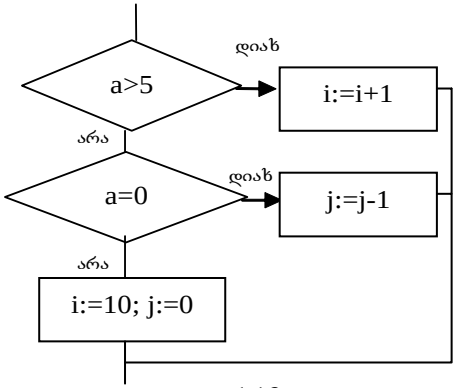
ნახ. 1.8.

ამორჩევა
თუ n=1 : y:=sin(x)
თუ n=2 : y:=cos(x)
თუ n=3 : y:=0
დასრულება



ნახ. 1.9.

ამორჩევა
თუ a>5 : i:=i+1
თუ a=0 : j:=j+1
წ.შ. i:=10; j:=0
დასრულება



ნახ. 1.10.

3. საბაზო სტრუქტურა „ციკლი“ უზრუნველყოფს მოქმედებათა ერთობლიობის, რომელსაც შეიძლება ვუწოდოთ ციკლის „ტანი“, მრავალჯერად შესრულებას. ციკლი შეიძლება იყოს სხვადასხვა სახის.

ციკლი „სანამ“ ასრულებს ციკლის „ტანს“ მანამ, სანამ შესრულდება პირობა, რომელიც იწერება „სანამ“ სიტყვის შემდეგ.

ციკლის დასაწყისი

სანამ პირობა

ციკლის „ტანი“

(მოქმედებათა მიმდევრობა)

ციკლის დასასრული

ნახ. 1.11.

ციკლი „მოცემული დიაპაზონისათვის“, უზრუნველყოფს ციკლის „ტანის“ შესრულებას ცვლადის (ციკლის პარამეტრის) ყველა იმ მნიშვნელობისათვის, რომელიც მოთავსებულია მოცემულ დიაპაზონში.

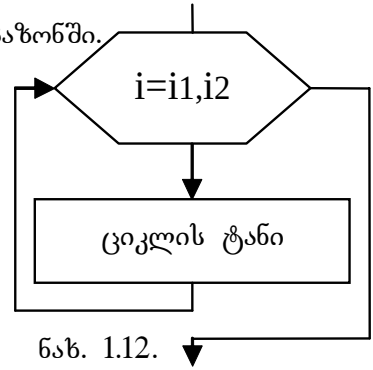
ციკლის დასაწყისი

i-სათვის i1-დან i2-მდე

ციკლის „ტანი“

(მოქმედებათა მიმდევრობა)

ციკლის დასასრული



ნახ. 1.12.

ციკლური სტრუქტურების მაგალითები

ციკლის დასაწყისი

სანამ i<=5

S:=S+A[i]

i:=i+1

ციკლის დასასრული

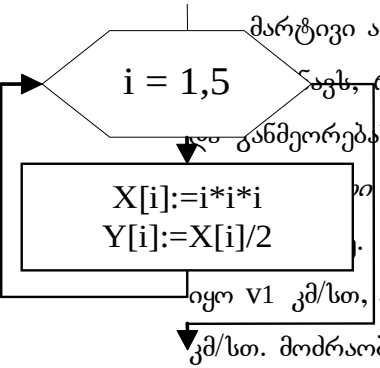
ნახ. 1.13.

ციკლის დასაწყისი
i-სათვის 1-დან 5-მდე
X[i]:=i*i*i
Y[i]:=X[i]/2
ციკლის დასასრული

ნახ. 1.14.

1.5. წრფივი, განშტოებადი და ციკლური სტრუქტურის ალგორითმების მაგალითები

წრფივი ალგორითმები.



მარტივი ამოცანები ამოიხსნება წრფივი ალგორითმებით, ვინაიდან, როგორც ვხედავთ, ისინი არ შეიცავენ პირობის შემოწმებას და განმეორებას.

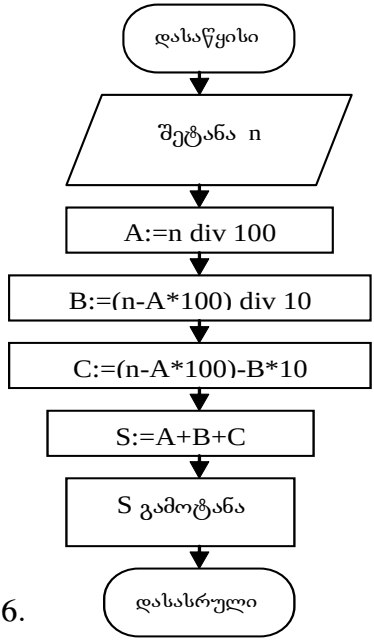
მაგალითი 1. მგზავრი მიდიოდა განსხვავებული რელიეფის გზით. მისი მოძრაობის სიჩქარე სწორ ადგილებში იყო v1 კმ/სთ, აღმართში - v2 კმ/სთ, ხოლო დაღმართში - v3 კმ/სთ. მოძრაობის დრო შესაბამისად იყო t1, t2, t3 საათი. რა გზა გაიარა მგზავრმა?(ნახ. 1.15.)

მაგალითი 2. მოცემულია ნატურალური სამნიშნა რიცხვი n, შეადგინეთ ალგორითმი, რომელიც გამოითვლის მოცემული რიცხვის შემადგენელ ციფრთა ჯამს. (ნახ. 1.16)

1. შეტანა v1,v2,v3,t1,t2,t3
2. S1:=v1*t1
3. S2:=v2*t2
4. S3:=v3*t3
5. S:=S1+S2+S3
6. S გამოტანა
7. დასასრული

ნახ. 1.15.

1. შეტანა n
2. A:=n div 100
3. B:=(n-A*100) div 10
4. C:=(n-A*100)-B*10
5. S:=A+B+C
6. S გამოტანა
7. დასასრული



ნახ. 1.16.

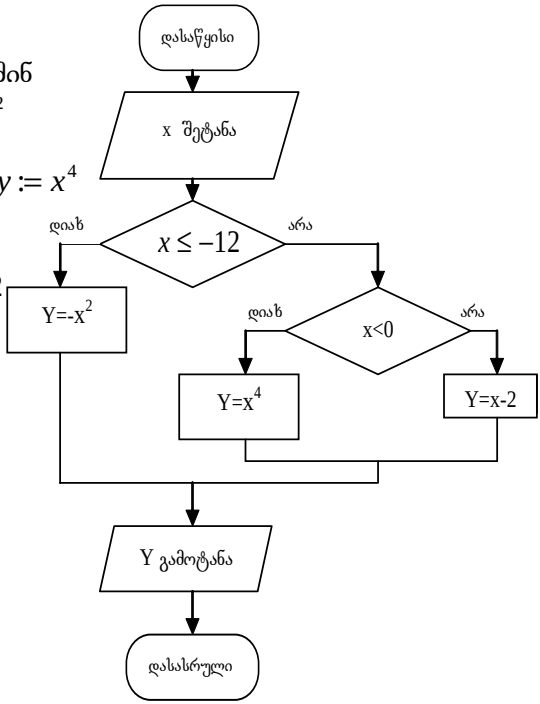
განშტოება.

ხშირად, ესა თუ ის მოქმედება უნდა შესრულდეს ლოგიკური გამოსახულების მნიშვნელობის მიხედვით. ასეთ შემთხვევებში გამოიყენება განშტოება.

მაგალითი 1. გამოითვალეთ ფუნქციის მნიშვნელობა:

$$y = \begin{cases} -x^2; & x \le -12, \\ x^4; & -12 < x < 0, \\ x - 2; & x \ge 0. \end{cases}$$

1. x შეტანა
2. თუ $x \le -12$, მაშინ $y := -x^2$
3. თუ x<0, მაშინ $y := x^4$
4. თუ $x \ge 0$, მაშინ y:=x-2
5. y გამოტანა
6. დასასრული

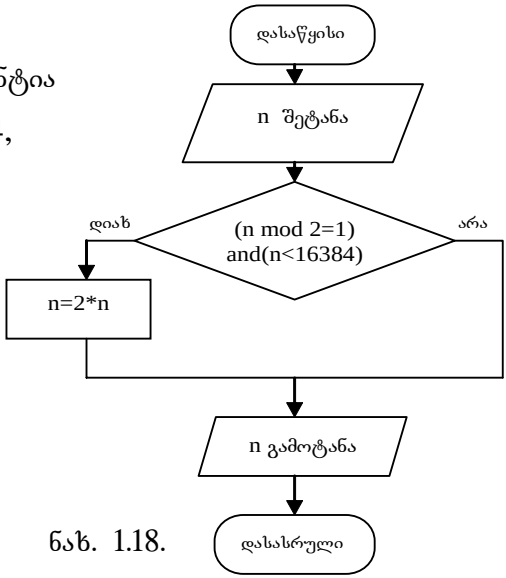


ნახ. 1.17.

განშტოებადი სტრუქტურის ალგორითმების ტესტირებისას აუცილებელია ისეთი საწყისი მონაცემების შერჩევა, რომ ყველა ტოტის შემოწმება იყოს შესაძლებელი. ზემოთ მოყვანილ მაგალითში უნდა გვქონდეს სულ მცირე სამი ტესტური ნაკრები.

მაგალითი 2. მოცემულია n ნატურალური რიცხვი. თუ რიცხვი კენტია და მისი გაორმაგების შედეგად იგი არ გადააჭარბებს 32767 (მთელი ტიპის ზედა ზღვარი), მაშინ გააორმაგეთ, წინააღმდეგ შემთხვევაში დატოვეთ უცვლელად. გაორმაგების პირობა რომ დაკმაყოფილდეს რიცხვი n უნდა იყოს კენტი და 16384-ზე ნაკლები.

1. n შეტანა
2. თუ რიცხვი n კენტია და ნაკლებია 16384, მაშინ n:=n*2
3. n გამოტანა
4. დასასრული



ნახ. 1.18.

მაგალითში ილუსტრირებულია არასრული განშტოება. ლოგიკური გამოსახულება (პირობა) შეიცავს ორ ოპერანდს.

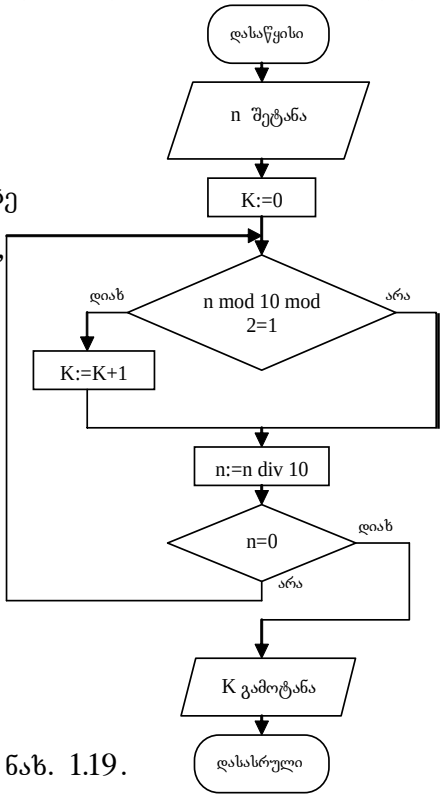
ციკლი.

იმ შემთხვევაში, როცა საჭიროა გარკვეული ოპერატორების რამოდენიმეჯერ შესრულება, უნდა მოხდეს ციკლის ორგანიზება.

მაგალითი 1. n ნატურალური რიცხვის ჩანაწერში დაითვალეთ კენტი ციფრების რაოდენობა.

მითითება. მოცემული რიცხვის უმცროსი თანრიგიდან აირჩიეთ ციფრები ერთმანეთის მიმდევრობით, სანამ იგი არ ამოიწურება, ე.ი. გახდება ნულის ტოლი. თითოეული კენტი ციფრი აღვრიცხოთ.

1. n რიცხვის შეტანა
2. K:=0 {მთვლელი}
3. თუ n=0, გადასვლა 7-ზე
4. თუ n mod 10 mod 2=1,
   მაშინ K:=K+1
5. n:=n div 10
6. გადასვლა 3-ზე
7. K გამოტანა
8. დასასრული



ნახ. 1.19.

მაგალითი 2. მოცემულია მიმდევრობა, რომლის ზოგადი წევრი გამოითვლება ფორმულით: $a_n = \frac{n-1}{n^2}$

n>2-სათვის გამოითვალეთ იმ წევრების ჯამი, რომლებიც მეტია მოცემულ R რიცხვზე.

1. R შეტანა
2. S:=0

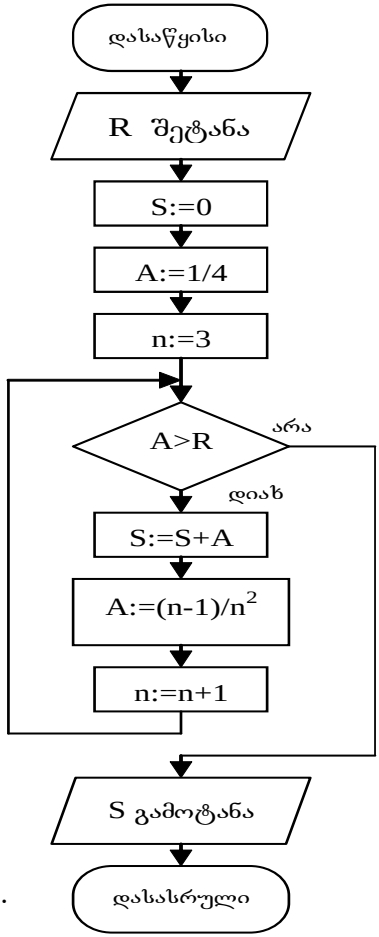
A:=1/4

4. n:=3

5 A შევადაროთ R-ს.

თუ A>=R, გადასვლა 10-ზე

6. S:=S+A
7. A:=(n-1)/(n*n)
8. n:=n+1
9. გადასვლა 5-ზე
10. S გამოტანა
11. დასასრული



ნახ. 1.20.

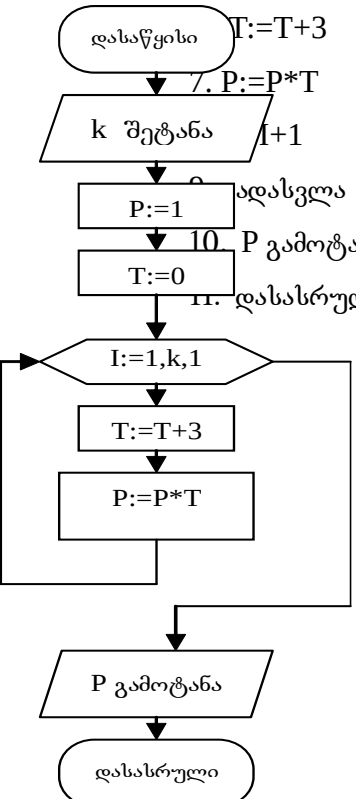
ზემოთ განხილულ მაგალითებში განმეორებათა რიცხვი წინასწარ არ არის ცნობილი. პირველ მაგალითში იგი დამოკიდებულია ნატურალური რიცხვის ჩანაწერში ციფრების

რაოდენობაზე, მეორეში კი - R რიცხვზე. იმ შემთხვევებში, როცა ბიჯების რაოდენობა ამოცანის პირობაში მოცემულია, უმჯობესია გამოვიყენოთ ციკლი პარამეტრით.

