ზ. წვერაიძე, ლ. გაჩეჩილაძე, ლ. ნონიკაშვილი

დაპროგრამების საფუძვლები

ალგორითმული ენა **Visual Basic**

მეთოდური მითითებები
ლაბორატორიული სამუშაოების შესასრულებლად

(I ნაწილი)

„ტექნიკური უნივერსიტეტი”

საქართველოს ტექნიკური უნივერსიტეტი

ზ. წვერაიძე, ლ. გაჩეჩილაძე, ლ. ნონიკაშვილი

დაპროგრამების საფუძვლები

ალგორითმული ენა **Visual Basic**

მეთოდური მითითებები
ლაბორატორიული სამუშაოების შესასრულებლად

(I ნაწილი)

რეკომენდებულია სტუ-ის
სარედაქციო-საგამომცემლო საბჭოს
მიერ. 21.01.2010, ოქმი №1

თბილისი
2010

განხილულია 13 ლაბორატორიული სამუშაოს შესრულების მეთოდიკა. ყოველი სამუშაოსთვის აღწერილია გრაფიკული ინტერფეისის დამუშავების, პროგრამული კოდის ჩაწერისა და Visual Basic – პროექტის შესრულების პროცედურები.

მოცემული მეთოდური მითითებანი განკუთვნილია დაპროგრამების საფუძვლების ასათვისებლად.

რეცენზენტი სრ. პროფ. ნ. ჯიბლაძე

ISBN 978-9941-14-798-2 (ყველა ნაწილი)
ISBN 978-9941-14-799-9 (პირველი ნაწილი)
http://www.gtu.ge/publishinghouse/

შესავალი

Visual Basic (ქვემოთ გამოვიყენებთ აბრევიატურას VB) დაპროგრამების პოპულარული სისტემაა, რომელიც საშუალებას იძლევა მარტივად და მოხერხებულად შეიქმნას ოპერაციული სისტემა Windows-ის პლატფორმაზე მომუშავე პროგრამები, ანუ, როგორც მათ უწოდებენ Windows-ის დანართები. მნიშვნელოვანია ის გარემოება, რომ ის თითქმის ნებისმიერი ხასიათის, მათ შორის ინტერნეტის შესაძლებლობების გამოყენების უნარის მქონე, დანართის შექმნის მძლავრი ინსტრუმენტული საშუალებაა.

Visual Basic-ის პროგრამული პროექტის მომზადება ხდება სპეციალურ გარემოში, რომელსაც VB-ის დაპროექტების გარემო ეწოდება. ის არის ეკრანული და დიალოგური ფანჯრების, მენიუების, ინსტრუმენტების ზოლების ერთობლიობა და ემსახურება გრაფიკული ინტერფეისის შექმნას, პროგრამული კოდის შედგენას, პროგრამის გამართვას და შესრულებას.

პროგრამული პროდუქტის შექმნის პროცესს ხშირად დაპროექტებას უწოდებენ, ხოლო თავად დაპროგრამების ზემოაღნიშნულ ენას – დაპროექტების სისტემას.

ამგვარად, VB, როგორც დაპროექტების სისტემა შეიცავს დაპროგრამების ენას Visual Basic, პროგრამის ტრანსლაციისა და გამართვის საშუალებებს, სამუშაო გარემოს, რომელშიც გრაფიკული ინტერფეისი შეიქმნება და მოხდება პროგრამის შედგენა და შესრულება.

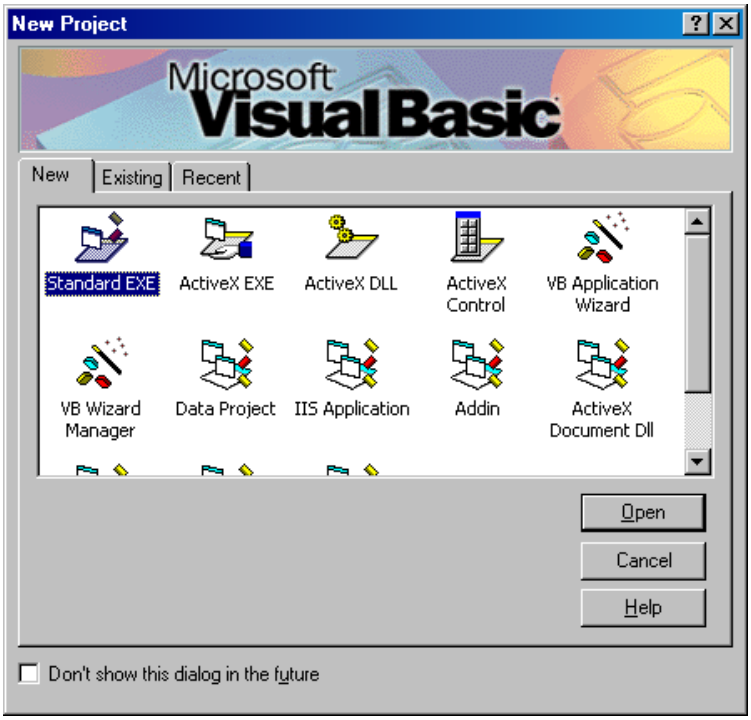
ლაბორატორიული სამუშაო №1
Visual Basic-ის სამუშაო გარემო

Visual Basic მიეკუთვნება ვიზუალური და ობიექტზე ორიენტირებული დაპროგრამების ენათა რიცხვს, რომელიც საშუალებას გვაძლევს შევქმნათ ოპერაციულ სისტემა **Windows**-ში მომუშავე პროგრამები.

VB-ის გაშვება ხდება **Start** მენიუს **Programs-Microsoft Visual Basic 6.0** ბრძანებით (აქ რიცხვითი მაჩვენებელი 6.0 მიუთითებს ვერსიის ნომერზე) ან თაგვის ორჯერ დაწკაპუნებით მის ნიშნაკზე, თუ კი ის მოთავსებულია ოპერაციულ სისტემა **Windows**-ის სამუშაო მაგიდაზე (**Desktop**). აღნიშნული პროცედურის შესრულების შედეგად ეკრანზე იხსნება **VB**-ის სამუშაო გარემო, რომლის წინა პლანზე თავს იჩენს **New Project** სახელის მქონე დიალოგური ფანჯარა (ის წარმოდგენილია 1-ელ ნახაზზე).

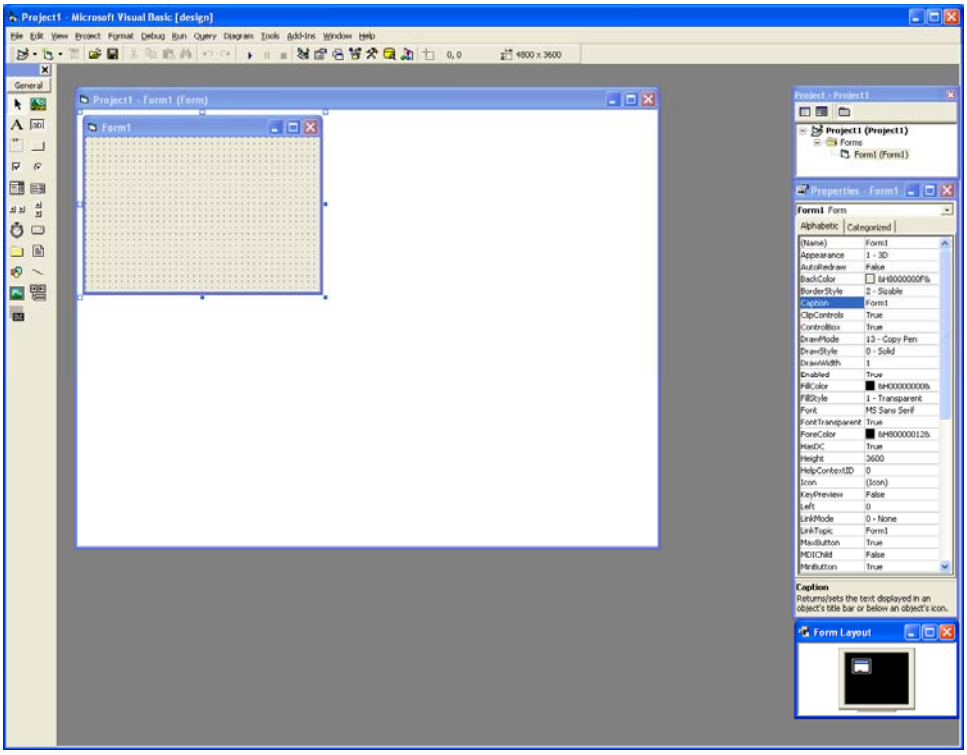
თუ აღნიშნულ დიალოგურ ფანჯარაში ჩავრთავთ ალამს **Don't show this dialog in the future**, მაშინ **VB**-ის მომდევნო გაშვებისას ის აღარ გამონათდება და პირდაპირ გადავალთ ახალი სტანდარტული პროექტის შექმნის პროცედურაზე.