მაგალითი 3. გამოითვალეთ k რაოდენობის ნატურალური 3-ის ჯერადი რიცხვების ნამრავლი.

მითითება. გაითვალისწინეთ, რომ პირველი სამის ჯერადი რიცხვი არის 3, ხოლო დანარჩენები მასზე 3-ით მეტია.

1. k შეტანა
2. P:=1 {ითვლის ნამრავლს}
3. T:=0 {3-ის ჯერადი რიცხვები}
4. I:=1
5. თუ I>k, გადასვლა 10-ზე
6. T:=T+3
7. P:=P*T
8. I:=I+1
9. უპირობო გადასვლა 5-ზე
10. P გამოტანა
11. დასასრული



ნახ. 1.21.

1.6. ჩადგმული ციკლები

ხშირად, ამოცანებში საჭიროა ციკლის შიგნით ოპერატორების მიმდევრობის განმეორება, რაც მოითხოვს შიდა ციკლის ორგანიზებას. ასეთ სტრუქტურას შეიძლება ვუწოდოთ ციკლი ციკლში ანუ ჩადგმული ციკლი. ციკლის ჩადგმის სიღრმე (ანუ ერთმანეთში ჩადგმულ ციკლთა რაოდენობა) შეიძლება იყოს განსხვავებული.

ასეთი სტრუქტურის ორგანიზებისას, მანქანური დროის ეკონომიის მიზნით საჭიროა შიდა ციკლიდან გარე ციკლში იმ ოპერატორების გატანა, რომლებიც არ არიან დამოკიდებული შიდა ციკლის პარამეტრზე.

მაგალითი 1. ციკლი „მოცემული დიაპაზონისათვის“. გამოითვალეთ მოცემული A(5,3) მატრიცის ელემენტების ჯამი.

```
 S:=0;
ციკლის დასაწყისი i-სათვის
       1-დან  5-მდე
   ციკლის დასაწყისი j-სათვის
             1-დან  3-მდე
       S:=S+A[i,j]
   ციკლის დასასრული
ციკლის დასასრული
```

ნახ. 1.22.

მაგალითი 2. ციკლი „სანამ“

იპოვეთ მოცემული A(10,10) მატრიცის იმ ელემენტების ნამრავლი, რომლებიც განთავსებულნი არიან ლუწი სტრიქონებისა და ლუწი სვეტების გადაკვეთაზე.

i:=2; P:=1;

ციკლის დასაწყისი სანამ i<=10

j:=2

ციკლის დასაწყისი სანამ j<=10

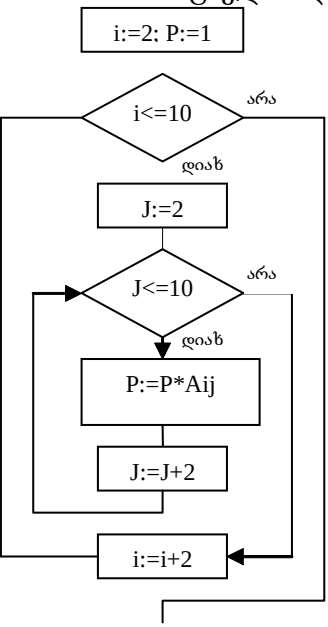
P:=P*A[i,j]

j:=j+2

ციკლის დასასრული

i:=i+2

ციკლის დასასრული



ნახ. 1.23.

1.7. პროგრამირების ენების დახასიათება

თანამედროვე პერიოდში არსებობს რამოდენიმე ასეული რეალურად გამოყენებული პროგრამირების ენა, რომელთაგან თითოეულს გააჩნია თავისი გამოყენების სფერო.

ნებისმიერი ალგორითმი, როგორც ვიცით არის სასრული რაოდენობის მოქმედებათა მიმდევრობა, რომელთა შესრულება საწყისი მონაცემებიდან ნაბიჯ-ნაბიჯ მიგვიყვანს შედეგებამდე. ჩანაწერები მეტყველებენ პროგრამირების ენის დონეზე, რაც უფრო ნაკლებია დეტალიზება, მით უფრო მაღალია პროგრამირების ენა.

ამ კრიტერიუმის მიხედვით შეიძლება გამოვყოთ პროგრამირების ენის დონეები:

- მანქანური;
- მანქანურ-ორიენტირებული(ასემბლერი);
- მანქანურ-დამოუკიდებელი(მაღალი დონის ენები).

მანქანური და მანქანურ-ორიენტირებული ენები - ეს არის დაბალი დონის ენები, რომლებიც მოითხოვენ მონაცემთა დამუშავების პროცესის წვრილმანი დეტალების ჩვენებას. მაღალი დონის ენები წარმოადგენენ ბუნებრივი ენის იმიტაციას, იყენებენ სალაპარაკო ენის ზოგიერთ სიტყვას და საზოგადოდ მიღებულ მათემატიკურ სიმბოლოებს. ეს ენები მეტად მოხერხებულია ადამიანებისათვის.

მაღალი დონის ენები იყოფიან:

- პროცედურული (ალგორითმული)-Basic, Pascal, C და

სხვა - რომლებიც ცალსახად აღწერენ ალგორითმებს. ამოცანის ამოსახსნელად პროცედურული ენები მოითხოვენ ამოხსნის პროცედურის ამა თუ იმ ფორმაში ცხადად ჩაწერას.

ლოგიკური (Prolog, Lisp და სხვა), რომლებიც ორიენტირებულნი არიან არა ამოცანის ამოხსნის ალგორითმის დამუშავებაზე, არამედ ამოცანის სისტემატურ და ფორმალიზებულ აღწერაზე.

- ობიექტურ-ორიენტირებული (Object Pascal, C++, Java და სხვა), რომელთაც საფუძვლად უდევთ ობიექტის ცნება, რომელთანაც შერწყმულია მონაცემები და მათზე მოქმედებები.

თითოეულ კომპიუტერს აქვს თავისი მანქანური ენა, ანუ მანქანური ბრძანებების თავისი ერთობლიობა, რომელიც განსხვავდება ბრძანებაში მისამართების რაოდენობით, მისამართებში მოცემული ინფორმაციის დანიშნულებით, მანქანის მიერ შესასრულებელი ოპერაციების ნაკრებით და სხვა.

მანქანურ ენაზე პროგრამირებისას პროგრამისტი უნდა აკონტროლებდეს თითოეულ ბრძანებას და მეხსიერების თითოეულ უჯრედს, უნდა გამოიყენოს მანქანური ოპერაციების ყველა შესაძლებლობა.

მანქანურ ენაზე პროგრამის დაწერა ძალიან შრომატევადია. პროგრამაც, შესაბამისად დიდი მოცულობისაა, რთულია მისი გაწყობა, შეცვლა და განვითარება. ამიტომაც ეფექტური პროგრამის მისაღებად, მაქსიმალურად ითვალისწინებენ რა კონკრეტული კომპიუტერის სპეციფიკას, მანქანური ენების

ნაცვლად გამოიყენება მათთან მიახლოვებული მანქანურ-ორიენტირებული ენები.

ალგორითმული ენების უპირატესობა მანქანურ ენებთან შედარებით შემდგომში მდგომარეობს:

- ალგორითმული ენების ანბანი გაცილებით ფართოა მანქანურთან შედარებით, რომელიც გაცილებით აიოლებს პროგრამის ტექსტის წაკითხვას;

- ოპერაციების ნაკრები არ არის დამოკიდებული მანქანური ოპერაციების ნაკრებთან, იგი აირჩევა განსაზღვრული კლასის ამოცანების ამოხსნის ალგორითმის ფორმულირებიდან გამომდინარე;

- წინადადებათა ფორმატი საკმაოდ მოქნილია და მოხერხებულია გამოყენების თვალსაზრისით, რაც საშუალებას იძლევა ერთი წინადადების მეშვეობით გადმოიცეს მონაცემთა დამუშავების გარკვეული მოცულობა;

- საჭირო ოპერაციები გამოსახულია საზოგადოდ მიღებული მათემატიკური აღნიშვნებით;

- ალგორითმულ ენებში მონაცემებს ენიჭებათ ინდივიდუალური სახელები, რომელთაც ირჩევს პროგრამისტი;

- ენაში გათვალისწინებულია მონაცემთა ტიპების მნიშვნელოვნად ფართო ნაკრები,მანქანურ მონაცემთა ტიპების ნაკრებთან შედარებით.

ასე რომ, ალგორითმული ენები არიან მანქანურ-დამოუკიდებელი. ისინი პროგრამისტს უმსუბუქებენ მუშაობას

და მაღლებენ შექმნილი პროგრამის საიმედოობას.

ალგორითმულ ენას, ისე როგორც სხვა ნებისმიერ ენას, გააჩნია: ანბანი, სინტაქსი და სემანტიკა.

ანბანი - ეს არის მოცემული ენისათვის ძირითადი სიმბოლოების ნაკრები, ანუ „ანბანის ასოები“, რომლისაგან შედგება ამ ენის ნებისმიერი ტექსტი.

სინტაქსი - ეს არის ფრაზის აგების წესი, რომელიც განსაზღვრავს სწორად არის თუ არა დაწერილი ესა თუ ის ფრაზა. უფრო ზუსტად, ენის სინტაქსი მოიცავს წესების ნაკრებს, რომელიც განსაზღვრავს სიმბოლოების რომელი კომბინაციაა გააზრებული წინადადება ამ ენაზე.

სემანტიკა განსაზღვრავს წინადადების აზრობრივ მნიშვნელობას. სემანტიკა ადგენს ენის ამა თუ იმ ფრაზებით მოქმედებათა რომელი მიმდევრობაა აღწერილი და საბოლოოდ, განსაზღვრავს მოცემული ტექსტით რომელი ალგორითმია მოცემული.

1.8. ალგორითმული ენის ზოგიერთი ცნება

ენის ცნება განისაზღვრება სინტაქსურ და სემანტიკურ წესებთან ერთობლიობაში. ალგორითმულ ენაში გვხვდება შემდეგი ძირითადი ცნებები:

1. სახელები (იდენტიფიკატორი) - გამოიყენება პროგრამის ობიექტების აღნიშვნისათვის (ცვლადები, ფუნქციები, მასივები და სხვა).

2. ოპერაციები. ოპერაციათა ტიპებია:

- არითმეტიკული ოპერაციები: +, -, *, / ;

- ლოგიკური ოპერაციები: და, ან, არა;

- დამოკიდებულების ოპერაციები: <, >, <=, >=, =, <>;

- დაკავშირების ოპერაციები („შეერთება“, „კონკატენაცია“), რომელიც აერთიანებს სიმბოლოებს სტრიქონად, გამოისახება „+“-ით.

3. მონაცემები - სიდიდეებია, რომლებსაც ამუშავებთ პროგრამები. ძირითადად, მონაცემები გვაქვს სამი სახის: მუდმივები, ცვლადები და მასივები.

- *მუდმივები*, ეს არის ფიქსირებული მონაცემები, რომლებიც არ იცვლებიან პროგრამის შესრულების პროცესში.

მუდმივების მაგალითები:

რიცხვითი: 75, 12;

ლოგიკური: დიახ (ჭეშმარიტი), არა (მცდარი);

სიმბოლური (შეიცავს ერთ სიმბოლოს): „A“, „+“;

ლიტერული (შეიცავს სიმბოლოს ნებისმიერ რაოდენობას) „a0“, „White“, ““(ცარიელი სტრიქონი).

- *ცვლადები* აღინიშნებიან სახელებით და შეუძლიათ მნიშვნელობის შეცვლა პროგრამის შესრულებისას. ცვლადები არის მთელი ნამდვილი, ლოგიკური, სიმბოლური და ლიტერული.

- *მასივები* - ფიქსირებული რაოდენობის ერთნაირი ტიპის ელემენტების მიმდევრობაა, რომელთაც აქვთ ერთი სახელი. ელემენტის მდებარეობა მასივში ცალსახად განისაზღვრება

მისი ინდექსით. ზოგჯერ მასივებს ცხრილებსაც უწოდებენ.

4. გამოსახულება, რომელთა დანიშნულებაა საჭირო გამოთვლების შესრულება. იგი შედგება მუდმივების, ცვლადების, ფუნქციის მაჩვენებლებისაგან (მაგალითად, exp(x)), რომლებიც გაერთიანებულნი არიან ოპერაციის ნიშნებით.

გამოსახულება ჩაიწერება სიმბოლოების წრფივი მიმდევრობით, რაც კლავიატურის შესაბამის ღილაკებზე მიმდევრობით დაჭერით მისი კომპიუტერში შეტანის საშუალებას იძლევა.

განასხვავებენ არითმეტიკულ, ლოგიკურ და სტრიქონულ გამოსახულებებს.

- არითმეტიკული გამოსახულებაში განისაზღვრება ერთი რიცხვითი მნიშვნელობა. მაგალითად, (1+sin(x))/2. ამ გამოსახულების მნიშვნელობა როცა x=0, უდრის 0.5, ხოლო როცა x=p/2 უდრის 1.

- ლოგიკური გამოსახულება აღწერს რაღაც პირობას, რომელიც კმაყოფილდება ან არ კმაყოფილდება. ამიტომ, ლოგიკურ გამოსახულებას შეიძლება ჰქონდეს ორი მნიშვნელობა - ჭეშმარიტი ან მცდარი (დიახ ან არა). მაგალითის სახით განვიხილოთ ლოგიკური გამოსახულება x*x+y*y<r*r, რომელიც განსაზღვრავს წერტილის, (x,y) კოორდინატებით, კუთვნილებას r რადიუსიან წრის მიმართ, რომლის ცენტრია კოორდინატთა სათავე. როცა x=1, y=1, r=2 ამ გამოსახულების მნიშვნელობა ,,ჭეშმარიტია‘‘, ხოლო

როცა x=2, y=2, r=1 - „მცდარია“.

- სტრიქონულ (ლიტერულ) გამოსახულებაში მნიშვნელობა არის ტექსტი. სტრიქონულ გამოსახულებაში შეიძლება შედიოდეს ლიტერული და სტრიქონული მუდმივები, ლიტერული და სტრიქონული ცვლადები, ლიტერული ფუნქციები, რომლებიც გამოყოფილია შეჭიდების ოპერაციით. მაგალითად, A+B აღნიშნავს A სტრიქონის ბოლოში B სტრიქონის შეერთებას. მაგალითად A=”პერსონალური “, A=”კომპიუტერი“, მაშინ A+B გამოსახულების მნიშვნელობა იქნება „პერსონალური კომპიუტერი“.

5. ოპერატორები (ბრძანებები). ოპერატორი - ეს ენის მეტად მსხვილი და შინაარსობრივი ცნებაა: თითოეული ოპერატორი წარმოადგენს ენის დასრულებულ ფრაზას და განსაზღვრავს მონაცემთა დამუშავების სავსებით დასრულებულ ეტაპს. ოპერატორების შემადგენლობაში შედიან:

- გასაღებური სიტყვები;
- მონაცემები;
- გამოსახულებები და ა.შ.

ოპერატორები შეიძლება იყოს შესრულებადი და არაშესრულებადი. არაშესრულებადი ოპერატორების დანიშნულებაა მონაცემთა და პროგრამის სტრუქტურის აღწერა, ხოლო შესრულებადი ოპერატორების - სხვადასხვა მოქმედებების შესრულება (მაგალითად მინიჭების ოპერატორი, შეტანის და გამოტანის ოპერატორები, პირობის ოპერატორი,

ციკლის ოპერატორი, პროცედურის ოპერატორი და სხვა).

კომპიუტერში სხვადასხვა ამოცანების ამოხსნისას საჭიროა ლოგარითმის ან რიცხვის მოდულის, კუთხის სინუსის გამოთვლა და ა.შ.

ხშირად გამოყენებადი მოქმედებების გამოთვლა ხორციელდება ქვეპროგრამის მეშვეობით, რომელთაც ეწოდებათ სტანდარტული ფუნქციები, რომლებიც წინასწარ არიან დაპროგრამებული და ენის ტრანსლატორში ჩაშენებული.

ცხრილში მოყვანილია ზოგიერთი სტანდარტული ფუნქცია, თუმცა პროგრამირების თითოეულ ენას გააჩნია თავისი სტანდარტული ფუნქციები.

ფუნქციის დასახელება და მათემატ. აღნიშვნა		ფუნქც.ჩაწერა
აბსოლუტური სიდიდე (მოდული)	$\lvert x \rvert$	abs(x)
კვადრატში ახარისხება, კვადრატული ფესვი	x^2 $\sqrt{x}$	sqr(x), sqrt(x)
ნატურალური ლოგარითმი	$\ln x$	ln(x)
ათობითი ლოგარითმი	lg	lg(x)
ექსპონენტა	e^x	exp(x)
განაყოფის მთელი ნაწილი		x div y
გაყოფისას მიღებული ნაშთი		x mod y
კუთხის სინუსი (კუთხე რადიანებში)		sin(x)
კუთხის კოსინუსი		cos(x)
კუთხის ტანგენსი		tg(x)
კუთხის კოტანგენსი		ctg(x)

არითმეტიკული გამოსახულება ჩაიწერება შემდეგი წესის მიხედვით:

- თანამამრავლთა შორის არ შეიძლება გამრავლების ნიშნის გამოტოვება, ასევე არ შეიძლება ოპერაციის ორი ნიშნის ერთმანეთის გვერდით ჩასმა.

- მასივის ელემენტების ინდექსები ჩაიწერება კვადრატულ ან მრგვალ ფრჩხილებში.

- ცვლადების აღსანიშნავად გამოიყენება ლათინური ანბანის ასოები.

- ოპერაციები სრულდება პრიორიტეტების დაცვით: ჯერ ფუნქციების გამოთვლა, შემდეგ ახარისხება, შემდეგ გამრავლება-გაყოფა, ხოლო ბოლოს შეკრება-გამოკლება.

- ერთნაირი პრიორიტეტის ოპერაციები სრულდება მარცხნიდან მარჯვნივ.

არითმეტიკული გამოსახულებების ჩაწერის მაგალითები:

მათემატიკ.ჩაწერა	ჩაწერა ალგორითმულ ენაზე
$\frac{xy}{z}$	x * y / z
$\frac{x}{yz}$	x / (y * z)
$\frac{a^2 + b^2}{bc}$	(a * a + b * b) / (b * c)
$\frac{a_{i+1} + b_{i-1}}{2xy}$	(a[i+1] + b[i-1]) / (2*x*y)
$\frac{-b + \sqrt{b^2 - 4ac}}{2a}$	(-b + sqrt (b*b - 4*a*c)) / (2*a)
$0{,}49e^{a^2-b^2} + \ln\cos a^2$	0.49*exp(a*a-b*b) + ln(cos(a*a))

ლოგიკურ გამოსახულებებში გარდა არითმეტიკული ოპერაციებისა, გამოიყენება დამოკიდებულების ოპერაციები: <(ნაკლებია), >(მეტია), <= (ნაკლებია ან ტოლი), >=(მეტია ან ტოლი), =(ტოლია), <>(არ უდრის).

ლოგიკური გამოსახულების ჩაწერის მაგალითები:

პირობა	ჩაწერა ალგორითმ.ენაზე
a მთელი რიცხვი - ლუწია	a mod 2 = 0
a მთელი რიცხვი - კენტია	a mod 2 = 1
a და b დადებითი რიცხვებია	(a>0) and (b>0)
a,b-დან ერთერთია დადებითი	(a>0) or (b>0)
a,b,c-დან ერთერთია უარყოფითი	(a<0) or (b<0) or (c<0)
a<x<b	(x>a) and (x<b)
x მოთავსებულია შუალედში[1,3]	(x>=1) and (x<=3)
კვადრატულ განტოლ. არა აქვს ფესვები	b*b - 4*a*c<0
a და b ურთიერთსაწინააღმდეგოა	a=-b
a და b ურთიერთშებრუნებულია	a*b=1
a მეტია b,c,d საშ. არითმეტიკულზე	a>(b+c+d)/3
წერტილი, რომლის კოორდინატებია (x,y) მდებარეობს პირველ ან მეოთხე მეოთხედში	((x>0) and (y>0)) or ((x<0) and (y>0))

II თავი

მონაცემთა ტიპები და საბაზო მონაცემთა სტრუქტურები

2.1. მონაცემთა ტიპის ცნება

ტიპები და მონაცემთა სტრუქტურები წარმოადგენენ საფუძველს, რომელზედაც აგებულია თანამედროვე პროგრამირების ტექნოლოგია. ტიპიზაციის ძირითად პრინციპს წარმოადგენს ის, რომ ნებისმიერი მუდმივა, ცვლადი, გამოსახულება და ფუნქცია მიეკუთვნება რომელიმე ტიპს, რაც საშუალებას იძლევა, რომ მონაცემის შესანახად კომპიუტერში გამოიყოს მეხსიერების ზუსტად იმდენი მოცულობა, რომელიც განისაზღვრება ტიპის მნიშვნელობის დასაშვები დიაპაზონით. თუმცა ტიპის კონცეფცია ამით არ ამოიწურება.

შესაძლებელია მონაცემთა ტიპების სხვადასხვა კლასიფიკაცია, მაგალითად, მარტივი და რთული ტიპები.

გამოვყოთ ტიპების შემდეგი კატეგორიები:

1. ჩადგმული მონაცემთა ტიპები ანუ ტიპები, რომლებიც წინასწარ არის განსაზღვრული პროგრამირების ენაში. ჩვეულებრივ, ენაში ფიქსირდება ამ ტიპების მნიშვნელობების გარე წარმოდგენა და ოპერაციების ნაკრები მათი სემანტიკის აღწერით. შიდა წარმოდგენა და ოპერაციის რეალიზება შეირჩევა კონკრეტულ კომპილატორში.

2. ჩამოთვლითი ტიპის კატეგორიას მიეკუთვნება ცხადად

განსაზღვრული მთელი ტიპები, მნიშვნელობათა სასრული რაოდენობით.

3. კონსტრუირებული (შედგენილი) ტიპები, რომელთა შორის ყველაზე გავრცელებულია: მასივები, ჩანაწერები და სიმრავლე.

4. მიმთითებლის ტიპი, რომლის მნიშვნელობა რომელიმე პროგრამული ობიექტის მისამართია.

2.2. ჩადგმული მონაცემთა ტიპები

თანამედროვე კომპიუტერებში ასეთ ტიპებს მიეკუთვნება სხვადასხვა ზომის მთელი ტიპები(1-დან 8-ბაიტამდე),ბულის მნიშვნელობა (გამოიყენება, ძირითადად, პირობის მიხედვით მართვის გადაცემის შემთხვევებში) და მცურავმძიმიანი რიცხვები ერთმაგი ან ორმაგი სიზუსტით (შესაბამისად, ოთხი ან რვა ბაიტი).

მთელრიცხვა ტიპს მიეკუთვნება:

Integer, შიდა წარმოდგენაში იკავებს 2 ბაიტს, მნიშვნელობათა დიაპაზონია -32768-დან +32767;

Long Integer, შიდა წარმოდგენაში იკავებს 4 ბაიტს, მნიშვნელობათა დიაპაზონია -2147483648-დან 2147483647;

პროგრამირების სხვადასხვა ენაში გამოიყენება ასევე:

Byte - მნიშვნელობათა დიაპაზონია 0-დან 255-მდე;

Word- მნიშვნელობათა დიაპაზონია 0-დან 65535-მდე;

ShortInt- მნიშვნელობათა დიაპაზონია -128-დან 127-

l

მდე;

Cardinal- მნიშვნელობათა დიაპაზონია 0-დან 2147483647-მდე;

ნამდვილრიცხვა (მცურავმძიმიანი) ტიპებიდან სხვადასხვა ენაში გვხვდება:

Float, შიდა წარმოდგენაში იკავებს 4 ბაიტს, მნიშვნელობათა დიაპაზონია -10^{38}-დან 10^{38}-მდე;

Real, შიდა წარმოდგენაში იკავებს 6 ბაიტს, მნიშვნელობათა დიაპაზონია 2,9E-39 დან 1,7E+38-მდე;

Double (ნამდვილრიცხვა ორმაგი სიზუსტით), შიდა წარმოდგენაში იკავებს 8 ბაიტს, მნიშვნელობათა დიაპაზონია -10^{307}-დან 10^{307}-მდე.

ჩადგმულ მონაცემთა ტიპებს მიეკუთვნება ტიპი Character (ანუ Char) სხვადასხვა ენებში ეს არის: ანბანის სიმბოლოების ნაკრები ან ერთიანების და ნულების ნაკრები, რომელიც იკავებს ერთ ბაიტს. თუ x ცვლადი აკმაყოფილებს პირობას: 0 <= x <= 9, მაშინ მისი მნიშვნელობა არის ციფრი, ხოლო თუ a <= x <= z, მაშინ x მნიშვნელობა არის სიმბოლო.

Boolean ტიპს აქვს ორი მნიშვნელობა: True(ჭეშმარიტი) და False(მცდარი). მიუხედავად, რომ ამ ტიპის მონაცემების შესანახად თეორიულად საკმარისია ერთი ბიტი, ჩვეულებრივ ამ ტიპის მონაცემები მეხსიერებაში იკავებენ ერთ ბაიტს. ბულის ტიპის მონაცემებისათვის გათვალისწინებულია კონიუნქციის (AND), დიზიუნქციის (OR) და უარყოფის

(NOT) ოპერაციები.

TRUE AND TRUE = TRUE

TRUE AND FALSE = FALSE

FALSE AND TRUE = FALSE

FALSE AND FALSE = FALSE

TRUE OR TRUE = TRUE

TRUE OR FALSE = TRUE

FALSE OR TRUE = TRUE

FALSE OR FALSE = FALSE

NOT FALSE = TRUE

NOT TRUE = FALSE

2.3. ჩამოთვლითი მონაცემთა ტიპი

ჩამოთვლითი ტიპი შედგება სასრული რაოდენობის, მიმდევრობის სახით დალაგებული მნიშვნელობებისაგან. კლასიკურ ვარიანტში, როგორიცაა Pascal-ის მიმართულების ენები, ტიპის განსაზღვრა შედგება მნიშვნელობათა სახელების ჩამონათვალისაგან, ამიტომ თითოეულ მნიშვნელობას შეესაბამება ნატურალური რიცხვები 1-დან n-მდე, სადაც n ჩამოთვლითი ტიპის მნიშვნელობათა რიცხვს შეესაბამება. ასე, რომ ჩამოთვლითი ტიპის მნისვნელობის მიხედვით შეიძლება მისი ნომრის პოვნა, ხოლო ნომრის მიხედვით ჩამოთვლითი ტიპის

მნიშვნელობის პოვნა. გარდა ამისა, ჩამოთვლითი ტიპისათვის გათვალისწინებულია შედარების ოპერაციები და შესაძლებელია მომდევნო და წინა ელემენტების პოვნა. რადგან ჩამოთვლითი ტიპის მონაცემებს ცალსახად შეესაბამება ნატურალური რიცხვები, შესაძლებელია ამ მნიშვნელობების ნებისმიერ რიცხვით მონაცემთა ტიპის მნიშვნელობად არაცხადი გარდაქმნა.

C-ის მიმართულების ენებში „ჩამოთვლითი ტიპი“ განსხვავებულად მოიაზრება, რამდენადაც ასეთი ტიპის განსაზღვრისას მნიშვნელობის სახელად შეიძლება გამოყენებულ იქნას რომელიმე მთელი რიცხვი (არ არის აუცილებელი, რომ იგი იყოს დადებითი). თუ მთელი რიცხვები არაცხადად არის მოცემული,მაშინ ჩამოთვლიტი ტიპის პირველ ელემენტს არაცხადად შეესაბამება 0, ხოლო დანარჩენ ელემენტებს წინაზე ერთით მეტი მნიშვნელობა. ამასთან, ცვლადის გამოცხადებისათვის ჩამოთვლითი ტიპის სახელის გამოყენება Integer ტიპის გამოყენების ექვივალენტურია და ასეთი ცვლადი შეიძლება შეიცავდეს ნებისმიერ მთელ მნიშვნელობას. ჩამოთვლითი ტიპის მნიშვნელობების სახელები შეიძლება გავიგოთ, როგორც მთელი კონსტანტების სახელები და ამ მნიშვნელობებისათვის მიღებულია მთელი რიცხვების ყველა ოპერაცია. ასე, რომ C ენაში ჩამოთვლითი ტიპი არ მიეკუთვნება ტიპს ამ სიტყვის სრული მნიშვნელობით.

2.4. კონსტრუირებული მონაცემთა ტიპები

ლიტერატურაში კონსტრუირებული მონაცემთა ტიპები მოიხსენიება როგორც რთული მონაცემთა ტიპები. კონსტრუირებული ტიპებიდან ყველაზე გავრცელებულია: მასივის ტიპი, ჩანაწერის ტიპი და სიმრავლის ტიპი.

2.4.1. მასივები

მასივის ტიპისათვის დამახასიათებელია, რომ მისი თითოეული ელემენტი მიეკუთვნება ერთიდაიგივე საბაზო ტიპს.

მასივისა და მასივის ტიპის ცნება რამდენადმე განსხვავებულია მკაცრად ტიპიზებულ და შედარებით სუსტად ტიპიზებულ ენებში. მკაცრად ტიპიზებულ ენებში, როგორიცაა პასკალი, მასივის ტიპი განისაზღვრება ორი დამხმარე ტიპის საფუძველზე, როგორიცაა: მასივის ელემენტების ტიპი(საბაზო ტიპი) და მასივის ინდექსის ტიპი. მასივის განსაზღვრა, პასკალში, გამოიყურება შემდეგი სახით: type T = array [I] of TO, სადაც TO საბაზო ტიპია, ხოლო I - ტიპის ინდექსია. TO შეიძლება იყოს ნებისმიერი ჩადგმული ან წინასწარ განსაზღვრული ტიპი. I ინდექსის ტიპი შედგება ჩამოთვლითი მნიშვნელობების სასრული რაოდენობისაგან. პასკალის მიმართულების ენებში დასაშვებია მასივის ტიპის არაცხადი განსაზღვრა. მაგალითად, მასივი შეიძლება განისაზღვროს შემდეგი სახით: type T = array [1..6] of integer ან type T = array [1..6] of real.

sul

თუ მასივის ინდექსის სიდიდეა n, მაშინ მასივის მნიშვნელობა არის სტრუქტურა, რომელიც შეიცავს n რაოდენობის საბაზო ტიპს. ნებისმიერი ტიპის მასივისათვის შეიძლება განისაზღვროს ორი ოპერაცია: მასივის კონსტრუირებისა და ელემენტის ამორჩევის. თუ x არის T მასივის ცვლადი, ხოლო i ინდექსის მნიშვნელობა, შეიძლება ავაგოთ მასივი x := T(c1, c2,...,cn), სადაც c1, c2,...,cn - საბაზო ტიპის მნიშვნელობებია. მასივის ელემენტის ამორჩევისათვის გამოიყენება x[i] კონსტრუქცია, რომლის მნიშვნელობაა მასივის i-ური ელემენტის მნიშვნელობა.