1-ელ ნახაზზე წარმოდგენილი დიალოგური ფანჯრის **Existing** ბარათი გვიჩვენებს არსებული პროექტების სიას, ხოლო ბარათი სახელწოდებით **Recent –** ბოლოს გახსნილი პროექტების ქრონოლოგიურ ჩამონათვალს. ბოლო ორი ბარათის (**Existing** და **Recent**) ჩამონათვლებიდან სასურველი პროექტის ამორჩევისა და **Open** ღილაკზე დაჭერის შედეგად, იგი გაიხსნება სამუშაო გარემოს მთავარ ფანჯარაში (იხილეთ მე-2 ნახაზი).



ნახ. 1.

მთავარი ფანჯარა სამუშაო გარემოს ძირითადი ელემენტია, რომლის სათაურის სტრიქონში მითითებულია დაპროგრამების სისტემის დასახელება **Microsoft Visual Basic** და პროექტის სახელი. სათაურის სტრიქონში სიტყვა **[design]** მიუთითებს მუშაობის მიმდინარე (დაპროექტების) რეჟიმზე. როდესაც პროექტის შესრულებაზე გაშვება ხდება, ეს დასახელება იცვლება სიტყვით **[run]**, რაც პროექტის შესრულების ეტაპზე გადასვლას ნიშნავს.



ნახ. 2.

ვიზუალური დაპროგრამების მთავარი ცნება არის **გრაფიკული ინტერფეისი**, რომელიც ობიექტების მეშვეობით უზრუნველყოფს მომხმარებლისა და პროგრამის ურთიერთკავშირს, პროგრამის შესრულების პროცესის მართვას და სხვ.

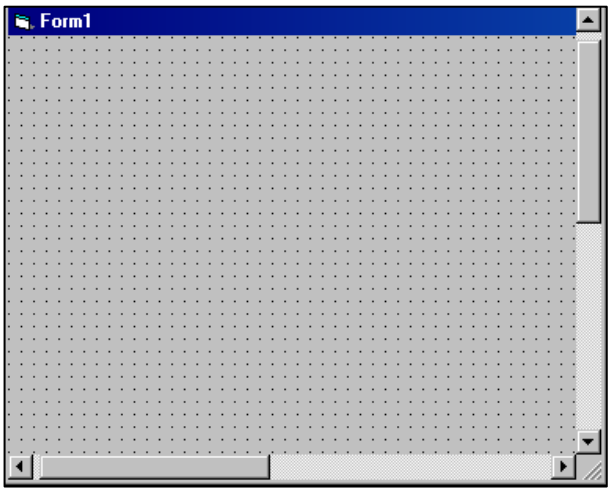
ვიზუალური დაპროგრამების სისტემებში პროგრამულ პროდუქტზე მუშაობა მისი გრაფიკული ინტერფეისის შექმნით დაიწყება. ის არის ეკრანული ფორმებისა და გამოსახულებების ერთობ-

ლიობა, რომელიც მოიცავს სხვადასხვა ღილაკებს, მენიუებს, ალმებს, გადამრთველებს, ინფორმაციულ ველებს და სხვ.

გრაფიკული ინტერფეისის მთავარ ელემენტებს ფორმა, მართვის ელემენტები და ობიექტები შეადგენს.

ფორმა (**Form**) გრაფიკული ინტერფეისის მთავარი ობიექტია, რომელზეც მისი დაპროექტება ხდება. ის არის მე-3 ნახაზზე ნაჩვენები ფანჯარა, სადაც თავსდება გრაფიკული ინტერფეისის ელემენტები. პროგრამული პროექტი შეიძლება ერთ ან რამდენიმე ფორმას შეიცავდეს.

მართვის ელემენტები (**Controls**) გრაფიკული ინტერფეისის მთავარი ინსტრუმენტებია. მათ მიეკუთვნება: ბრძანებითი ღილაკები, გადამრთველები, ალმები, მენიუები, ტექსტური და გრაფიკული ველები და სხვ.



ნახ. 3.

ფორმაზე მოთავსების შემდეგ მართვის ელემენტების მეშვეობით ხორციელდება პროგრამის მართვა. ისინი ასევე, გამოიყენება პროგრამაში მონაცემთა შეტანა-გამოტანის მიზნითაც.

ფორმაზე ელემენტების განთავსებას ემსახურება მე-4 ნახაზზე წარმოდგენილი ინსტრუმენტების ზოლი, რომლის ძირითადი შესაძლებლობებია:

ნახ. 4.

Pointer – მაჩვენებელი (მიმთითებელი);
PictureBox – ნახატის ჩარჩო (ფანჯარა);
Label – წარწერა (ჭდე);
TextBox – ტექსტური ველი;
Frame – ჩარჩო;
CommandButton – ბრძანებითი ღილაკი;
CheckBox – ალამი;
OptionButton – გადამრთველი;
ComboBox – კომბინირებული ფანჯარა;
ListBox – ჩამონათვალი;
HscrollBar – ჰორიზონტალური სქროლინგის ხაზი;
VscrollBar – ვერტიკალური სქროლინგის ხაზი;
Timer – დროის აღმრიცხველი (ტაიმერი);
DriveListBox – მოწყობილობათა სია;
DirListBox – საქაღალდეების სია;
FileListBox – ფაილების სია;
Shape – ფიგურა;
Line – ხაზი;
Image – გამოსახულება;
Data – მონაცემები;
OLE – ობიექტთა მოთავსების ფანჯარა.

მართვის ელემენტები ფორმაზე მოთავსების შემდეგ წარმოგვიდგება ობიექტების სახით.

ობიექტზე ორიენტირებულ დაპროგრამებაში მთავარი ელემენტია გარკვეული ქცევის მატარებელი **ობიექტი**, რომლის ქცევის წესს განსაზღვრავს თავად პროგრამა ანუ **პროგრამული კოდი**. ამ თვალსაზრისით მთელი პროგრამა წარმოგვიდგება როგორც ცალკეული ობიექტისათვის **შედგენილი პროგრამული პროცედურების** ერთობლიობა.

ყოველი ობიექტი ხასიათდება სამი კატეგორიის მახასიათებლებით: **თვისებებით, მეთოდებით და მოვლენებით**. ობიექტები კლასებში ერთიანდება.

კლასი ერთნაირი თვისებების, მეთოდებისა და მოვლენების მქონე ელემენტების ერთობლიობაა. კლასში შემავალ ყოველ ობიექტს **კლასის ეგზემპლარი** ეწოდება.

გრაფიკული ინტერფეისის ყოველი ობიექტი **თვისებათა** გარკვეული კომპლექტით ხასიათდება, რომელიც მის გარეგნულ სახეს განსაზღვრავს.

ობიექტების თვისებათა საწყისი მნიშვნელობები შეიძლება ორი გზით შევცვალოთ:

1. სამუშაო გარემოს **Properties** (თვისებები) სახელწოდების მქონე სპეციალურ ფანჯარაში, რომელიც წარმოდგენილია მე-5 ნახაზზე;
2. პროგრამის შესრულების პროცესში. (ამ შემთხვევაში ოპერაცია ობიექტის შესაბამის პროგრამულ კოდში აღიწერება).

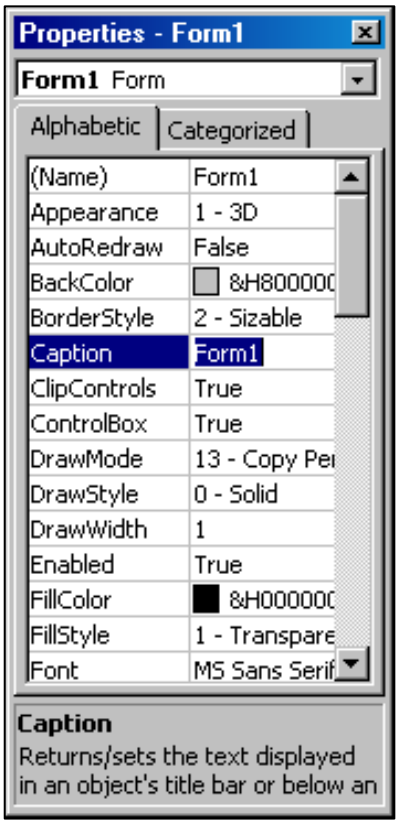
როგორც ზემოთ აღვნიშნეთ, ობიექტებს გარკვეული **მეთოდები** უკავშირდება, რომელთა მეშვეობით შესაძლებელია ობიექტების თვისებების შეცვლა და მათზე სხვადასხვა ოპერაციების შე-

სრულება. მეთოდები, თვისებებისგან განსხვავებით, ინტერფეისის შექმნის ეტაპზე არ გამოიყენება. მათთან მუშაობა შესაძლებელია მხოლოდ პროგრამიდან. მეთოდები იწერება, როგორც პროგრამის ოპერატორები, მაგრამ დაპროგრამების ენის ჩვეულებრივი ოპერატორებისგან განსხვავებით, ისინი მიბმულია გრაფიკული ინტერფეისის კონკრეტულ ობიექტებზე.