მასივის ტიპის საბაზო ტიპი შეიძლება იყოს ნებისმიერი ჩადგმული ან განსაზღვრული ტიპი, მათ შორის მასივის ტიპიც. ამ ბოლოს აღნიშნულ შემთხვევაში, საქმე გვაქვს მრავალგანზომილებიან მასივებთან ანუ მატრიცებთან. მრავალგანზომილებიან მასივებთან მუშაობისათვის ენაში გამოიყენება შემოკლებული ჩანაწერი. მაგალითად, იმის ნაცვლად, რომ მასივი განისაზღვროს, როგორც type T = array [1..10] of array [1..5] of real, შეიძლება დაიწეროს შემდეგი სახით: type T = array [1..10],[1..5] of real, ხოლო ნებისმიერი ელემენტის ამორჩევისათვის ნაცვლად x[i][j]-სა დაიწერება x[i,j].

სუსტად ტიპიზებულ ენებში მასივებთან მუშაობის ილუსტრირების მიზნით შეიძლება განვიხილოთ ენა-C. მსგავს ენებში არის მასივის ცვლადების განსაზღვრის შესაძლებლობა. მასივის ცვლადების ელემენტების რაოდენობა განისაზღვრება

ცხადად ან საბაზო ტიპის ინიციალიზებული მნიშვნელობების სიის მეშვეობით. მაგალითად, მასივის ტიპი, რომელიც შედგება ოთხი მთელი ელემენტისაგან, შეიძლება განისაზღვროს, როგორც int x[4] (არაინიციალიზებული ვარიანტი), ან, როგორც int x[ ] = { 0, 12, 18, 22}(ინიციალიზებული ვარიანტი). მასივის ნებისმიერ ელემენტზე მიმართვა ხორციელდება იგივე კონსტრუქციით, როგორც მკაცრად ტიპიზებულ ენებში x[i] სადაც i არის მნიშვნელობა, რომელიც იღებს მხოლოდ მთელ მნიშვნელობას.

2.4.2. ჩანაწერები

მასივის ტიპისაგან განსხვავებით ჩანაწერის ელემენტები შეიძლება მიეკუთვნებოდეს სხვადასხვა ჩადგმულ ან ცხადად განსაზღვრულ მონაცემთა ტიპებს. ჩანაწერის(სტრუქტურული) ტიპი შეიცავს ველებს, რომელთათვისაც უნდა განისაზღვროს სახელი და ტიპი. ამის შემდეგ შეიძლება თითოეულ ველზე მიმართვის განხორციელება.

Modul-2 ენაში ჩანაწერი ,,კომპლექსური რიცხვები'' შეიძლება გამოიყურებოდეს შემდეგი სახით:

```
type complex = record re : real;
                      im : real
               end
```

C ენაში იგივე შეიძლება ჩაიწეროს შემდეგი სახით:

```
struct complex { float re;
```

```
        float im;
    }
```

ამის შემდეგ x ცვლადი შეიძლება გამოცხადდეს, როგორც კომპლექსური ტიპი (var x : complex; ან struct complex x;) და მივმართოთ x ცვლადის ნამდვილ და წარმოსახვით ნაწილებს x.re (ან შესაბამისად x.in) კონსტრუქციის მეშვეობით.

ჩანაწერში ერთი სახელის ქვეშ შეიძლება გაერთიანდეს სხვადასხვა ტიპის მქონე ველების ნაკრები. ჩანაწერები ძირითადად გამოიყენება რაიმე ობიექტის შესახებ ინფორმაციის შესანახად. ასეთი ობიექტი შეიძლება იყოს, მაგალითად, ადამიანი, რომლის ატრიბუტებია: გვარი, სახელი, ასაკი, მისამართი, ხელფასი, ტელეფონის ნომერი. ამ ინფორმაციის მთელ ჯგუფს შეიძლება მივცეთ ერთი სახელი.

ჩანაწერის ტიპი ვარიანტებით, პასკალში შეიძლება განისაზღვროს შემდეგი სახით:

```
type person = record   Lname, Fname : alfa;
                       birthday : date;
                       marstatus : (single, married);
                  case sex : (male, female) of
                  male : (weight : real;
                       bearded : boolean);
                  female : (size: array[1..3] of integer)
              end
```

იგულისხმება, რომ alfa და date მონაცემთა ტიპები უკვე

განსაზღვრულია. person ტიპის ცვლადის განსაზღვრის შემდეგ ნებისმიერ მომენტში შეიძლება მივმართოთ weight და bearded ველებს, ასევე size მასივის ელემენტებს, რა თქმა უნდა sex დისკრიმინანტის მნიშვნელობის მიხედვით.

C-ის მიმართულების ენებში გვხვდება მონაცემთა ტიპების სპეციალური სახე ე.წ. „ნარევი“ (Union). ეს არის ჩანაწერი ვარიანტებით, მაგრამ არაცხადად მხარდამჭერი დისკრიმინანტით. ზემოთ მოყვანილი მაგალითი C ენისათვის შეიძლება დაიწეროს შემდეგი სახით:

```
struct person { char Lname[10], Fname[10];
                integer birthday;
                enum { single, married ] marstatus;
                enum { male, female } sex;
        union {
                struct { float weight;
                integer bearded } male;
                integer female[3];
                } pers;
        }
```

უნდა აღინიშნოს, რომ ცხადად განსაზღვრული დისკრიმინანტის გამოყენება პასკალის მიმართულების ენებისთვისაც ითვლება არააუცილებლად.

2.4.3. სიმრავლე

კონსტრუირებული ტიპების ერთერთი სახესხვაობაა სიმრავლის ტიპი. ასეთი ტიპები გამოიყენება მხოლოდ განვითარებულ მკაცრად ტიპიზებულ ენებში. პასკალში სიმრალის ტიპი განისაზღვრება კონსტრუქციით: type T = set of TO, სადაც TO ჩადგმული ან წინასწარ განსაზღვრული მონაცემთა ტიპია (საბაზო ტიპი). T ცვლადის მნიშვნელობებია TO ტიპის ელემენტების სიმრავლე.

სიმრავლე ერთნაირი ტიპის ელემენტების ნაკრებია, რომლებიც რაღაც სახით არიან ერთმანეთთან დაკავშირებულნი. სიმრავლე, რომელიც არ შეიცავს არც ერთ ელემენტს, ჰქვია - ცარიელი. ცარიელი სიმრავლე აღინიშნება [ ] სიმბოლოთი.

ორი სიმრავლე ექვივალენტურია, თუ მათი ყველა ელემენტი ერთნაირია. ელემენტების მიმდევრობა შეიძლება იყოს ნებისმიერი. თუ ერთი სიმრავლის ყველა ელემენტი შედის მეორეში, შეიზლება ითქვას, რომ პირველი სიმრავლე ჩართულია მეორეში.

ვთქვათ მოცემულია რამოდენიმე სიმრავლე:

S1 := [‘1’,’2’,’3’]; S2 := [’3’,’2’,’1’];

S3 := [‘2’,’3’]; S4 := [0..3, 6];

S5 := [4, 5]; S6 := [3..9];

ამ მაგალითში სიმრავლე S1 ექვივალენტურია S2-სა, S3 ჩართულია S2 –ში, მაგრამ მისი ექვივალენტური არ არის.

სიმრავლის ელემენტები ერთმანეთისაგან გამოიყოფა

მძიმეებით და ჩაისმება კვადრატულ ფრჩხილებში.

სიმრავლეებისათვის განსაზღვრულია შემდეგი ოპერაციები:

* სიმრავლეთა თანაკვეთა, შედეგი შეიცავს მოცემულ სიმრავლეთა საერთო ელემენტს.

მაგალითად S4 * S6 შეიცავს [3, 6]-ს.

+ სიმრავლეთა გაერთიანებაა. შედეგი შეიცავს პირველი სიმრავლის ყველა ელემენტს მიმატებულს მეორე სიმრავლის განსხვავებულ ელემენტებს.

- სიმრავლეთა სხვაობა, შედეგი შეიცავს პირველი სიმრავლის იმ ელემენტებს, რომლებიც არ შედიან მეორე სიმრავლეში.

S6-S5 შეიცავს [3, 6, 7, 8, 9];

= სიმრავლეთა შემოწმება ექვივალენტურობაზე, შედეგია TRUE, თუ სიმრავლეები ექვივალენტურია.

< > არაექვივალენტურობაზე შემოწმება. შედეგია TRUE, თუ სიმრავლეები არაექვივალენტურია.

<= სიმრავლეთა შედარება. შედეგია TRUE, თუ პირველი სიმრავლე ჩართულია პირველში.

>= სიმრავლეთა შედარება, შედეგია TRUE, თუ მეორე სიმრავლე ჩართულია პირველში.

in შემოწმება კუთვნილებაზე. აქ პირველი ელემენტია გამოსახულება, მეორე კი - იგივე ტიპის სიმრავლე. შედეგია TRUE, თუ გამოსახულების მნიშვნელობა შედის სიმრავლეში.

3 in S6 = TRUE; 2 * 2 in S1 = FALSE.

III თავი.

ამოცანის მომზადებისა და ამოხსნის ტექნოლოგია კომპიუტერისათვის

3.1. კომპიუტერზე ამოცანის ამოხსნის ეტაპები

კომპიუტერის საშუალებით ამოცანის ამოხსნა მოიცავს შემდეგ ძირითად ეტაპებს, რომელთაგან ზოგიერთი მათგანი ხორციელდება კომპიუტერის გარეშე.

1. ამოცანის დასმა:

- ამოცანის შესახებ ინფორმაციის შეგროვება;
- ამოცანის პირობის ფორმულირება;
- ამოცანის ამოხსნის საბოლოო მიზნის განსაზღვრა;
- შედეგების გამოტანის ფორმის ანსაზღვრა;
- მონაცემთა აღწერა (ტიპების, სიდიდეთა დიაპაზონის, სტრუქტურის განსაზღვრა).

2. ამოცანის, მოდელის ანალიზი და გამოკვლევა:

- არსებული ანალოგების ანალიზი;
- ტექნიკური და პროგრამული საშუალებების ანალიზი;
- მათემატიკური მოდელის დამუშავება;
- მონაცემთა სტრუქტურის დამუშავება.

3. ალგორითმის დამუშავება:

- ალგორითმის პროექტირების მეთოდის არჩევა;
- ალგორითმის ჩაწერის ფორმის არჩევა (ბლოკ-სქემა, ფსევდოკოდი და სხვა);
- ტესტების და ტესტირების მეთოდის არჩევა;

- ალგორითმის პროექტირება.

4. პროგრამირება:

- პროგრამირების ენის არჩევა;

- მონაცემთა ორგანიზაციის ხერხების დაზუსტება;

- ალგორითმის ჩაწერა არჩეულ პროგრამირების ენაზე.

5. ტესტირება და გაწყობა:

- სინტაქსური გაწყობა;

- სემანტიკისა და ლოგიკური სტრუქტურის გაწყობა;

- ტესტირების შედეგების ანალიზი;

- პროგრამის სრულყოფა.

6. ამოცანის ამოხსნის შედეგების ანალიზი და საჭიროების შემთხვევაში მათემატიკური მოდელის დაზუსტება 2-5 ეტაპების განმეორებით შესრულებით.

7. პროგრამის დასრულება:

- პროგრამის დასრულება კონკრეტული ამოცანის ამოხსნისათვის;

- ამოხსნილი ამოცანისათვის საჭირო დოკუმენტაციის შედგენა.

3.2 ამოცანის ამოხსნის მათემატიკური მოდელი

ამოცანის ამოხსნის მათემატიკური მოდელის შესადგენად საჭიროა:

- შეირჩეს წინადადებები, რომელზეც აიგება მათემატიკური მოდელი;

- განისაზღვროს რა უნდა ჩაითვალოს საწყის მონაცემებად და შედეგებად;

- დაიწეროს მათემატიკური დამოკიდებულებები, რომელიც დააკავშირებს შედეგებს საწყის მონაცემებთან.

მათემატიკური მოდელის აგებისას ყოველთვის არ მოიძებნება ფორმულები, რომლებიც ცხადად ასახავენ საძიებო სიდიდეებისა და საწყის მონაცემების ურთიერთდამოკიდებულებას. ამ შემთხვევებში გამოიყენება მათემატიკური მეთოდები, რომლებიც სიზუსტის რაღაც ხარისხში იძლევიან ამონახსნს.

არსებობს რაიმე მოვლენის არა მარტო მათემატიკური მოდელირება არამედ ვიზუალური მოდელირება, რომელიც ხორციელდება ამ მოვლენის ასახვით მანქანური გრაფიკის საშუალებით.

3.3. პროგრამის დამუშავების პროცესის ეტაპები

პროგრამის დამუშავების პროცესი შეიძლება აღიწეროს ეტაპობრივად:

- სამუშაოს საწყის ეტაპზე ანალიზდება და ფორმულირდება პროგრამის მოთხოვნები, დამუშავდება იმის ზუსტი აღწერა, თუ რას უნდა აკეთებდეს პროგრამა და მისი მეშვეობით რა შედეგები უნდა მიიღწეოდეს.

- შემდგომში პროგრამა დამუშავდება პროგრამირების ამა თუ იმ ტექნოლოგიის გამოყენებით (მაგალითად, სტრუქტურული პროგრამირება).

პროგრამის მიღებული ვარიანტი გაივლის რამოდენიმე ტესტირებას, შეცდომების საბოლოოდ აღმოფხვრის მიზნით. პრაქტიკულად შეუძლებელია რთული, რეალური ამოცანის უშეცდომოდ შედგენა. არ შეიძლება დასკვნის გამოტანა, რომ პროგრამა სწორია, იმ შემთხვევაშიც კი თუ პროგრამაში აღარ არის არც ერთი შეცდომა და იგი იძლევა რაღაც შედეგს, რომელიც შეიძლება სულაც არ არის სწორი. ამ შემთხვევაში პროგრამაში შეიძლება იყოს დაშვებული უამრავი ლოგიკური შეცდომა. ამ ეტაპზე უნდა გადამოწმდეს პროგრამის მეტად საპასუხისმგებლო მონაკვეთები თუ რამდენად სწორად არის ჩამოყალიბებული ლოგიკური თვალსაზრისით.

3.4. პროგრამის გაწყობა და ტესტირება

პროგრამის ტექსტის კონტროლი შეიძლება დათვალიერებით, შემოწმებით და გადახვევით.

- დათვალიერების დროს უნდა შემოწმდეს ყველა ციკლის ორგანიზაცია, პირობის ოპერატორებში თუ რამდენად სწორად არის დასმული პირობა, რამდენად სწორად არის ორგანიზებული არგუმენტები ქვეპროგრამებში და ა.შ.

- შემოწმების პროცესში პროგრამისტი პროგრამის ტექსტის მიხედვით აზრობრივად ცდილობს აღადგინოს ის გამოთვლითი პროცესი, რომელსაც ასრულებს დაწერილი პროგრამა. ამ დროს ყურადღება არ უნდა მიექცეს თუ რისი შესრულება მოეთხოვება პროგრამას და ეს უნდა იქნას

გაგებული პროგრამის ტექსტის მიხედვით. პროგრამის შემოწმების დასრულების შემდეგ უნდა მოხდეს პროგრამის რეალური და მოთხოვნილი მოქმედებების შედარება.

- გადახვევის საფუძველია პროგრამისტის მიერ პროგრამის შესრულების იმიტაცია. ამ პროცესში უნდა იქნას შერჩეული რაღაც საწყისი მონაცემები და მათზე ჩატარდეს აუცილებელი გამოთვლები. ეს პროცესი საკმაოდ შრომატევადია, ამიტომ იგი მიზანშეწონილია პროგრამის ლოგიკურად რთული მონაკვეთების კონტროლის მიზნით.

პროგრამის გაწყობა კომპიუტერზე პროგრამის გაშვებისას შეცდომების ძიებისა და შეცდომების გასწორების პროცესია.

ტესტირება ახორციელებს პროგრამის შემადგენელი ნაწილებისა და მთლიანად პროგრამის მუშაობის სისწორის შემოწმებას.

გაწყობა და ტესტირება ორი ერთმანეთისაგან განსხვავებული ეტაპია:

- გაწყობისას მიმდინარეობს სინტაქსური შეცდომებისა და კოდირების ცხადი შეცდომების ლოკალიზება და აღმოფხვრა;

- ტესტირების პროცესში მოწმდება პროგრამის შრომისუნარიანობა, რომელიც არ შეიცავს ცხად შეცდომებს.

ტესტირება უჩვენებს შეცდომების არსებობას, ხოლო გაწყობა არკვევს შეცდომების მიზეზებს.

თანამედროვე პროგრამულ სისტემებში (Turbo Basic, Turbo

Pascal, Turbo C და სხვა) გაწყობა ხორციელდება სპეციალური პროგრამული საშუალებების გამოყენებით, რომელთაც ეწოდებათ გამწყობები. ეს საშუალებები იკვლევენ პროგრამის შიდა ქცევას.

პროგრამა-გამწყობს აქვს შემდეგი შესაძლებლობები:

- პროგრამის ნაბიჯ-ნაბიჯ შესრულება;

- ნებისმიერი ცვლადის მიმდინარე მნიშვნელობის დათვალიერება ან ნებისმიერი გამოსახულების მნიშვნელობის პოვნა, მათ შორის სტანდარტული ფუნქციების გამოყენებით;

- პროგრამაში ,,საკონტროლო წერტილების'' დაყენება, ანუ წერტილებისა, რომელშიც პროგრამა დროებით წყვეტს თავის მუშაობას, შუალედური შედეგების შეფასების მიზნით და ა.შ.

პროგრამის გაწყობისას საყურადღებოა:

- გაწყობის პროცესის დასაწყისში უნდა იქნას გამოყენებული მარტივი ტესტური მონაცემები;

- წარმოშვებული პრობლემები უნდა გამოიკვეთოს და ნაბიჯ-ნაბიჯ აღმოიფხვრას;

- შეცდომის მიზეზად არ უნდა ჩაითვალოს კომპიუტერი, რადგან თანამედროვე მანქანებსა და ტრანსლატორებს გააჩნიათ მაღალი საიმედობა.

პროგრამა პირობითად შეიძლება სწორად ჩაითვალოს, თუ მისი გაშვება სხვადასხვა საწყისი მონაცემებისათვის იძლევა სწორ შედეგს. ასე, რომ მნიშვნელოვანია ტესტების

სისტემაში პროგრამის შესრულებით მიღებული შედეგების კონტროლი.

ტესტური მონაცემები უნდა უზრუნველყოფდეს ყველა შესაძლო პირობის შემოწმებას.

- უნდა შემოწმდეს ალგორითმის ყველა ტოტი;

- ყველა მომდევნო ტესტი უნდა აკონტროლებდეს იმას, რაც არ იყო შემოწმებული წინა ტესტით;

- პირველი ტესტი უნდა იყოს მაქსიმალურად მარტივი, რათა შემოწმდეს მუშაობს თუ არა პროგრამა საერთოდ;

- ტესტებში არითმეტიკული ოპერაციები უნდა გამარტივდეს გამოთვლების მოცულობის შემცირების მიზნით;

- მიმდევრობათა ელემენტების რაოდენობა, ციკლებზე მიმართვის რაოდენობა, ტესტურ მაგალითებში, მოცემული უნდა იყოს გამოთვლათა მოცულობის შემცირების მიზნით;

- გამოთვლათა მინიმიზაცია არ უნდა ადაბლებდეს კონტროლის საიმედობას;

- ტესტირება უნდა იყოს მიზანმიმართული და სისტემატიზებული, რადგანაც საწყისი მონაცემების შემთხვევითი არჩევა მეტად გაართულებს მოსალოდნელი შედეგების მიღებას. გარდა ამისა ტესტური მონაცემების შემთხვევითი არჩევისას შეიძლება მრავალი სიტუაცია აღმოჩნდეს შეუმოწმებელი;

- ტესტური მონაცემების გართულება უნდა მიმდინარეობდეს თანდათან.

მაგალითი. ტესტების სისტემა კვადრატული განტოლების

ფესვების პოვნისათვის.

ტესტის ნომერი	შესამოწმებელი შემთხვევები	კოეფიციენტები			შედეგები
		a	b	c	
1	d > 0	1	1	-2	x1=1, x2=-2
2	d = 0	1	2	1	ფესვები ტოლია x1,x2=-1
3	d < 0	2	1	2	ამონახსნი არა აქვს
4	a=0, b=0, c=0	0	0	0	x ნებისმიერი რიცხვია
5	a=0, b=0, c<>0	0	0	2	განტოლება არასწორია
6	a=0, b<>0	0	2	1	წრფივი განტოლება. ერთი ფესვი აქვს: x =-0,5
7	a<>0, b<>0, c=0	2	1	0	x1=0, x2=-0,5

ტესტირების პროცესი შეიძლება დაიყოს სამ ეტაპად:

1. შემოწმება ნორმალურ პირობებში - ითვალისწინებს ტესტირებას იმ მონაცემების საფუძველზე, რომელიც დამახასიათებელია პროგრამის ფუნქციონირების რეალური პირობებისათვის.

2. შემოწმება ექსტრემალურ პირობებში. ტესტური მონაცემები შეიცავენ შემავალი მონაცემების ცვლილების არეს ზღვრულ მნიშვნელობებს, რომელიც პროგრამამ სწორ მონაცემებად უნდა აღიქვას. ტიპიური მაგალითია ძალიან მცირე ან ძალიან დიდი რიცხვები ან მონაცემთა არარსებობა. ექსტრემალური პირობის კიდევ ერთი ტიპია მონაცემთა ზღვრული მოცულობა, როდესაც მასივები შედგება ელემენტთა ძალიან დიდი ან ძალიან მცირე რაოდენობისაგან.

3. შემოწმება გამონაკლის სიტუაციებში, მიმდინარეობს იმ მონაცემთა გამოყენებით, რომელთა მნიშვნელობები იმყოფება ცვლილებათა დასაშვებ არეს გარეთ. ცნობილია, რომ ყველა პროგრამა მუშავდება რაღაც განსაზღვრული მონაცემების დამუშავებაზე დაყრდნობით, ამიტომ მნიშვნელოვანია შემდეგი საკითხების გათვალისწინება:

- რა მოხდება თუ პროგრამაში არ არის გათვალისწინებული უარყოფითი ან ნულოვანი მონაცემების დამუშავება და რაღაც შეცდომის გამო პროგრამას მოუხდება სწორედ ასეთი მონაცემების დამუშავება?

- როგორ მოიქცევა პროგრამა მასივების დამუშავებისას, თუ წევრთა რაოდენობა გადააჭარბებს გამოცხადებულ წევრთა რაოდენობას?

- რა მოხდება თუ რიცხვითი მნიშვნელობები იქნება ძალიან დიდი ან ძალიან მცირე?

ყველაზე უარესი სიტუაცია შეიქმნება მაშინ, თუ პროგრამა მცდარ მონაცემებს აღიქვამს სწორ მონაცემებად და მიიღებს არასწორ, მაგრამ სწორის მსგავს შედეგს.

პროგრამამ თვითონ უნდა გამოავლინოს ისეთი მონაცემები, რომლის სწორად დამუშავება მას არ შეუძლია.

3.5. პროგრამაში დაშვებული შეცდომების დახასიათება

შეცდომები შეიძლება დაშვებულ იქნას ამოცანის ამოხსნის ყველა ეტაპზე - ამოცანის დასმიდან დაწყებული, მისი გაფორმების ჩათვლით.

სხვადასხვა სახის შეცდომები და მათი შესაბამისი მაგალითები მოყვანილია ცხრილში:

შეცდომათა სახეობა	მაგალითი
ამოცანის არასწორი დასმა	არასწორად ფორმულირებული ამოცანის სწორი ამონახსნი
არასწორი ალგორითმი	ალგორითმის ამორჩევა, რომელიც იძლევა მცდარ ამონახსნს.
შეცდომები ანალიზში	სიტუაციის არასრული გათვლა; ლოგიკური შეცდომა.
სემანტიკური შეცდომები	ოპერატორების შესრულების მიმდევრობაში გაუგებრობა.
სინტაქსური შეცდომები	პროგრამირების ენაში გათვალისწინებული წესის დარღვევა.
შეცდომები ოპერაციის შესრულებისას	ძალზედ დიდი რიცხვია; ნულზე გაყოფა; უარყოფითი რიცხვიდან კვადრატული ფესვის ამოღებაა და ა.შ.
შეცდომები მონაცემებში	მონაცემთა შესაძლო ცვლილების დიაპაზონის წარუმატებელი განსაზღვრა.
შეცდომები ბეჭდვაში	შეცდომით არის დაბეჭდილი სიმბოლო, მაგალითად, ციფრი 1 და ასო *I, l*
შეცდომები შეტანა-გამოტანაში	შესატანი მონაცემების არასწორი გათვლა; მონაცემთა ფორმატის არასწორად განსაზღვრა.

ჩვეულებრივ, სინტაქსური შეცდომები გამოვლინდება კომპილაციის ეტაპზე. სხვა მრავალ შეცდომას კომპილატორი

ვერ იპოვის, რადგან მისთვის არ არის ცნობილი პროგრამისტის ნააზრები.

სინტაქსურ შეცდომებზე შეტყობინების არარსებობა არის აუცილებელი, მაგრამ არასაკმარისი პირობა იმისა, რომ პროგრამა გასწორებულად ჩაითვალოს.

სინტაქსური შეცდომების მაგალითებია:

- პუნქტუაციის ნიშნის გამოტოვება;
- ფრჩხილების შეუთანხმებლობა;
- ოპერატორის არასწორი ფორმირება;
- ცვლადის სახელების არასწორად გამოსახვა;
- ციკლის დასასრულების პირობის გამოტოვება;
- მასივის აღწერის გამოტოვება და ა.შ.

არსებობს მრავალი შეცდომა, რომლის გამოვლენა არ შეუძლია კომპილატორს, თუ პროგრამაში გამოყენებული ოპერატორები სწორად არის ფორმირებული. ასეთი შეცდომების მაგალითებია:

ა) ლოგიკური შეცდომები:

- ალგორითმში პირობის შემოწმების შემდეგ განშტოების არასწორად მითითება;
- შესაძლო პირობების არასწორი გათვლა;
- პროგრამაში ერთი ან მეტი ალგორითმული ბლოკის გამოტოვება;.

ბ) შეცდომები ციკლში:

- ციკლის დასაწყისის არასწორად მითითება;

- ციკლის დასასრულის პირობის არასწორად მითითება;

- ციკლის განმეორებათა რიცხვის არასწორად მითითება;

- უსასრულო ციკლი.

გ) შეცდომები შეტანა-გამოტანაში; შეცდომები მონაცემებთან მუშაობისას:

- მონაცემთა ტიპების არასწორად მითითება;

- მოთხოვნილზე მეტი ან მცირე მოცულობის მონაცემების გამოყენება;

- მონაცემთა არასწორი რედაქტირება.

დ) შეცდომები ცვლადების გამოყენებაში:

- ცვლადების გამოყენება მათზე საწყისი მონაცემების მინიჭების გარეშე;

- ერთი ცვლადის ნაცვლად, შეცდომით, სხვა ცვლადის მითითება.

ე) შეცდომები მასივებთან მუშაობისას:

- მასივები წინასწარ არ არის განულებული;

- მასივები არასწორად არის აღწერილი;

- ინდექსებს აქვთ არასწორი მიმდევრობა.

ვ) შეცდომები არითმეტიკულ ოპერაციებში:

- ცვლადის ტიპის არასწორად მითითება (მაგალითად, მთელრიცხვა, წილადის ნაცვლად);

- მოქმედებათა მიმდევრობის არასწორად განსაზღვრა;

- ნულზე გაყოფა;

- უარყოფითი რიცხვიდან კვადრატული ფესვის ამოღება;

- რიცხვის თანრიგის დაკარგვა.

ყვალა ეს შეცდომა გამოვლინდება ტესტირების პროცესში.

პროგრამის დასრულების პროცედურები დაკავშირებულია პროგრამის ექსპლოატაციასთან.

პროგრამა გამოყენების თვალსაზრისით მოითხოვს შემდეგ დამატებით სამუშაოებს:

- გამოვლენილი შეცდომების გასწორება;

- პროგრამის მოდიფიკაცია ცვალებადი საექსპლოატაციო მოთხოვნების დასაკმაყოფილებლად;

- კონკრეტული ამოცანის ამოსახსნელად პროგრამის დაყვანა საბოლოო სახემდე;

- დამატებითი ტესტური შემოწმების ჩატარება;

- მუშა დოკუმენტაციაში შესწორებათა შეტანა;

- პროგრამის სრულყოფა და ა.შ.

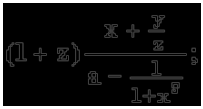
პროგრამას, რომელიც გათვალისწინებულია ხანგრძლივი ექსპლოატაციისათვის, თან უნდა ახლდეს დოკუმენტაცია და ინსტრუქცია მისი გამოყენების შესახებ.

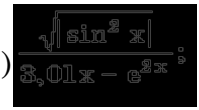
IV თავი

სავარჯიშოები

4.1. ჩაწერეთ გამოსახულებები ალგორითმული ენის წესების მიხედვით:

1) 

2) 

3) 

4) $\dfrac{\left|\cos x^2 - \sin^2 y\right|}{\sqrt{\left|\ln x\right| + xy}}$.