Visual Basic-ის დაპროგრამების ენაში გრაფიკული ინტერფეისის ობიექტები გარკვეულ **მოვლენაზე** რეაგირებს, რომლის ქვეშ შეიძლება ვიგულისხმოთ თაგვის ღილაკზე ან კლავიატურის რომელიმე კლავიშზე დაჭერა, თაგვის გადაადგილება, ფაილის გახსნა და სხვა.

მოვლენას შეიძლება მომხმარებელი იწვევდეს.

VB-ში პროგრამული პროცედურები ცალკეული ობიექტებისთვის იწერება, ხოლო მოვლენები კი მათ შესრულებას იწვევს. მაშასადამე, პროგრამის შესრულების პროცესის მართვა მოვლენათა საშუალებით ხორციელდება.

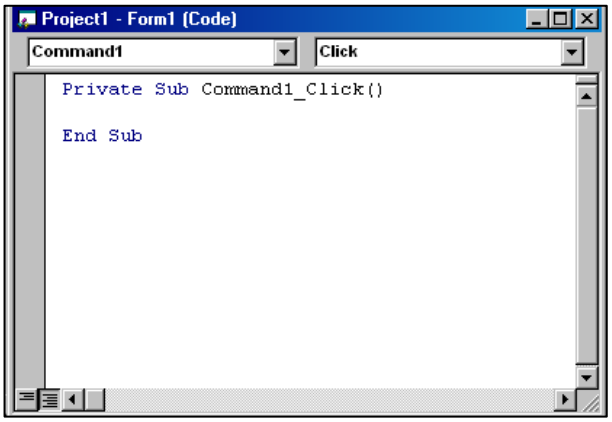


ნახ. 5.

ცალკეული პროგრამული პროცედურები **პროგრამულ მოდულებში** ერთიანდება. იგი, როგორც წესი, ცალკე არ არსებობს და ყოველთვის მიბმულია გარკვეულ გრაფიკულ ინტერფეისზე – ფორმასა და მის ობიექტებზე და ერთი კონკრეტული ამოცანის გადაწყვეტას ემსახურება.

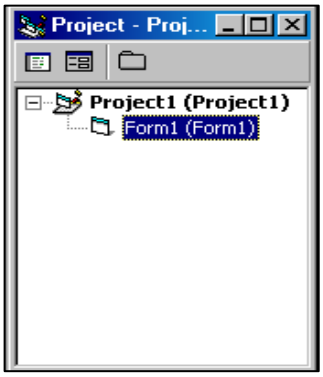
VB-ის ენაზე ჩაწერილ პროგრამის ტექსტს **პროგრამული კოდი** ეწოდება. თუ გრაფიკული ინტერფეისი პროექტის შესრულების გარემოს გარეგნულ მხარეს განსაზღვრავს, პროგრამული კოდი - მის შინაარსობრივ მხარეს, ანუ სხვა სიტყვებით რომ ვთქვათ, ის კონკრეტული ალგორითმის რეალიზაციას ემსახურება. პროგრამული კოდის ყოველ სტრიქონში შეიძლება მოთავსდეს ერთი ან ერთმანეთისგან ორწერტილით (:) გამოყოფილი რამდენიმე ოპერატორი.

პროგრამული კოდის აკრეფა ხდება სამუშაო გარემოს Code სახელის მქონე სპეციალურ ფანჯარაში, რომელიც წარმოდგენილია მე-6 ნახაზზე.



ნახ. 6.

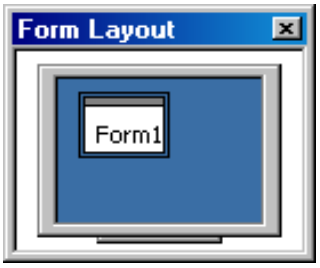
VB-ში შესაქმნელ პროგრამულ პროდუქტს **პროექტი** ეწოდება. ის მომხმარებლის მიერ შემუშავებული გრაფიკული ინტერფეისისა და პროგრამული მოდულის ერთობლიობაა. პროექტის სტრუქტურა სამუშაო გარემოს Project სახელწოდების მქონე ფანჯარაში აისახება, რომელიც წარმოდგენილია მე-7 ნახაზზე.



ნახ. 7.

ფორმა და მასთან დაკავშირებული პროგრამული მოდული ინახება **.frm** გაფართოების ფაილში, ხოლო პროექტი - **.vbp** გაფართოების მქონე ფაილში.

ეკრანზე ფორმის განთავსების ფანჯარა **Form Laout**, წარმოდგენილია მე-8 ნახაზზე.



ნახ. 8.

როგორც ადრე აღვნიშნეთ, ფორმა გრაფიკული ინტერფეისის მთავარი ელემენტია. მასთან სამუშაოდ შემდეგი ბრძანებები გამოიყენება.

- პროექტში **ფორმის დამატება** შესაძლებელია **Project** მენიუს **Add Form** ბრძანებით, ან ინსტრუმენტების ზოლზე **Add Form** ინსტრუმენტის ჩამონათვალიდან **Form** პუნქტის ამორჩევით;
- დახურული **ფორმის გახსნა** პროექტში შესაძლებელია **Project** ფანჯარაში (მე-7 ნახაზი შესავლიდან) პროექტის სტრუქტურულ ხეზე ფორმის დასახელებაზე ორჯერ დაჭერით;
- **ფორმის წაშლა** ხდება ბრძანებით **Project-Remove Form**, სადაც წინასწარ მონიშნულ იქნეს ის ფორმა, რომლის წაშლაც გვსურს. საჭიროა გვახსოვდეს, რომ ამ დროს ფორმასთან ერთად მისი შესაბამისი პროგრამული მოდულიც იშლება.

დასასრულს, ის ძირითადი თვისებები განვიხილოთ, რომლებიც ობიექტთა უმრავლესობას გააჩნია.

1. **Name -** სახელი. ის ფორმის ყველა ობიექტის თვისებაა და განსაზღვრავს მისი, როგორც გარკვეული კლასის ობიექტის დასახელებას. თავდაპირველად ობიექტებს სტანდარტული სახელები აქვთ, თუმცა აუცილებლობის შემთხვევაში შესაძლებელია მათი შეცვლა ობიექტის თვისებათა **Properties** ფანჯარაში.
2. **Caption –** სათაური. ის ობიექტის სათაურს, წარწერას განსაზღვრავს, რომელიც მასზეა მოთავსებული. ზოგ ელემენტს ეს თვისება არ აქვს. მისი მნიშვნელო-

ბის შეცვლა შესაძლებელია როგორც **Properties** ფანჯარაში, ისე პროგრამულადაც – მინიჭების ოპერატორის გამოყენებით.

3. **Visible –** ხილვადობა. თვისება, რომელსაც შესაძლოა ორიდან ერთი მნიშვნელობა ჰქონდეს: **True** – ჭეშმარიტი (ელემენტი ხილვადია) და **False** – მცდარი (ელემენტი უხილავია).
4. **Enable –** ელემენტის ხელმისაწვდომობა. მსგავსად წინა შემთხვევისა, მასაც შესაძლოა ორიდან ერთი მნიშვნელობა ჰქონდეს: **True** – ჭეშმარიტი (ელემენტი ხელმისაწვდომია) და **False** – მცდარი (ელემენტი არახელმისაწვდომია).
5. **Left, Top, Height, Width –** ეს თვისებები ფორმაზე ობიექტის პოზიციას განსაზღვრავს. **Left** და **Top** – ობიექტის მარცხენა ზედა კუთხის კოორდინატებს; **Height** და**Width –** ობიექტის სიმაღლესა და სიგანეს.
6. **Alignment –** თვისება, რომელიც ობიექტში უზრუნველყოფს არსებული ტექსტის ან გრაფიკის განთავსებას მისი კიდეების მიმართ. აქ შემდეგი ვარიანტები განიხილება: **0** – **Left Justify** (სწორება მარცხნივ), **1** – **Right Justify** (სწორება მარჯვნივ), **2** – **Center** (სწორება ცენტრში).
7. **Font –** შრიფტი. ის ობიექტის გასაფორმებლად გამოყენებული შრიფტის პარამეტრების მართვას უზრუნველყოფს.