4.2. ჩაწერეთ გამოსახულებები ჩვეულებრივი მათემა-ტიკური ფორმით:

1) `a+b/c+1;`

2) `1/a*b/c;`

3) `a/b/c/d*p*q;`

4) `4/3*3.14*r*3;`

5) `b/sqrt(a*a+b);`

6) `5*ctg(x)-tg(y)/4;`

7) `lg(u*(1/3)+sqrt(v)+z);`

8) `ln(y*(-sqrt(abs(x))));`

9) `abs(x*(y/x)-(y/x)*(1/3));`

10) `sqrt((x1-x2)*2+(y1-y2)*2);`

11) `sqrt(exp(a*x)*sin(x)*n)/cos(x);`

12) `abs(cos(x)+cos(y))*(1+sin(y)*2);`

13) `sqrt(sin(ctg(u))*2+abs(cos(v)));`

14) `lg(sqrt(exp(x-y))+x*abs(y)+z);`

4.3. გამოითვალეთ ლოგიკური გამოსახულების მნიშვნელობა (შედეგი: True ან False):

1)`x*x+y*y<=9` თუ `x=1, y=-2`

პასუხი: True;

2) `b*b-4*a*c<0` თუ `a=2, b=1, c=-2;`

3)`a>=1)` და `(a<=2)` თუ `a=1.5;`

4) `(a<1)` ან `(a>1.2)` თუ `a=1.5;`

5) (a mod 7 = 1) და (a div 7 = 1) თუ `a=8;`

6) არა`((a>b)` და `(a<9)` ან `(a*a=4))` თუ `a=5, b=4.`

4.4. შეადგინეთ წრფივი სტრუქტურის ალგორითმი:

1) მოცემულია მართკუთხა სამკუთხედის კათეტები, გამოითვალეთ ჰიპოტენუზა და პერიმეტრი;

2) მოცემულია ორი რიცხვი, გამოითვალეთ მათი ჯამი, სხვაობა, ნამრავლი და განაყოფი;

3) მოცემულია სამი რიცხვი გამოითვალეთ მათი საშუალო არითმეტიკული და საშუალო გეომეტრიული;

4) გამოითვალეთ გამოსახულების მნიშვნელობა:

$$S = \frac{\left|\cos a^2 - \sin^2 b\right|}{\sqrt{\left|\ln a\right| + ab}}.$$

4.5. შეადგინეთ განშტოებადი სტრუქტურის ალგორითმი:

1) $$z=\begin{cases} x^2+y^2 & \text{თუ} \quad x^2+y^2<1, \\ x+y & \text{თუ} \quad x^2+y^2>1, \\ 0{,}5 & \text{თუ} \quad x^2+y^2=1. \end{cases}$$

2) $$z=\begin{cases} \lg(-x) & \text{თუ} \quad x<0 \\ \sqrt{x+1} & \text{წინააღმდეგ შემთხვევაში.} \end{cases}$$

3) $$V=\begin{cases} x+y & x>1;\ y>1, \\ x-y & \text{თუ} \quad x>1;\ y\le 1, \\ -x+y & \text{თუ} \quad x\le 1;\ y>0, \\ -x-y & \text{თუ} \quad x\le 1;\ y\le 0. \end{cases}$$

4) მოცემულია ორი რიცხვი იპოვეთ მათ შორის უდიდესი.

5) მოცემულია ორი რიცხვი იპოვეთ მათ შორის უმცირესი.

6) მოცემულია სამი რიცხვი იპოვეთ მათ შორის უდიდესი.

7) მოცემულია სამი რიცხვი იპოვეთ მათ შორის უმცირესი.

8) რიცხვი შეამოწმეთ ლუწობაზე და გამოიტანეთ შესაბამისი შეტყობინება(„ლუწია“ ან „კენტია“).

9) მოცემულია ორი ერთმანეთისაგან განსხვავებული რიცხვი, მათ შორის უდიდესი შეცვალეთ მათი საშუალო არითმეტიკულით, ხოლო უმცირესი - მათი გაორმაგებული ნამრავლით.

10) მოცემულია ორი ერთმანეთისაგან განსხვავებული რიცხვი, თუ პირველი მეტია მეორეზე, მაშინ ისინი შეცვალეთ

ნულებით, წინააღმდეგ შემთხვევაში ორივე დატოვეთ უცვლელად.

11) მოცემულია ორი ერთმანეთისაგან განსხვავებული რიცხვი, თუ ორივე უარყოფითია, მაშინ ისინი შეცვალეთ მათი მოდულებით, თუ ორივე დადებითია, მაშინ ისინი შეამცირეთ 5-ჯერ, ხოლო თუ ერთ-ერთი უარყოფითია, მაშინ ისინი გაადიდეთ 10-ჯერ.

4.6. შეადგინეთ ციკლური სტრუქტურის ალგორითმი:

1) გამოითვალეთ მითითებულ N1-N2 (N1<N2) დიაპაზონში მოთავსებულ მთელ რიცხვთა ჯამი და ნამრავლი.

2) გამოითვალეთ ყველა სამნიშნა რიცხვის ჯამი.

3) გამოითვალეთ ყველა ორნიშნა რიცხვის ნამრავლი.

4) გამოითვალეთ ნამრავლი: 1.1*1.2*1.3*1.4*...*2.

5) გამოითვალეთ A^N=A*A*...*A(N-ჯერ), სადაც A,N მოცემულია.

6) გამოითვალეთ n!=n*(n-1)*(n-2)*...*2*1.

7) გამოითველეთ ხუთი შეტანილი რიცხვის საშუალო არითმეტიკული.

8) გამოიტანეთ 1-დან N-მდე რიცხვების კვადრატული და კუბური მნიშვნელობები.

9) გამოიტანეთ A რიცხვის 1-დან N-მდე ხარისხების მნიშვნელობები.

10) შეიტანეთ A,N. გამოითვალეთ $A^1+A^2+...+A^N$.

11) გამოითვალეთ 1+1/2+1/3+...+1/N.

12) შეიტანეთ A. იპოვეთ პირველი N მნიშვნელობა, რომლისთვისაც ჭეშმარიტია უტოლობა: 1+1/2+1/3+...+1/N>A.

13) გამოითვალეთ y=sin(x) ფუნქციის მნიშვნელობები, თუ x იცვლება a-დან b-მდე, h ბიჯით. (a,b,h მოცემულია).

14) შეიტანეთ არანულოვანი მთელი რიცხვები, შეტანის დასასრულია რიცხვი - 0:

- დაითვალეთ [5,10] დიაპაზონში მოთავსებულ რიცხვთა რაოდენობა;

- დაითვალეთ [3,7] დიაპაზონის გარეთ მოთავსებულ რიცხვთა რაოდენობა;

- დაითვალეთ შეტანილ სამნიშნა რიცხვთა რაოდენობა;

- დაითვალეთ 3-ების და 5-ების რაოდენობა;

- გამოითვალეთ შეტანილ რიცხვთა ჯამი და ნამრავლი;

- გამოითვალეთ შეტანილ რიცხვთა საშუალო არითმეტიკული;

- დაითვალეთ შეტანილ რიცხვთა რაოდენობა;

- შეამოწმეთ იყო თუ არა შეტანილი რიცხვი 5. გამოიტანეთ შესაბამისი შეტყობინება: True ან False.

15) მოცემულია ნამდვილ რიცხვთა მიმდევრობა, შეადგინეთ ალგორითმის ბლოკ-სქემა, რომლითაც განისაზღვრება:

- წევრთა ჯამი;

- დადებითი წევრების ჯამი;

- უარყოფითი წევრების ჯამი;

- ლუწინდექსებიანი წევრების ჯამი;

- კენტინდექსებიანი წევრების ჯამი;
- ლუწინდექსებიანი დადებითი წევრების ჯამი;
- კენტინდექსებიანი დადებითი წევრების ჯამი;
- ლუწინდექსებიანი უარყოფითი წევრების ჯამი;
- კენტინდექსებიანი უარყოფითი წევრების ჯამი;
- დადებითი წევრების რაოდენობა;
- უარყოფითი წევრების რაოდენობა;
- მაქსიმალური ელემენტი;
- მინიმალური ელემენტი მოდულით;
- კენტინდექსებიან ელემენტთა შორის უდიდესი;
- ლუწინდექსებიან ელემენტთა შორის უდიდესი;
- ნულოვანი ელემენტების რაოდენობა;
- 1-ის ტოლ ელემენტთა რაოდენობა;
- წევრების ნამრავლი;
- დადებითი წევრების ნამრავლი;
- უარყოფითი წევრების ნამრავლი;
- ლუწინდექსიანი ელემენტების ნამრავლი;
- მოცემული რიცხვების კვადრატების ჯამი;
- $a_1a_2+a_2a_3+...+a_{n-1}a_n$.
- $a_1a_2+a_3a_4+...+a_{2n-1}a_{2n}$.

16) შეიტანეთ 10 რიცხვი:

- იპოვეთ მინიმალური ელემენტი და მისი ნომერი;
- იპოვეთ მაქსიმალური ელემენტი და მისი ნომერი;
- იპოვეთ მინიმალური ლუწი ელემენტი, თუ ასეთი არ

არის გამოიტანეთ 0.

- იპოვეთ მინიმალურ ელემენტთა რაოდენობა.

17) შეიტანეთ რიცხვთა მიმდევრობა, განსაზღვრეთ:

- წევრთა ჯამი, ნამრავლი და საშუალო არითმეტიკული;

- [3, 7] დიაპაზონში მოთავსებულ წევრთა ჯამი;

- [5, 8] დიაპაზონის გარეთ მოთავსებულ წევრთა ნამრავლი;

- წევრთა რაოდენობა, რომლებიც მეტია წინა წევრზე;

- წევრთა რაოდენობა, რომლებიც მეტია მეზობელ წევრზე;

- შეამოწმეთ ყველა წევრი ლუწია თუ არა და გამოიტანეთ შესაბამისი შეტყობინება;

- მინიმალურ და მაქსიმალურ ელემენტებს გაუცვალეთ ადგილები;

- ლუწ ელემენტებს დაუმატეთ ერთი ერთეული და გამოიტანეთ მიღებული მიმდევრობა.

18) მოცემულია მთელი რიცხვები $a_1 ... a_{100}$ მიღეთ ახალი მიმდევრობა ამ რიცხვებიდან ისე, რომ თუ $|a_i|$ არ უდრის $\max(a_1 ... a_{100})$ მაშინ ისინი შეცვალეთ 0-ებით, ხოლო წინააღმდეგ შემთხვევაში 1-ით.

19) მოცემულია ნამდვილ რიცხვთა მიმდევრობა $x_1, x_2, ..., x_{20}$ იპოვეთ ახალი $y_1, y_2, ..., y_n$ მიმდევრობა, რომლის წევრებიც გამოითვლებიან შემდეგნაირად:

$$y_1 = \frac{x_1}{x_{20}}, \quad y_2 = \frac{x_2}{x_{19}}, \cdots, y_{20} = \frac{x_{20}}{x_1}$$

20) მოცემულია ნამდვილ რიცხვთა $a_1, a_2, ..., a_{20}$

მიმდევრობა, იპოვეთ ახალი მიმდევრობა, რომლის წევრები განლაგებული იქნებიან შემდეგი მიმდევრობით:

ა) $a_1, a_{11}, a_2, a_{12}, \ldots, a_{10}, a_{20}$

ბ) $a_{20}, a_1, a_{19}, a_2, \ldots, a_{11}, a_{10}$

21) მოცემულია ნამდვილ რიცხვთა ორი მიმდევრობა:

$a_1, a_2, \ldots, a_{21}$ და $b_1, b_2, \ldots, b_{21}$ იპოვეთ შემდეგი გამოსახულებების მნიშვნელობების მნიშვნელობები:

ა) $\frac{a_1 b_2}{b_1 a_2} + \frac{a_2 b_3}{b_2 a_3} + \cdots + \frac{a_{20} b_{21}}{b_{20} a_{21}}.$

ბ) $a_1 b_{21} + a_2 b_{20} + \cdots + a_{21} b_1.$

გ) $a_1 b_2 + a_2 b_3 + \cdots + a_{20} b_{21}.$

დ) $a_1 b_2 + a_3 b_4 + \cdots + a_{19} b_{20}.$

ე) $\frac{(a_1 b_1 + a_3 b_3 + \cdots + a_{19} b_{19})}{(a_2 b_2 + a_4 b_4 + \cdots + a_{20} b_{20})}.$

22) იპოვეთ სამკუთხედის უდიდესი ფართობი, თუ გვერდები მოცემულია ნამდვილ რიცხვთა შემდეგი მიმდევრობის სახით:

$a_1, a_2, \ldots, a_{20}$; $b_1, b_2, \ldots, b_{20}$; $c_1, c_2, \ldots, c_{20}$

სამკუთხედის ფართობი გამოითვალეთ ჰერონის ფორმულით:

$$s = \sqrt{p(p-a)(p-b)(p-c)}$$

23) მოცემულია სიმბოლოების მიმდევრობა $s_1 \ldots . s_{80}$ განსაზღვრეთ არასწორი ტოლობების რაოდენობა:

ა) $s_1=s_{41}$, $s_2=s_{42}$, ... , $s_{40}=s_{80}$ ტოლობებს შორის.

ბ) $s_1=s_{80}$, $s_2=s_{79}$, ... , $s_{40}=s_{41}$ ტოლობებს შორის.

24) მოცემული *B(N)* მასივის ყველა წევრი გადაწერეთ ახალ *A(N)* მასივში უკუმიმდევრობით.

25) მოცემული *A(3N)* მიმდევრობიდან მიიღეთ ახალი *B(N)* მიმდევრობა, რომლის ყოველი მომდევნო კომპონენტი უდრის *A* მიმდევრობის ყოველი მომდევნო სამი კომპონენტის საშუალო არითმეტიკულს.

26) გამოითვალეთ მნიშვნელობები:

- $\sin x + \sin^2 x + ... + \sin^n x;$

- $\sin x + \sin x^2 + ... + \sin x^n;$

- $\sin x + \sin^2 x^2 + ... + \sin^n x^n;$

- $\sin x + \sin \sin x + ... + \sin \sin...\sin x$ *(n-ჯერ);*

27) გამოითვალეთ მოცემული *X(N)* მიმდევრობის ყველა წევრის კვადრატების ჯამი, გარდა 5-ის ჯერადი წევრებისა.

28) გამოითვალეთ $z = (a + b + c_i) / i$, ფუნქციის მნიშვნელობა, თუ *a* იცვლება 0-დან 1-ის ბიჯით, *b* იცვლება 5-დან 1-ის ბიჯით, c_i წარმოადგენს *C(N)* მიმდევრობის ელემენტებს. მავე დროს *a* და *b* იცვლება *i*-სთან ერთდროულად.

29) მოცემულ *A(N)* მიმდევრობაში განსაზღვრეთ იმ ელემენტების რაოდენობა, რომლებიც ნაკლებია მოცემულ მნიშვნელობაზე.

30) განახორციელეთ მოცემულ *A(N)* მიმდევრობის ელემენტების ციკლური დაძვრა ერთი პოზიციით მარცხნივ, ანუ მიიღეთ მიმდევრობა $A = (a_2, a_3, ..., a_n, a_1)$.

31) განახორციელეთ მოცემულ *A(N)* მიმდევრობის ელემენტების ციკლური დაძვრა ორი პოზიციით მარჯვნივ, ანუ მიიღეთ მიმდევრობა $A = (a_{N-1}, a_N, a_1, a_2, ..., a_{N-2})$.

32) *B(2N)* მიმდევრობიდან ლუწინდექსიანი ელემენტები გადაწერეთ *A(N)* მიმდევრობაში, ხოლო კენტინდექსიანი *B(N)* მიმდევრობაში.