8. **BackColor, ForeColor –** განსაზღვრავს შესაბამისად ობიექტის ფონისა და ობიექტში მოთავსებული ტექსტის ან გრაფიკის ფერს.
9. **Appearance –** ობიექტის გარეგნული სახე. მისი შესაძლო მნიშვნელობებია: **0 – Flat** (გაფორმება ვიზუალური ეფექტების გარეშე) და **0 – 1-3D** (სამგანზომილებიანი გაფორმება ვიზუალური ეფექტებით).
10. **ToolTipText –** კარნახის ტიპის ტექსტი, რომელიც განმარტავს ობიექტის დანიშნულებას.

პროექტის შენახვის (დამახსოვრების) მიზნით ინსტრუმენტების ზოლზე ვაწვებით ინსტრუმენტს სახელწოდებით **Save Project**, რის შემდეგაც გამოსულ დიალოგურ ფანჯარაში ავკრეფთ ფორმის ფაილის სასურველ სახელს (იგი შეინახება **.frm** გაფართოების მქონე ფაილში) და დავაჭერთ **Save** ღილაკს. აღნიშნული პროცედურის შესრულების შემდეგ ეკრანზე გამოდის მეორე დიალოგური ფანჯარა სახელწოდებით **Save Project As**, სადაც საჭიროა პროექტის ფაილის სახელწოდების მითითება და **Save** ღილაკის გამოყენება (პროექტის შენახვა ხდება **.vbp** გაფართოების მქონე ფაილში).

იმისათვის, რომ **პროექტის ფაილი ეგმ-ის მეხსიერებიდან** თავის კომპონენტებთან ერთად **წავშალოთ**, საჭიროა:

1. ავირჩიოთ **File** მენიუს **Open Project** ბრძანება;
2. გამოსულ დიალოგურ ფანჯარაში გავააქტიუროთ ბარათი სახელწოდებით **Existing**;
3. დავაჭიროთ მარჯვენა ღილაკს წასაშლელი პროექტის ფაილზე და კონტექსტურ მენიუში ავირჩიოთ ბრძანება **Delete.**

სამუშაო გარემოში **პროგრამის შესრულებაზე გაშვება** შესაძლებელია მთავარი ფანჯრის ინსტრუმენტების ზოლზე არსებული ▶ (**Start**) ღილაკის გამოყენებით ან **Run** მენიუს **Start** ბრძანებით. ■ (**End**) ღილაკზე დაჭერით ან **Run** მენიუს **End** ბრძანებით ხდება პროგრამის შესრულების პროცესის დამთავრება. ❚❚ (**Break**) ღილაკზე დაჭერით ან **Run** მენიუს **Break** ბრძანებით პროგრამის შესრულება წყდება.

შესრულებადი .exe ფაილის შესაქმნელად გამოიყენება **File** მენიუს **Make *.exe** ბრძანება. შემდგომში ამ პროგრამული ფაილის გაშვება შესაძლებელი იქნება ოპერაციულ სისტემა **Windows**-ში მიღებული ნებისმიერი ხერხით.

დავალება

Visual Basic-ის დაპროექტების სისტემაში შექმენით პროგრამული პროდუქტი, რომელიც ითვალისწინებს გრაფიკული ინტერფეისის ფორმაზე **Print** მეთოდის გამოყენებით პროექტის ავტორის სახელისა და გვარის გამოტანას ქართულ ენაზე და სასურველი სურათის ჩასმას. დავალების შესასრულებლად გამოიყენეთ:

ა) **Label1** ტიპის ობიექტი წარწერით - "პირველი პროექტი", რომელიც პროექტის შესრულებამდე ეკრანზე არ უნდა გამოჩნდეს;

ბ) პროგრამის შესრულება დაიწყეთ **Command1** ბრძანებით ღილაკზე თაგვის დაჭერით, რომლის სათაურიც იქნება "დასაწყისი" (მიმართეთ **Caption თვისებას**);

გ) სათანადო **პროგრამული ოპერატორის** გამოყენებით **Command1** ბრძანებით ღილაკზე წარწერის ("დასაწყისი") შრიფტის ზომა აიღეთ 12-ის ტოლი;

დ) სათანადო **პროგრამული ოპერატორის** გამოყენებით შეცვალეთ 16 ზომის შრიფტით ფორმაზე გამოსატანი პროექტის ავტორის სახელი და გვარი;

ე) **Image** ობიექტის თვისებების შეცვლით ფორმაზე განათავსეთ თქვენთვის სასურველი სურათი, რომელიც პროექტის შესრულებამდე ეკრანზე არ უნდა იყოს ხილვადი;

ვ) პროექტის გამართვისა და შესრულებაზე გაშვების შემდეგ სამუშაოს დასასრულს ჩაწერეთ თქვენ მიერ **Command1** ბრძანებითი ღილაკისთვის შექმნილი პროგრამული პროცედურა.

ზ) ინტერფეისის ობიექტთა თვისებების შესაცვლელად გამოიყენეთ შემდეგი ცხრილი:

ობიექტი	თვისება	თვისების საწყისი მნიშვნელობა	თვისების ახალი მნიშვნელობა
Form1	Caption	Form1	First Project
Command1	Caption	Command1	დასაწყისი
	Font(Name)	MS Sans Serif	AcadNusx
Label1	Caption	Label1	პირველი პროექტი
	Font(Name)	MS Sans Serif	AcadNusx
	Font(Size)	8	16
Image1	Picture	-	c:\..\My Picture
	Stretch	False	True

---------- ----------
(ქულა) (ხელმოწერა)

ლაბორატორიული სამუშაო №2
მონაცემთა ტიპები, გამოსახულებები და ფუნქციები დაპროგრამების ენა **Visual Basic**-ში

დაპროგრამების ენებში მონაცემები ის სიდიდეებია, რომლებიც პროგრამაში წარმოდგენილია ცხადი სახით ანუ კონკრეტული მნიშვნელობებით. განასხვავებენ მონაცემთა სამ ძირითად კლასს: რიცხვით, ტექსტურ და ლოგიკურ მონაცემებს.

რიცხვითი მონაცემები სხვადასხვა დიაპაზონისა და სიზუსტის რიცხვებია ნამდვილ რიცხვთა ღერძიდან. ათწილადი რიცხვების ჩაწერის დროს რიცხვის მთელი ნაწილი წილადი ნაწილისგან გამოყოფილია ათობითი წერტილით.

ტექსტური მონაცემები ენის ანბანის სიმბოლოთა ნებისმიერი ერთობლიობაა, რომელიც მოთავსებულია ორმაგ აპოსტროფებში (ბრჭყალებში).

ლოგიკური მონაცემები არის **True** (ჭეშმარიტი) და **False** (მცდარი) ლოგიკური მნიშვნელობები. იგი მიიღება ლოგიკური პირობების შემოწმების შედეგად.

Visual Basic-ში მონაცემთა სპეციფიკურ კლასს ობიექტთა თვისებების მნიშვნელობები ქმნის.

დაპროგრამების ენის უმარტივეს კონსტრუქციებს ცვლადები და კონსტანტები (მუდმივები) მიეკუთვნება.

თავისი ფიზიკური არსით ცვლადი მეხსიერების არეა, რომელსაც გარკვეული სახელი (იდენტიფიკატორი) გააჩნია. ცვლადის სახელი უნდა იწყებოდეს ასოთი; მასში დასაშვებია ასოების, ციფრების და ხაზგასმის ("_") სიმბოლოთა გამოყენება. ცვლადის სახელში სიმბოლოების რაოდენობა არ უნდა აღემატებოდეს 255-ს და დაუშვებელია (**Space**) ინტერვალის გამოყენება. ცვლადების სა-

ხელებად არ შეიძლება დაპროგრამების ენის ოპერატორების და ობიექტის თვისებების აღმნიშვნელი სახელების გამოყენება.

მეხსიერება, რომელსაც ცვლადი მიმართავს, სხვადასხვა ტიპის მონაცემებს შეიძლება შეიცავდეს.