33) მოცემულ *A(N)* მიმდევრობაში a_1 შეცვალეთ მიმდევრობის უდიდესი ელემენტით, ხოლო a_N - უმცირესი ელემენტით.

34) მოცემულ *A(N)* მიმდევრობაში იპოვეთ:

- უარყოფით წევრებს შორის უდიდესი;
- დადებით წევრებს შორის უმცირესი;
- სიდიდის მიხედვით მეორე ელემენტი.

35) მოცემულ *A(N)* მიმდევრობაში დაითვალეთ რამდენია მეზობლად მდგომი:

- ორი დადებითი რიცხვი;
- ორი საწინააღმდეგო ნიშნის რიცხვი;
- ორი ერთნაირი ნიშნის რიცხვი, თანაც პირველი რიცხვის აბსოლუტური მნიშვნელობა მეტი უნდა იყოს მეორე რიცხვის აბსოლუტურ მნიშვნელობაზე;
- ლუწი რიცხვები.

36) მოცემულ *A(N)* მიმდევრობაში დადებითი ელემენტები ორჯერ შეამცირეთ, ხოლო უარყოფითი ელემენტები შეცვალეთ მათი ინდექსების მნიშვნელობებით.

37) მოცემულია დადებითელემენტებიანი $A(3, N)$ მატრიცა. განსაზღვრეთ a_{1i}, a_{2i}, a_{3i} $(i = 1, ..., N)$ სამეულიდან რომლები შეიძლება გამოდგეს სამკუთხედის გვერდებად. გამოიტანეთ $b_1, ..., b_N$ მიმდევრობა (მასივი), რომელიც შედგება ნულებისა და ერთიანებისაგან. თუ a_{1i}, a_{2i}, a_{3i} სამეული გამოდგება სამკუთხედის გვერდებად, მაშინ b_i=1, წინააღმდეგ შემთხვევაში b_i=0.

38) აეროფლოტის სალაროსთან დგას N ადამიანი. მოლარეს თითოეული i კლიენტის მომსახურეობისათვის სჭირდება $T_i(i = 1, ..., n)$ დრო.

- განსაზღვრეთ თითოეული კლიენტის რიგში დგომის დრო;

- იპოვეთ კლიენტის ნომერი, რომლის მომსახურეობაზეც დაიხარჯა ყველაზე მეტი დრო.

39) ფიგურულ სრიალში შეჯიბრების დროს ჟიურის N წევრი ერთმანეთისაგან დამოუკიდებლად უჩვენებს ქულებს. გამოცხადებული ქულებიდან ამოაგდებენ ერთ ყველაზე მაღალ და ერთ ყველაზე დაბალ ქულას. დარჩენილი ქულებიდან გამოითვლება საშუალო არითმეტიკული, რომელიც გამსვლელად ჩაითვლება. ჟიურის მოცემული შეფასებების მიხედვით განსაზღვრეთ სპორტსმენის გამსვლელი ქულა.

40) მასწავლებელმა გამოაცხადა გამოცდების შედეგები. განსაზღვრეთ გამოცხადებული: ხუთების, ოთხების, სამების ორების პროცენტობა.

41) ფუნტი სტერლინგი დიდი ბრიტანეთის ფულადი ერთეულია, 1971 წლამდე იგი უდრიდა 20 შილინგს ან 240 პენსს. ლივერპულის პორტში შემოსული გემიდან ჩამოვიდა N მგზავრი, რომელთაგან თითოეულს ჰქონდა თითო ათფუნტიანი კუპიურა. მათ იყიდეს სუვენირები p_1, p_2, ..., p_n თანხის შესაბამისად. რამდენი ფუნტი, შილინგი და პენსი მიიღო თითოეულმა მგზავრმა ხურდის სახით.

42) მოცემულია *A(N, M)* მატრიცა. იპოვეთ მისი უდიდესი ელემენტი და იმ სტრიქონისა და სვეტის ნომერი, რომლის გადაკვეთაზეც იგი მდებარეობს.

43) მოცემულ *A(N, M)* მატრიცის თითოეულ სტრიქონში გამოითვალეთ დადებითი ელემენტების ჯამი, რაოდენობა და საშუალო არითმეტიკული.

44) მოცემულ *A(N, M)* მთელრიცხვა მატრიცაში განსაზღვრეთ არის თუ არა ლუწი მისი ელემენტები ჯამი და გამოიტანეთ შესაბამისი შეტყობინება.

45) მოცემულია *A(N, M)* მატრიცა. იპოვეთ ამ მატრიცის იმ ელემენტთა რაოდენობა, რომლებიც მეტია ყველა ელემენტის საშუალო არითმეტიკულზე.

46) მოცემულია *A(N, M)* მთელრიცხვა მატრიცა. გაუცვალეთ ადგილები რომელიმე ორ სვეტს.

47) მოცემულია *A(N, M)* მთელრიცხვა მატრიცა. გაუცვალეთ ადგილები მის უდიდეს და უმცირეს ელემენტებს.

48) მოცემულია *A(N, N)* მატრიცა. გადაწერეთ მისი

მთავარი დიაგონალის წევრები ერთგანზომილებიან *Y(N)* მასივში და გაყავით ისინი მთავარი დიაგონალის მაქსიმალურ ელემენტზე.

49) ააგეთ *A(N, N)* მატრიცა, რომლის ელემენტები განისაზღვრება $a_{ij} = i + 2 \cdot j$ ტოლობით, აგრეთვე იპოვეთ ამ მატრიცის ლუწი ელემენტების ნამრავლი, რომლებიც აკმაყოფილებენ $a_{ij} < P\ (0< P<3N)$ პირობას.

50) მოცემული სამი *A(N, N), B(N, N)* და *C(N, N)* მატრიციდან ააგეთ იმავე ზომის *X* მატრიცა, რომლის თითოეული ელემენტი გამოითვლება $x_{ij} = max\{a_{ij}, b_{ij,} c_{ij}\}$. ფორმულით.

51) მოცემულია *A(N, N)* მატრიცა და *P* მთელი რიცხვი. გარდაქმენით მატრიცა შემდეგი წესით: *P* ნომრის სტრიქონი გარდაქმენით *P* ნომრის სვეტად, ხოლო *P* ნომრის სვეტი - *P* ნომრის სტრიქონად.

52) მოცემულია *A(N, N)* მატრიცა. განსაზღვრეთ:

- მატრიცის თითოეულ სტრიქონში არანულოვანი ელემენტების რაოდენობა;

- მატრიცაში არანულოვანი ელემენტების საერთო რაოდენობა;

- თითოეულ სტრიქონში არანულოვანი ელემენტების რაოდენობის თანაფარდობა მატრიცაში არანულოვანი ელემენტების საერთო რაოდენობასთან.

53) მოცემულ *A(N, M)* მატრიცაში წაშალეთ K

ნომრის სტრიქონი და P და Q ნომრის სვეტები. მიღებული მატრიცა შეამჭიდროვეთ.

54) მოცემულ *X(N, M)* მატრიცაში ყველა რიცხვი განსხვავებულია. თითოეულ სტრიქონში აირჩევა მინიმალური ელემენტი, შემდეგ ამ რიცხვებს შორის აირჩევა მაქსიმალური. დაბეჭდეთ მატრიცის სტრიქონის ნომერი, რომელშიც არის დაბეჭდილი არჩეული რიცხვი.

55) მოცემულ *A(N, M)* მატრიცაში სტრიქონები გადააადგილეთ ისე, რომ მათი ელემენტების ჯამები დალაგდნენ ზრდის მიხედვით.

56) მოცემულ *A(N, M)* მატრიცაში სტრიქონებისა და სვეტების გადაადგილებით მიაღწიეთ იმას, რომ უდიდესი ელემენტი (ერთ-ერთი მათგანი) აღმოჩნდეს მარცხენა ზედა კუთხეში.

57) ავტობუსის ბილეთის ექვსნიშნა ნომერს უწოდებენ „ბედნიერს“, თუ ტოლია პირველი სამი და ბოლო სამი ციფრების ჯამი. დაითვალეთ „ბედნიერი“ ბილეთების რაოდენობა.

58) მოცემულია $a_1, a_2, \ldots, a_n$ მთელ რიცხვთა მიმდევრობა. ააგეთ ახალი იმ რიცხვებისაგან შედგენილი მიმდევრობა, რომლებიც საწყის მიმდევრობაში მხოლოდ ერთხელ შედიან.

59) განსაზღვრეთ მოცემულ *A(N)* მიმდევრობაში არის თუ არა მეზობელი რიცხვებიდან თუნდაც ერთი საწინააღმდეგო ნიშნის წყვილი.

60) განსაზღვრეთ არის თუ არა მოცემულ *A(N)* მიმდევრობაში რიცხვი,რომელიც ჯერადია A და B რიცხვების, ხოლო არ არის ჯერადი C რიცხვის,

61) განსაზღვრეთ არის თუ არა მოცემული ნატურალური რიცხვი „პოლინდრომი“ (პოლინდრომი - არის რიცხვი, რომელიც ერთნაირად იკითხება მარცხნიდან და მარჯვნიდან).

62) მოცემულია მთელი რიცხვი *A>1*. იპოვეთ უმცირესი არაუარყოფითი მთელი *k* რიცხვი, რომლისთვისაც $5^k > A$.

63)* დასახლება შედგება *N* რაოდენობის მრავალბინიანი სახლისაგან. რომლებიც განლაგებულნი არიან სწორი გზის გასწვრივ და ერთმანეთისაგან თანაბრად არიან დაშორებულნი. დასახლებაში გაჰყავთ სატელეფონო ხაზი. ცნობილია, თუ რამდენი ტელეფონის აპარატი უნდა დაყენდეს თითოეულ სახლში. განსაზღვრეთ რომელ სახლში უნდა დაყენდეს ატეესი, რათა ატეესიდან ყველა ტეელეფონის აპარატამდე ჯამური დაშორება იყოს მინიმალური. თუ ასეთი სახლი რამოდენიმეა საკმარისია ნებისმიერი მათგანის პოვნა. გაითვალისწინეთ, რომ თითოეული ტელეფონი დაკავშირებულია ატეესთან ცალკეული მავთულით.

64)* მთვლელის გარშემო დგას *N* ადამიანი, რომელთაგან ერთ-ერთ ითვლება პირველ ნომრად. დანარჩენები დანომრილია საათის ისრის მიმართულებით 2-დან *N*-მდე რიცხვებით. მთვლელი ითვლის ერთიდან *M*-მდე. ადამიანი, რომელზეც გაჩერდა დათვლა გადის წრიდან. თვლა გრძელდება გასულის

მომდევნოდან და გრძელდენა მანამ, სანამ არ დარჩება ერთი ადამიანი. განსაზღვრეთ საბოლოოდ დარჩენილი ადამიანის საწყისი ნომერი.

65)* მოცემულია *S* სტრიქონი. შეამოწმეთ არის თუ არა ეს სტრიქონი ჩაწერილი რომაული ციფრებით. თუ არის, მაშინ გამოიტანეთ მომდევნო ცხრა რიცხვი რომაული ციფრებით. (მაგალითად თუ *S ='VI',* უნდა გამოიტანოთ 'VII, VIII, IX, X, XI, XII, XIII, XIV, XV'). თუ სტრიქონი არ არის რომაული რიცხვი, მაშინ გამოიტანეთ შეტყობინება შეცდომის შესახებ.

შენიშვნა. რიცხვთა რომაულ სისტემაში ზოგიერთი საბაზო რიცხვები აღინიშნება მაღალი რეგისტრის ლათინური ასოებით. 1 - I, 5 - V, 10 - X, 50 - L, 100 - C, 500 - , 1000 - M. ყველა დანარჩენი რიცხვი აიგება საბაზოს კომბინაციებით შემდეგი წესების შესაბამისად:

- თუ ნაკლები სიდიდის ციფრი დგას დიდი მნიშვნელობის ციფრის მარჯვნივ, მაშინ მათი მნიშვნელობები შეჯამდება;

- თუ პირიქით, მაშინ დიდ მნიშვნელობას აკლდება მცირე მნიშვნელობა, თანაც შესაძლებელია მხოლოდ შემდეგი შერწყმა: IV, IX, X, XC, CD, CM;

- I, X, C, M ციფრებიდან თითოეული მიმდევრობით შეიძლება გვხვდებოდეს მხოლოდ სამჯერ;

- V, L, D ციფრებიდან თითოეული მიმდევრობით შეიძლება გვხვდებოდეს მხოლოდ ერთჯერ.

V თავი

ზოგიერთი მათემატიკური ალგორითმი

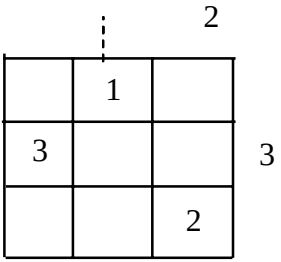
მოცემულ თავში განხილულია რამოდემიმე მაგალითი. თითოეული შეიცავს ძალიან ეფექტურ და ზუსტ ალგორითმს, რომელიც საკმაოდ მარტივად აიხსნება, თუმცა პროგრამულად რთულად რეალიზებადია.

5.1. კენტგანზომილებიანი მაგიური კვადრატის აგების ალგორითმი

მაგიური კვადრატი არის NxN მატრიცაში ისეთი განლაგება, რომ თითოეულ სტრიქონში, თითოეულ სვეტში და ორ მთავარ დიაგონალში ჯამები ერთმანეთის ტოლია. ჩვეულებრივ აიღება მთელი რიცხვები 1,2, . . . , N^2. განვიხილოთ კენტგანზომილებიანი მაგიური კვადრატის აგების მაგალითი.

რიცხვი 1 განვათავსოთ ზედა სტრიქონის შუა პოზიციაში. შემდეგი რიცხვის განთავსებისათვის გადავიდეთ ერთი ბიჯით ზემოთ და მარჯვნივ. ამ გადაადგილებამ შეიძლება გაგვიყვანოს მასივის გარეთ, მაგრამ სინამდვილეში ეს რიცხვი უნდა განთავსდეს სამი პოზიციით ქვემოთ ან მარცხნივ (3 შეესაბამება მატრიცის განზომილებას). გადაადგილება „ზემოთ და მარჯვნივ“ არის მაგიური კვადრატის აგების აუცილებელი პირობა. თითოეული N (მოცემულ შემთხვევაში 3) გადაადგილების შემდეგ, უნდა დავიძრათ ქვემოთ შემდეგი რიცხვის განსათავსებლად.

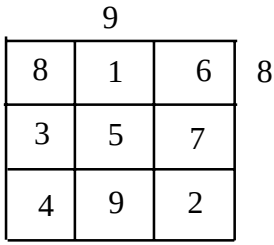
ნახ. 4.1-ზე ნაჩვენებია 3X3 განზომილების მაგიური კვადრატის აგება ეტაპობრივად.



ა)

	1	6
3	5	7
4		2

ბ)



გ)

8	1	6
3	5	7
4	9	2

დ)

ნახ. 5.1.

ალგორითმის სისწორის დასამტკიცებლად შეგიძლიათ ააგოთ სხვა სიდიდის კენტგანზომილებიანი მაგიური კვადრატი. 5X5 განზომილებიანი მაგიური კვადრატი გამოიყურება შემდეგი სახით:

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

ნახ. 5.2.