VB-ში განასხვავებენ მონაცემებისა და ცვლადების შემდეგ ძირითად ტიპებს:

1. **Byte** – მთელი ტიპის რიცხვითი მონაცემები 0 ÷ 255 დიაპაზონიდან (ეგმ-ის მეხსიერებაში ერთ ბაიტს ანუ მეხსიერების ერთ უჯრას იკავებს);
2. **Integer** – მთელი ტიპის რიცხვითი მონაცემები **-** 32768÷32767 დიაპაზონიდან (ეგმ-ის მეხსიერებაში იკავებს ორ ბაიტს);
3. **Long –** გრძელი მთელი ტიპის რიცხვითი მონაცემები დიაპაზონიდან 2147483648÷2147483647 (ეგმ-ის მეხსიერებაში იკავებს ოთხ ბაიტს);
4. **Single –** სტანდარტული სიზუსტის (ექვსი ნიშანი ათობითი წერტილის შემდეგ) როგორც დადებითი ასევე უარყოფითი რიცხვები დიაპაზონიდან $1.4013*10^{-45} \div 3.4028*10^{38}$ (ეგმ-ის მეხსიერებაში იკავებს ოთხ ბაიტს);
5. **Double –** ორმაგი სიზუსტის (თოთხმეტი ნიშანი ათობითი წერტილის შემდეგ) ნამდვილი რიცხვები $4.940656484*10^{-324} \div 1.79769\,31349*10^{308}$ დიაპაზონიდან (ეგმ-ის მეხსიერებაში იკავებს რვა ბაიტს);
6. **String –** ტექსტური (სტრიქონული) მონაცემები, რომლებიც ყოველთვის ბრჭყალებში თავსდება (ეგმ-ის მეხსიერებაში ყოველი სიმბოლო ერთ ბაიტში ინახება). გვაქვს მისი კერძო შემთხვევაც **String*n**, სადაც **n** სტრიქონულ მონაცემში სიმბოლოთა მაქსიმალური დასაშვები რაოდენობაა;

7. **Boolean –** ლოგიკური მონაცემების ტიპი. პროგრამებში ორი შესაძლო მნიშვნელობიდან იღებს ერთ-ერთს. კერძოდ, 1 (**True**) ან 0 (**False**). მეხსიერებაში იკავებს ორ ბაიტს;
8. **Currency –** ნამდვილი რიცხვები წარმოდგენილი ფულადი ერთეულის სტილში (ფულად ფორმატში). ამ ტიპის რიცხვებში მთელი ნაწილი შეიძლება 15 ნიშანს შეიცავდეს, ხოლო წილადი ნაწილი – 4 თანრიგს. (ეგმ-ის მეხსიერებაში იკავებს რვა ბაიტს). ძირითადად გამოიყენება ფინანსური ხასიათის ინფორმაციის დამუშავების დროს;
9. **Date –** ტიპის მონაცემები შეიცავს ინფორმაციას თარიღისა და დროის შესახებ;
10. **Variant –** მონაცემთა განსაკუთრებული ტიპი, რომლის წინასწარ განსაზღვრული მნიშვნელობა არ არსებობს. მან შეიძლება მიიღოს ზემოთ ჩამოთვლილი ნებისმიერი ტიპის მნიშვნელობა. (ეგმ-ის მეხსიერებაში თექვსმეტ ბაიტს იკავებს, რის გამოც მისი გამოყენება პროგრამებში შეზღუდულია).

ცნობილია, რომ ცვლადები პროგრამებში გამოყენებამდე წინასწარ აღწერილ იქნეს, რათა მოხდეს ეგმ-ის მეხსიერებაში არეების გამოყოფა შესაბამისი ტიპის მონაცემთა მოსათავსებლად. თუ ცვლადებს წინასწარ არ აღვწერთ, მათ ავტომატურად მიენიჭებათ **Variant** ტიპი, რაც მეხსიერების არაეკონომიურ გამოყენებას გამოიწვევს.

ცვლადების აღწერის ორი ხერხი არსებობს: **Dim** ოპერატორით წინასწარი აღწერა და პროგრამის ტექსტში ცვლადის ტიპის შესაბამისი იდენტიფიკატორის გამოყენებით.

ცვლადების წინასწარი აღწერის **Dim** ოპერატორს აქვს შემდეგი სახე:

Dim ცვლადი As ტიპი

Dim და **As** ოპერატორული სიტყვებია; დასაშვებია ერთი **Dim** ოპერატორის მოქმედების არეში რამდენიმე სხვადასხვა ცვლადის აღწერა, მაგრამ იმ შემთხვევაში, თუ თითოეულ ცვლადს ექნება მითითებული თავისი ტიპი.

მაგალითად:

Dim X As Integer

Dim Name As String

Dim A As Integer, B As Integer, C As Long, D As Double

როდესაც ცვლადების აღწერა მათი შესაბამისი ტიპის იდენტიფიკატორებით ხდება, ამ შემთხვევაში სპეციალური ნიშნები (ტიპის იდენტიფიკატორები) გამოიყენება, რომლებიც ცვლადის სახელის ბოლოს მოთავსდება. ტიპის იდენტიფიკატორი ყველა ტიპს არ აქვს. ამ წესით შეიძლება მხოლოდ ექვსი ტიპის ცვლადი აღვწეროთ, რომელთა დასახელება და შესაბამისი იდენტიფიკატორები წარმოდგენილია 1-ელ ცხრილში.

ცხრილი 1

ტიპი	**Integer**	**Long**	**Single**	**Double**	**String**	**Cuurency**
იდენტიფიკატორი	**%**	**&**	**!**	**#**	**$**	**@**

მაგ.: Dim X% იგივეა, რაც ჩანაწერი **Dim X As Integer**

Dim Name$ იგივეა, რაც ჩანაწერი **Dim Name As String**

ცვლადების დიაპაზონის აღწერა შესაძლებელია **DefType** ოპერატორით, რომლის ჩაწერის ზოგადი სტრუქტურა შემდეგია:

DefType დიაპაზონის საწყისი ასო – დიაპაზონის ბოლო ასო

DefType ოპერატორის კონკრეტული ფორმები მოყვანილია მე-2 ცხრილში.

ცხრილი 2

ცვლადის ტიპი	DefType ოპერატორის ფორმა
Byte	**DefByte**
Integer	**DefInt**
Long	**DefLng**
Single	**DefSng**
Double	**DefDbl**
String	**DefStr**
Boolean	**DefBool**
Currency	**DefCur**
Date	**DefDate**
Variant	**DefVar**

მაგალითად: DefInt A-Z ოპერატორით ყველა ცვლადს, რომელიც იწყება **A-Z** დიაპაზონის ნებისმიერი ასოთი, ექნება ტიპი **Integer**.

კონსტანტები ის მუდმივი სიდიდეებია, რომელთა მნიშვნელობების შეცვლა პროგრამის შესრულების პროცესში დაუშვებელია. მათ აღწერას **Const** ოპერატორი ემსახურება, რომლის ჩაწერის ზოგადი სახე ასეთია:

Const სახელი As ტიპი = მნიშვნელობა

მაგალითად:

Const Pi As Double=3.14159265358979

Const Name As String=”Lela Gachechiladze”

VB-ში განასხვავებენ არითმეტიკულ, ლოგიკურ და სტრიქონულ გამოსახულებებს. მე-3 ცხრილში წარმოდგენილია არითმეტიკულ გამოსახულებებში გამოყენებული არითმეტიკული ოპერა-

ციების ნიშნები, ხოლო მე-4 ცხრილში - ლოგიკურ გამოსახულებებში გამოყენებული ლოგიკური ოპერაციების ნიშნები.

ცხრილი 3

არითმეტიკული ოპერაცია	**არითმეტიკული ოპერაციის ჩანაწერი VB-ში**
შეკრება	+
გამოკლება	-
გამრავლება	*
გაყოფა	/
ახარისხება	^
მთელრიცხვული გაყოფა	\
მთელრიცხვული გაყოფის ნაშთი	**Mod**

ცხრილი 4

ლოგიკური ოპერაცია	**ლოგიკური ოპერაციის ჩანაწერი VB-ში**
ტოლია	=
არ არის ტოლი	< >
მეტია	>
ნაკლებია	<
მეტია ან ტოლი	> =
ნაკლებია ან ტოლი	< =

უფრო რთული ლოგიკური გამოსახულებების ასაგებად გამოიყენება ლოგიკური ფუნქციები **And** (ლოგიკური გამრავლება) და **Or** (ლოგიკური შეკრება).

არითმეტიკულ გამოსახულებებში გამოყენებული სტანდარტული ფუნქციები წარმოდგენილია მე-5 ცხრილში. არითმეტიკულ გამოსახულებებში ოპერაციათა შესრულების პრიორიტეტი ისეთივეა, როგორც მათემატიკაში: ფუნქციის მნიშვნელობის გამოთვლა, ახარისხება, გამრავლება, გაყოფა, შეკრება და გამოკლება. ეს კანონზომიერება ირღვევა ფრჩხილების გამოყენების შემთხვევაში.

ცხრილი 5

სტანდარტული ფუნქციები	**სტანდარტული ფუნქციების ჩანაწერი VB-ში**
sina-ს გამოთვლა	**sin(a)**
cosa-ს გამოთვლა	**cos(a)**
tga-ს გამოთვლა	**tan(a)**
arctga-ს გამოთვლა	**atan(a)**
კვადრატული ფესვი	**sqr(a)**
ლოგარითმი e-ს ფუძით	**log(a)**
$\mathbf{e}^{\mathbf{a}}$ ფუნქციის გამოთვლა	**exp(a)**
a-ს მოდული	**abs(a)**

<u>დავალება</u>

VB-ის დაპროექტების სისტემაში შექმენით პროგრამული პროდუქტი, რომელიც შემდეგი ამოცანების გადაწყვეტას ითვალისწინებს:

1. მართკუთხედის ფართობისა და პერიმეტრის გამოთვლას;
2. სამკუთხედის ფართობის გამოთვლას ჰერონის ფორმულის საფუძველზე;
3. მართკუთხა პარალელეპიპედის მოცულობის გამოთვლას.

პროგრამაში საწყისი მონაცემების შესაყვანად გამოიყენეთ მინიჭების ოპერატორი, ხოლო მიღებული შედეგების გამოტანა გრაფიკული ინტერფეისის ფორმაზე **Print** მეთოდით მოახდინეთ. ცვლადების აღსაწერად I და II ამოცანებში გამოიყენეთ **Dim** ოპერატორი, ხოლო III ამოცანაში ამ მიზნით მიმართეთ ცვლადების ტიპების შესაბამის იდენტიფიკატორებს. პროგრამის შესრულებაზე გაშვება განახორციელეთ ბრძანებითი ღილაკის გამოყენებით, რომლის სათაურია "დასაწყისი". შექმენით შესრულებადი (**.exe** გაფართოების) ფაილი.

პროგრამული კოდი და მიღებული შედეგები:

---------- ----------
(ქულა) (ხელმოწერა)

ლაბორატორიული სამუშაო №3.
მონაცემთა შეტანა-გამოტანა
TextBox ობიექტის გამოყენებით

პროგრამაში ინფორმაციის შეტანა-გამოტანის მოხერხებული საშუალება არის გრაფიკული ინტერფეისის **TextBox** (ტექსტური ველი) ობიექტი. ფორმაზე მოთავსებისას ამ კლასის ობიექტებს ენიჭება სახელები: **Text1, Text2,** და ა.შ.

ობიექტის მთავარი თვისებაა **Text** თვისება, რომელიც საზოგადოდ ტექსტის მნიშვნელობას იღებს. ეს მნიშვნელობა მონაცემთა შეტანისას ჩვენ მიერ აიკრიფება კლავიატურაზე, გამოტანისას კი ტექსტურ ველში პროგრამის მეშვეობით მოთავსდება.

TextBox ველში მონაცემთა შეტანის ოპერატორს შემდეგი სახე აქვს:

ცვლადი = ობიექტი.Text

"ობიექტი" **TextBox** ტიპის ობიექტის დასახელებაა (მაგ. **Text1, Text2**). აღნიშნული ოპერატორით ცვლადს მიენიჭება ტექსტურ ველში აკრებილი (მოთავსებული) ტექსტის ანუ ობიექტის **Text** თვისების მნიშვნელობა.

ტექსტურ ველში მონაცემთა გამოტანის ოპერატორს აქვს შემდეგი სახე:

ობიექტი.Text = მნიშვნელობა

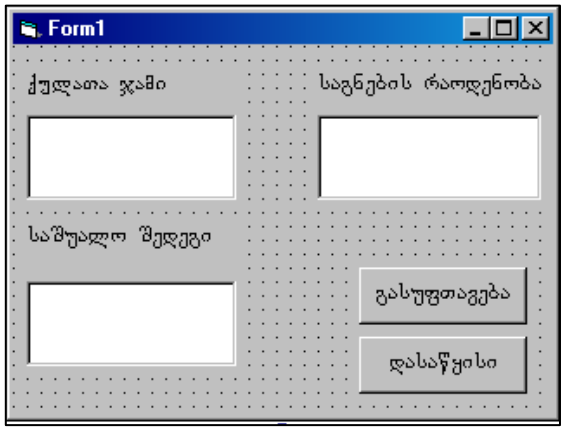
ამ ბრძანებით ტექსტურ ველში აისახება მინიჭების ოპერატორის მარჯვენა ნაწილში მდგომი მნიშვნელობა.

პროგრამის შესრულების პროცესში შეიძლება დაგვჭირდეს შევსებული ტექსტური ველების გასუფთავება მათში ახალი მონაცემების შეტანა-გამოტანის მიზნით. ამ ოპერაციას შემდეგი ოპერატორი ემსახურება: **ობიექტი.Text = Clear**

დავალება

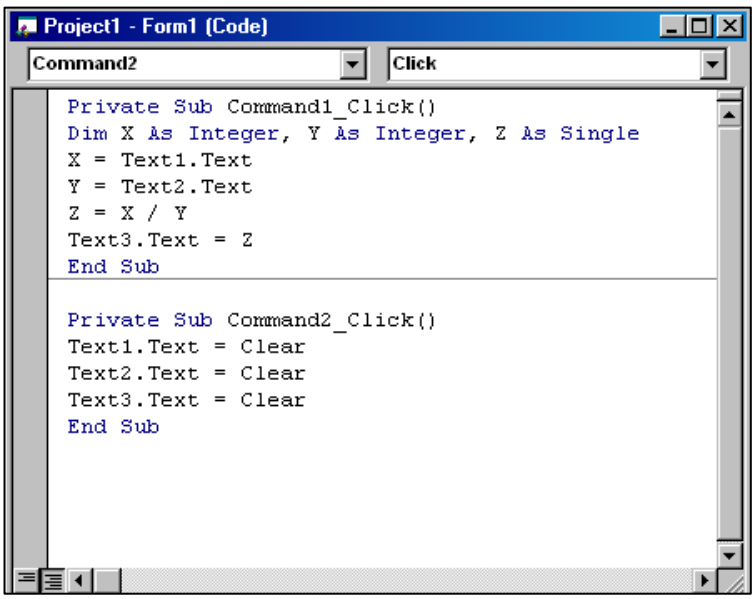
VB-ის დაპროექტების სისტემაში შექმენით პროგრამული პროდუქტი, რომელიც **Text1** და **Text2** ტექსტურ ველებში შეიტანს გარკვეული პერიოდის მანძილზე სტუდენტის მიერ დაგროვილ ქულათა ჯამს და საგნების რაოდენობას. ქულათა ჯამის საგნების რაოდენობაზე გაყოფით განისაზღვრება სტუდენტის მოსწრების საშუალო შედეგი. ტექსტურ ველებს გაუკეთეთ სათანადო წარწერა, რისთვისაც გამოიყენეთ **Label** ტიპის სამი ობიექტი დასათაურებით: "ქულათა ჯამი", "საგნების რაოდენობა" და "საშუალო შედეგი". პროგრამული პროცედურა შეადგინეთ **Command1** და **Command2** ბრძანებითი ღილაკებისთვის, რომელთაგან I ბრძანებით ღილაკს მიეცით სათაური წარწერით "დასაწყისი" (ის მოახდნეს პროგრამის შესრულებას), ხოლო II ბრძანებით ღილაკს – სათაური "გასუფთავება" (ის მოახდენს ტექსტური ველების მონაცემთა წაშლას).

ფორმაზე განთავსებული ობიექტების სათანადო თვისებების (**Caption, Font**) შეცვლის შემდეგ ფორმას 1-ელ ნახაზზე წარმოდგენილი სახე უნდა ჰქონდეს:



ნახ. 1.

ხოლო პროგრამულ პროცედურებს, რომლებიც უნდა შეადგინოთ ორივე ბრძანებითი ღილაკისთვის – მე-2 ნახაზზე ნაჩვენები სახე ექნება:



ნახ. 2.