5.2. ოთხის ჯერადი განზომილების მაგიური კვადრატის აგების ალგორითმი

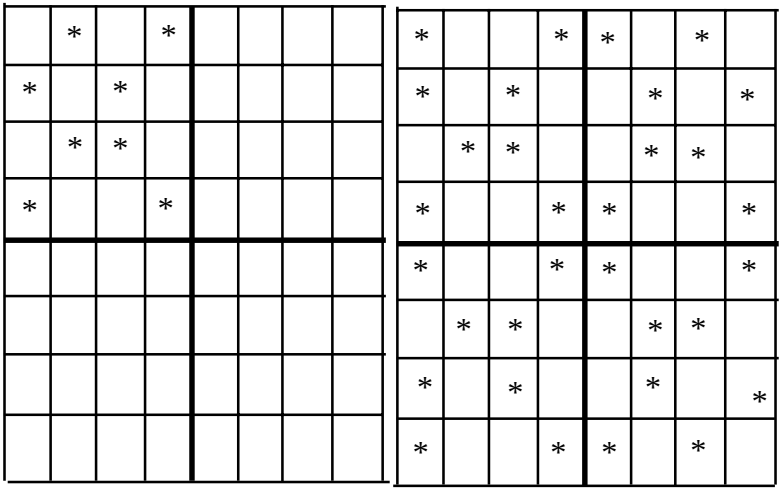
ოთხის ჯერადი განზომილების მაგიური კვადრატის აგება რამდენადმე რთულია წინაგანხილულთან შედარებით. ავაგოთ 8X8 ზომის მაგიური კვადრატი, რისთვისაც მთელი არე დავყოთ ოთხ მეოთხედად. მოხერხებულობის მიზნით პირველ მეოთხედად ავიღოთ მარცხენა ზედა მეოთხედი და უჯრები აღვნიშნოთ ნებისმიერად, მაგრამ გავითვალისწინოთ, რომ თითოეულ სტრიქონში და თითოეულ სვეტში მონიშნული იყოს ორ-ორი უჯრა. ნახ. 4.3.ა.

ავსახოთ ეს აღნიშვნები სიმეტრიის ვერტიკალური ღერძის შესაბამისად ზედა მარჯვენა მეოთხედში, შემდეგ კი სიმეტრიის ჰორიზონტალური ღერძის შესაბამისად - ქვედა მარცხენა მეოთხედში. ბოლოს ავსახოთ ეს აღნიშვნები ქვედა მარცხენა კუთხიდან ზედა მარჯვენა კუთხეზე გამავალი დიაგონალის შესაბამისად, ქვედა მარჯვენა მეოთხედში. შედეგად მივიღებთ აღნიშვნათა სიმრავლეს. ნახ. 4.3.ბ.

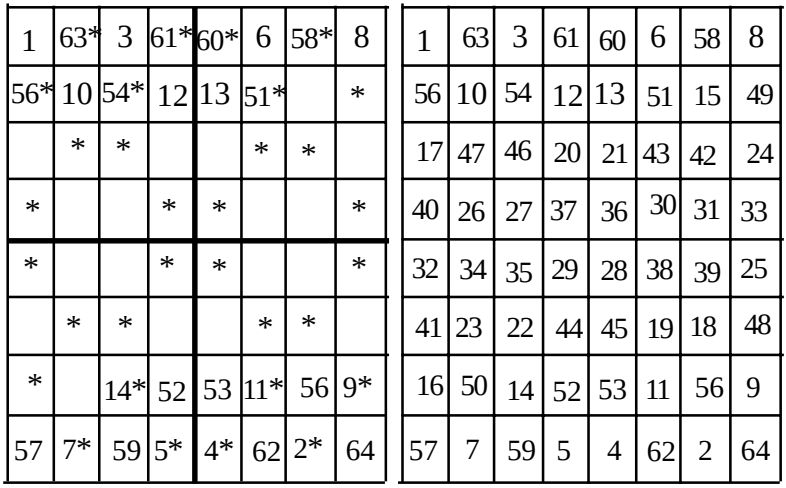
განვიხილოთ ყველა უჯრა მარცხნიდან მარჯვნივ და ზემოდან ქვემოთ. თუ კვადრატს არა აქვს აღნიშვნა ჩავსვათ მასში პირველი ჯერ კიდევ აურჩეველი K რიცხვი, 1, 2, . . . , N^2 მიმდევრობიდან, ხოლო რიცხვი $(N^2+1)-K$ ჩავსვათ ამ უჯრის ცენტრის მიმართ სიმეტრიულ უჯრაში. თუ მომდევნო უჯრა შეიცავს აღნიშვნას, მასში ჩავსვათ $(N^2+1)-K$ რიცხვი, ხოლო K ჩავსვათ მის სიმეტრიულ უჯრაში. გავაგრძელოთ

განხილული პროცედურა მასივის სრულად შევსებამდე.

ნახ. 4.3.-ზე ნაჩვენებია ზემოთგანხილული მაგიური კვადრატის აგება ეტაპობრივად.



ა) ბ)



გ) ნახ. 5.3. დ)

5.3. სორტირების ალგორითმები

1. სორტირება ამორჩევით (Selection sort).

განვიხილოთ მარტივი შემთხვევა, როდესაც მიმდევრობის წევრები წარმოადგენენ ნამდვილი რიცხვების $a_0, a_1, \ldots a_{n-1}$ მასივს. ალგორითმის მიმდევრობა ასეთია:

1. საწყის მასივში ვიპოვოთ მინიმალური ელემენტი ყველა წევრს შორის a_i (min) და მან ადგილი გაუცვალოს პირველ ელემენტს;

2. 1-დან n-1 ინდექსის მქონე რიცხვებს შორის ვიპოვოთ მინიმალური და მან ადგილი გაუცვალოს მეორე ელემენტს;

3. პირველ და მეორე პუნქტში აღწერილი მოქმედებები განმეორდეს i-დან n-1 ინდექსის მქონე რიცხვებისათვის.

ცხრილში ნაჩვენებია, თუ როგორ მუშაობს ალგორითმი ნაბიჯ-ნაბიჯ კონკრეტული მასივის შემთხვევაში.

0 ბიჯი	14	11	8	16	2	12	19	7
1 ბიჯი	2	11	8	16	14	12	19	7
2 ბიჯი	2	7	8	16	14	12	19	11
3 ბიჯი	2	7	8	16	14	12	19	11
4 ბიჯი	2	7	8	11	14	12	19	16
5 ბიჯი	2	7	8	11	12	14	19	16
6 ბიჯი	2	7	8	11	12	14	19	16
7 ბიჯი	2	7	8	11	12	14	16	19

ნახ. 5.3.1.

2. სორტირება ბუშტულების მეთოდით ***(buble sort).***

ალგორითმის იდეაა მოცემულ $a_0, a_1, \ldots a_{n-1}$ რიცხვებში დავათვალიეროთ ყველა წყვილი და დავალაგოთ, სადაც დარღვეულია ზრდადობით დალაგება. დალაგების ციკლს ექნება შემდეგი სახე:

ციკლი (i = n - 1; i > 0; i - -)

ციკლი (j = 1; j < = i; j + +)

თუ (a [j - 1] > a [j]), მაშინ

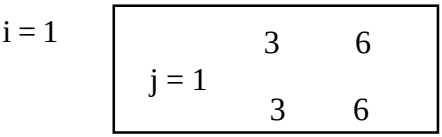
შევუცვალოთ ადგილები a [j - 1] და a [j]. ვნახოთ თუ როგორ მუშაობს ალგორითმი რომელიმე კონკრეტული მონაცემებისათვის.

i = 3

j = 1	7	6	3	10
	6	7	3	10
j = 2				
	6	3	7	10
j = 3				
	6	3	7	10

i = 2

	6	3	7	10
j = 1				
	3	6	7	10
j = 2				
	3	6	7	10



ნახ. 5.3.2.

ამ ალგორითმის დადებითი თვისებაა, რომ გარე ციკლის ყოველი გავლის შემდეგ ერთი ელემენტი მაინც იკავებს თავის უკიდურეს მარჯვენა ადგილს. სწორედ ეს არის გათვალისწინებული, როცა ავჯამვა ხდება i-მდე.

3. სორტირება ჩასმით. მასივის თითოეული ელემენტისათვის მეორედან დაწყებული მიმდინარეობს შედარება მცირეინდექსიან ელემენტებთან (a_i ელემენტი შეედარება a_{i-1}, a_{i-2}, ...) და მანამ, სანამ მომდევნო a_j ელემენტისათვის სრულდება a[j] > a[i] პირობა, a[i] და a[j] ელემენტები ცვლიან ადგილებს. თუ შეგვხვდება ისეთი a[j] ელემენტი, რომლისთვისაც a[j] <= a[i], ანუ თუ მიღწეულია მასივის ქვედა ზღვარი, დაიწყება a[i+1] ელემენტის დამუშავება, სანამ არ მიიღწევა მასივის ზედა ზღვარი. განვიხილოთ კონკრეტული მაგალითი:

მასივის საწყ. მდგომარ.	8 23 5 65 44 33 1 6
1 ბიჯი	8 23 5 65 44 33 1 6
2 ბიჯი	8 5 23 65 44 33 1 6 5 8 23 65 44 33 1 6
3 ბიჯი	5 8 23 65 44 33 1 6
4 ბიჯი	5 8 23 44 65 33 1 6
5 ბიჯი	5 8 23 44 33 65 1 6 5 8 23 33 44 65 1 6

6 ბიჯი	5 8 23 33 44 1 65 6 5 8 23 33 1 44 65 6 5 8 23 1 33 44 65 6 5 8 1 23 33 44 65 6 5 1 8 23 33 44 65 6 1 5 8 23 33 44 65 6
7 ბიჯი	1 5 8 23 33 44 6 65 1 5 8 23 33 6 44 65 1 5 8 23 6 33 44 65 1 5 8 6 23 33 44 65
საბოლოო შედეგი	1 5 6 8 23 33 44 65

ნახ. 5.3.3.

5.4. განტოლების ამოხსნა ვარიანტების გადარჩევით

განვიხილოთ $a+2b+3c+4d=30$ განტოლების ამოხსნა, სადაც a, b, c, d მთელი დადებითი რიცხვებია. რა თქმა უნდა a, b, c, d ყველა დასაშვები მნიშვნელობის განხილვა მიგვიყვანდა სწორ ამონახსნამდე, მაგრამ ამისათვის საჭირო იქნებოდა საკმაოდ დიდი დრო. შემოთავაზებული ალგორითმი სასურველ ამონახსნს პოულობს გაცილებით მცირე დროში. ალგორითმის არსი მდგომარეობს იმაში, რომ დასაწყისში ვირჩევთ რამოდენიმე შემთხვევით მნიშვნელობას და გადარჩევის გზით ვპოულობთ სასურველ ამონახსნს.

დასაწყისისათვის ავირჩიოთ ხუთი შემთხვევითი ამონახსნი: $1 =< a, b, c, d <= 30$.

ვარიანტი	***a, b, c, d***
1	(1,28,15,3)
2	(14,9,2,4)
3	(13,5,7,3)
4	(23,8,16,19)
5	(9,13,5,2)

ცხრ. 5.4.

იმის შესამოწმებლად, თუ რამდენად ახლოს არის ჩვენს მიერ მიღებული ამონახსნი ნამდვილ ამონახსნთან, ვიპოვოთ მიღებული ამონახსნის განსხვავება 30-თან. ჩავსვათ a, b, c, d მნიშვნელობები განტოლებაში, გამოვითვალოთ მიახლოვების კოეფიციენტები.

ვარიანტი	მიახლოვების კოეფიც.
1	|114-30|=84
2	|54-30|=24
3	|56-30|=26
4	|163-30|=133
5	|58-30|=28

ცხრ. 5.5.

რაც უფრო მცირეა სხვაობა, ვარიანტი მით უფრო ახლოს არის ამონახსნთან. ე.ი. ჩვენთვის საინტერესოა დაბალი მიახლოვების კოეფიციენტის მქონე მნიშვნელობები.

გამოვითვალოთ სხვაობათა კოეფიციენტების შებრუნებული

მნიშვნელობები და მათი ჯამი:

ვარიანტი	მიახლოვების კოეფიც.
1	1/84=0.011904
2	1/24=0.041666
3	1/26=0.038461
4	1/133=0.007518
5	1/28=0.035714
ჯამი	0.135266

ცხრ. 5.6.

გამოვიანგარიშოთ მიახლოვების კოეფიციენტების პროცენტული მნიშვნელობები:

ვარიანტი	მიახლოვების კოეფიციენტ.%
1	(1/84)/0.135266=8.8%
2	(1/24)/0.135266=30.8%
3	(1/26)/0.135266=28.4%
4	(1/133)/0.135266=5.56%
5	(1/28)/0.135266=26.4%

ცხრ. 5.7.

აქედან გამომდინარე, ყველაზე კარგი მიახლოვების კოეფიციენტი აქვს მეორე და მესამე ვარიანტებს. ამ ორი ვარიანტისაგან სხვადასხვა კომბინაციის გზით მივიღებთ რამოდენიმე ვარიანტს:

ვარიანტი	***a,b,c,d*** კომბინაციები
2	14 5 7 3
	14 9 7 3
	14 9 2 3
3	13 9 2 4
	13 5 2 4
	13 5 7 4

ცხრ. 5.8.

მიღებული ვარიანტებისათვის კვლავ გამოვითვალოთ მიახლოვების კოეფიციენტები. კომბინაცია (13, 5, 2, 4) დანარჩენ მნიშვნელობებთან შედარებით ყველაზე უფრო უახლოვდება შედეგს, რადგან მიახლოვების კოეფიციენტი უდრის 15-ს. შეგვიძლია მნიშვნელობები შევცვალოთ 1-დან 30-მდე შემთხვევითი რიცხვებით. მაგალითად, თუ *b*-ს მნიშვნელობას ჩავანაცვლებთ 3-ით, მიახლოვების კოეფიციენტი გახდება 11. *a*-ს მნიშვნელობა, ანუ რიცხვი 13, ჩავანაცვლოთ 2-ით, რაც გამოიწვევს მისი მნიშვნელობის 11-ით შემცირებას, რის შედეგადაც მიახლოვების კოეფიციენტი გახდება 0-ის ტოლი. *a, b, c, d* -ს მიღებული კომბინაცია (2, 3, 2, 4) არის მოცემული განტოლების ამონახსნი.

თუ თავიდან ავიღებდით შემთხვევით მნიშვნელობათა არა 5, არამედ, მაგალითად 50 ან მეტ ვარიანტს, შედეგი იქნებოდა უფრო სწრაფი და სტაბილური.

ლიტერატურა

1. *ცცცცც ც.ც. ცცცცცცცცცც. ცცცცცცცცცც ცცცცცც. - ც.: ცცცც, 1997.*

2. ცცცცც ც.ც., ცცცცც ც.ც. ცცცც ც.ც. ცცცც ც ცცცცცცცც. ც.:ABF, 1996.

3. ცცცც ც.ც. ცცცც ცცცცცცცც და ცცცცც-ცცცც ცცცც. - ც.:ცცცცცცც, 1989.

4. ცცცც ც. , ცცცცც ც. ცცცც და ცცცცცც ც ცცცც ცცცცცც. ც.1991.

5. თ. ბახტაძე, დ. თარხნიშვილი, ლ. კიკნაძე, ბ. მებუკე, ზ. ჩხაიძე. ინფორმატიკა, საქ. ტექნიკური უნივერსიტეტი, თბ. 1991.