მიღებული შედეგები:

---------- ----------

(ქულა) (ხელმოწერა)

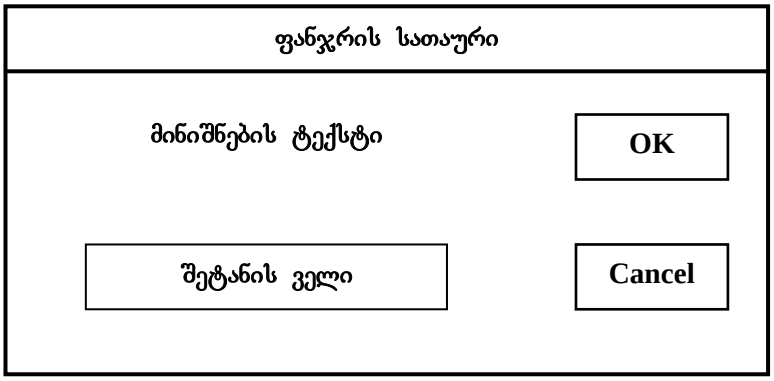
ლაბორატორიული სამუშაო №4
მონაცემთა შეტანა
InputBox ოპერატორ-ფუნქციის გამოყენებით

პროგრამებში საწყისი მონაცემების შეტანის უმარტივესი გზაა მინიჭების ოპერატორის გამოყენება, რომლის ნაკლი იმაში მდგომარეობს, რომ საწყისი მონაცემების სხვა კომპლექტისათვის პროგრამის შესასრულებლად გვჭირდება მასში ცვლილებების განხორციელება და ხელახალი ტრანსლაცია.

VB-ში არსებობს მონაცემთა მნიშვნელობების შეტანის **InputBox** ფუნქცია, რომელიც საშუალებას გვაძლევს თავად შევიტანოთ სასურველი საწყისი მონაცემები პროგრამის მოთხოვნით გახსნილ დიალოგურ ფანჯარაში. ამ ტიპის კონსტრუქციას ოპერატორ-ფუნქციას უწოდებენ და მისი ზოგადი სახე შემდეგია:

ცვლადი = **InputBox(**"მინიშნება", "სათაური", "მნიშვნელობა"**)**

InputBox ფუნქციის მოქმედების შედეგია 1-ელ ნახაზზე ნაჩვენები დიალოგური ფანჯარა.



ნახ. 1.

შეტანის ველში აკრეფილი მონაცემის მნიშვნელობა **OK** ღილაკზე დაჭერის შემდეგ ოპერატორ-ფუნქციის მარცხენა მხარეს მდგომ ცვლადს მიენიჭება. **InputBox** ფუნქციას სამი არგუმენტი აქვს, რომელთაგან პირველი (მინიშნება) აუცილებელია, ხოლო დანარჩენი ორი – შეიძლება არ გამოვიყენოთ.

მინიშნება InputBox ფუნქციის სავალდებულო არგუმენტია, რომელიც არის ორმაგ აპოსტროფებში მოთავსებული ტექსტი (შესატანი მონაცემის მოკლე აღწერილობა);

სათაური არის დიალოგური ფანჯრის სათაურის სტრიქონში მოთავსებული ტექსტი (ის უფრო ზოგადი ხასიათისაა);

მნიშვნელობა არააუცილებელი არგუმენტია, რომლის მნიშვნელობა შეტანის ველში ლურჯ ფონზე ჩანს და შეგვიძლია შევცვალოთ.

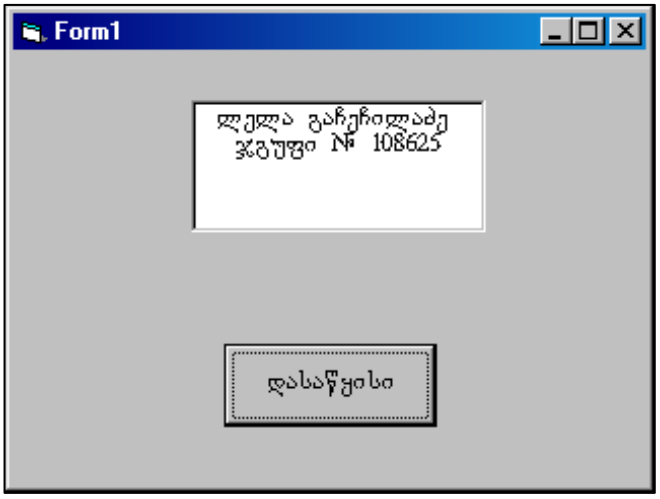
ძირითადად გამოიყენება **InputBox** ფუნქციის ერთ და ორ არგუმენტიანი ფორმები.

დავალება

VB-ის დაპროექტების სისტემაში შექმენით პროგრამული პროდუქტი, რომელიც ორ არგუმენტიანი **InputBox** ფუნქციის რამდენჯერმე გამოყენებით სტუდენტის გვარს, სახელს (ქართულ ენაზე), ჯგუფის ნომერს შეიტანს. მონაცემებს გააერთიანებს სტრიქონთა კონკატენაციის (გაერთიანების) "**&**" ოპერაციით და მიღებულ შედეგს გამოიტანს **TextBox** ობიექტის ტექსტურ ველში.

ამოცანის გადაწყვეტის შესაბამისი პროგრამული კოდი ქვემოთაა წარმოდგენილი, ხოლო მიღებულ შედეგს უნდა ჰქონდეს მე-2 ნახაზზე ნაჩვენები სახე:

```
Private Sub Command1_Click( )
Dim Name As String, LastName As String, Group As String
Dim Result As String
Name = InputBox("Input Name", "Input Data")
LastName = InputBox("Input Last Name", "Input Data")
Group = InputBox("Input Group", "Input Data")
Result = Name & " " & LastName &" "& "jgufi # " & Group
Text1.Text = Result
End Sub
```



ნახ. 2.

---------- ----------

(ქულა) (ხელმოწერა)

ლაბორატორიული სამუშაო №5
ინფორმაციის გამოტანა
MsgBox ოპერატორის გამოყენებით

ოპერატორი **MsgBox** გამოიყენება სპეციალურ ფანჯარაში ინფორმაციის გამოსატანად. მას ჩაწერის შემდეგი სინტაქსი გააჩნია:

MsgBox შეტყობინება, რიცხვითი კოდები, "სათაური"

შეტყობინება ოპერატორის აუცილებელი პარამეტრია და განსაზღვრავს ფანჯარაში გამოსატან ინფორმაციას. ამ პარამეტრს შეიძლება წარმოადგენდეს რაიმე ტექსტი ან ცვლადის სახელი.

სათაური არააუცილებელი პარამეტრია და არის **MsgBox** ფანჯრის სათაური.

რიცხვითი კოდები განსაზღვრავს შეტყობინების ფანჯრის ტიპს და მისი მართვის მიზნით საჭირო ღილაკების კომპლექტს. ის წარმოდგენილია ორშესაკრებიანი ჯამის სახით. პირველი შესაკრები ფანჯრის ტიპის განმსაზღვრელი პიქტოგრამის რიცხვითი კოდია, ხოლო მეორე – ღილაკთა კომპლექტის კოდი.

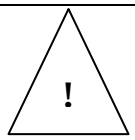
სხვადასხვა ტიპის **MsgBox** ფანჯრების შესაბამისი პიქტოგრამები და რიცხვითი კოდები წარმოდგენილია 1-ელ ცხრილში, ხოლო შეტყობინების ფანჯრის ღილაკთა შესაძლო კომპლექტები და შესაბამისი რიცხვითი კოდები ასახულია მე-2 ცხრილში.

მაგალითად: განვიხილოთ შემდეგი ოპერატორი:

MsgBox "Are You Ready", 32+4, "Message"

ამ ოპერატორში პირველი შესაკრები (კოდი 32) აღნიშნავს შეკითხვის ტიპის ფანჯარას, ხოლო მეორე (კოდი 4) შეესაბამება ღილაკთა კომპლექტს **Yes, No.**

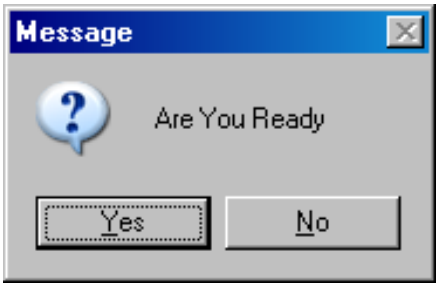
ცხრილი 1

პიქტოგრამა	რიცხვითი კოდი	ფანჯრის ტიპი
	16	კრიტიკული შეტყობინება
	32	შეკითხვა
	48	გაფრთხილება
	64	ინფორმაცია

ცხრილი 2

ღილაკთა კომპლექტი	რიცხვითი კოდი
OK	**0**
OK, Cancel	**1**
Abort, Retry, Ignore	**2**
Yes, No, Cancel	**3**
Yes, No	**4**
Retry, Cancel	**5**

აღნიშნული ოპერატორის შესრულების შედეგად ეკრანზე გახსნილ **MsgBox** ფანჯარას 1-ელ ნახაზზე წარმოდგენილი სახე ექნება:



ნახ. 1.

პროგრამის გასაგრძელებლად ამ ფანჯრის დახურვა აუცილებელია ერთ-ერთ ღილაკზე დაჭერით.

ოპერატორის ჩანაწერში შესაკრებთა ჯამის ნაცვლად შეიძლება ჯამური კოდიც მივუთითოთ. მაგალითად, ჩვენ მიერ განხილულ ოპერატორში **32+4** პარამეტრის ნაცვლად შეიძლება ჩავწეროთ ჯამური კოდი **36.**

თუ განვიხილავთ ორპარამეტრიან **MsgBox** ფანჯარას, სადაც გამოტოვებულია რიცხვითი კოდები, მათ ნაცვლად აუცილებელია თავისუფალი პოზიციის შენარჩუნება (მძიმეების საშუალებით); ამ შემთხვევაში მიიღება მარტივი ფანჯარა, რომელიც შეიცავს გამოსატან შეტყობინებას, ფანჯრის სათაურს და **OK** ღილაკს, რომელიც ფანჯრის დახურვისთვის აუცილებელია.

დავალება

შეადგინეთ პროგრამული კოდი, რომელიც **InputBox** ფანჯარაში შეიტანს ორგანიზაციის შემოსავალს, გამოთვლის დამატებითი ღირებულების გადასახადს, როგორც შემოსავლის 20%-ს და ორპარამეტრიან **MsgBox** ფანჯარაში მიღებულ შედეგს გამოიტანს.

ამოცანის გადაწყვეტის საწყის ფანჯარას უნდა ჰქონდეს მე-2 ნახაზზე წარმოდგენილი სახე.

ნახ. 2.

პროგრამული კოდი და მიღებული შედეგები:

(ქულა) (ხელმოწერა)

ლაბორატორიული სამუშაო №6
განშტოებადი ალგორითმების დაპროგრამება

წრფივი სტრუქტურის ალგორითმები ბუნებაში ძალზე მცირეა. ძირითადად განშტოებადი და ციკლური სტრუქტურის ალგორითმებს ვხვდებით.

განშტოებათა დაპროგრამებას **VB**-ის დაპროექტების სისტემაში მართვის გადაცემის შემდეგი ოპერატორებია:

1. **If – Then** პირობითი გადასვლის ოპერატორი, რომლის მარტივი ფორმის ზოგადი სახე შემდეგია:

If პირობა Then ოპერატორი

პირობა ლოგიკური გამოსახულებაა, რომლის მნიშვნელობა შეიძლება ჭეშმარიტი (True) ან მცდარი (False) იყოს. აღნიშნული ოპერატორის მოქმედების პრინციპი შემდეგში მდგომარეობს: პირველ რიგში პირობა მოწმდება, თუ იგი ჭეშმარიტია, პროგრამაში სრულდება **Then** საკვანძო სიტყვის მომდევნო ოპერატორი, წინააღმდეგ შემთხვევაში, ის არ შესრულდება და მართვა გადაეცემა მომდევნო სტრიქონზე მდგომ ოპერატორს.

ზემოაღნიშნული ოპერატორის ჩაწერის ბლოკური ფორმაც არსებობს, რომელსაც შემდეგი სახე აქვს:

If პირობა Then

ოპერატორი

. . . . . .

ოპერატორი

End If

საზოგადოდ, **If – Then** პირობითი გადასვლის ოპერატორის ბლოკური ფორმა უნივერსალურია და უფრო ხშირად გამოიყენება.

2. **If – Then - Else** პირობითი გადასვლის ოპერატორი, რომლის მარტივი ფორმის ზოგადი სახეა:

If პირობა Then ოპერატორი-1 Else ოპერატორი-2

აღნიშნული ოპერატორის მოქმედების პრინციპი შემდეგში მდგომარეობს: პირველ რიგში პირობა მოწმდება, თუ ის ჭეშმარიტია, სრულდება **Then** საკვანძო სიტყვის მომდევნო "ოპერატორი-1" პროგრამაში და არ სრულდება **Else** საკვანძო სიტყვის მომდევნო "ოპერატორი-2", წინააღმდეგ შემთხვევაში (თუ პირობა მცდარია) - არ სრულდება **Then** საკვანძო სიტყვის მომდევნო "ოპერატორი-1" და სრულდება **Else** საკვანძო სიტყვის მომდევნო "ოპერატორი-2".

ზემოაღნიშნული ოპერატორის ჩაწერის ბლოკური ფორმაც არსებობს, რომელსაც შემდეგი სახე აქვს:

If პირობა Then

ოპერატორი-1

Else

ოპერატორი-2

End If

3. **If – Then – ElseIf - Else** პირობითი გადასვლის ოპერატორი, რომლის ზოგადი სახე ასეთია:

If პირობა-1 Then

ოპერატორი-1

ElseIf პირობა-2 Then

ოპერატორი-2

. . . . . .

Else

ოპერატორი-*n*

End If

ზემოაღნიშნული ოპერატორი ორზე მეტი მიმართულების მქონე ალგორითმების დაპროგრამების დროს გამოიყენება.

ოპერატორის მოქმედების პრინციპი შემდეგია: თუ პირობა-1 ჭეშმარიტია, სრულდება "ოპერატორი-1" და სხვა ოპერატორები აღნიშნული ბლოკიდან არ სრულდება; თუ პირობა-1 მცდარია, მოწმდება პირობა-2, თუ ის ჭეშმარიტია, სრულდება "ოპერატორი-2" და სხვა ოპერატორები აღნიშნული ბლოკიდან არ სრულდება და ა.შ. თუ ბლოკში ყველა პირობა მცდარია, მხოლოდ ამ შემთხვევაში შესრულდება "ოპერატორი - ***n***".

4. **Iif** ფუნქცია, რომლის ჩაწერის სინტაქსი ასეთია:

ცვლადი = Iif(პირობა, მნიშვნელობა-1, მნიშვნელობა-2)

თუ პირობა ჭეშმარიტია, მაშინ ცვლადს მიენიჭება "მნიშვნელობა-1", წინააღმდეგ შემთხვევაში – "მნიშვნელობა-2".

5. **GoTo** უპირობო გადასვლის ოპერატორი;

6. **Selsect Case** მრავალჯერადი ამორჩევის სტრუქტურა. (ბოლო ორ ოპერატორს შემდეგ ლაბორატორიულ სამუშაოებში განვიხილავთ).

დავალება

1. VB-ის დაპროექტების სისტემაში პროგრამული პროდუქტი შექმენით, რომელიც **If Then Else** მმართველი კონსტრუქციის გამოყენებით გამოთვლის შემდეგი გამოსახულების მნიშვნელობას:

$$y = \begin{cases} ax^2 + b, & \text{თუ} \quad x < 0 \\ \sqrt{x^2 + 2}, & \text{თუ} \quad 0 \le x \le 20 \\ \frac{1}{x} + 15, & \text{თუ} \quad x > 20 \end{cases}$$

2. VB-ის დაპროექტების სისტემაში პროგრამული პროდუქტი შექმენით, რომელიც **If Then ElseIf** მმართველი კონსტრუქციის გამოყენებით გამოთვლის შემდეგი გამოსახულების მნიშვნელობას:

$$z = \begin{cases} \sin^2 x + 2, & \text{თუ} \quad x \le K, \\ \frac{\cos x}{\sqrt{x+5}}, & \text{თუ} \quad K < x < P, \\ \frac{arctgx}{e^{x+1}}, & \text{თუ} \quad x \ge P \end{cases}$$

3. **VB**-ის დაპროექტების სისტემაში პროგრამული პროდუქტი შექმენით, რომელიც **Iif** ფუნქციის გამოყენებით გამოთვლის შემდეგი გამოსახულების მნიშვნელობას:

$$D=\begin{cases} a^x+bx, & \text{თუ } x\le 0, \\ \dfrac{a^{x+2}}{\sqrt{bx}}, & \text{თუ } x>0 \end{cases}$$

პროგრამული კოდი და მიღებული შედეგები:

---------- ----------
(ქულა) (ხელმოწერა)

ლაბორატორიული სამუშაო №7
პირობითი და უპირობო გადასვლის ოპერატორები.
MsgBox ფუნქციის გამოყენება

მრავალჯერადი ამორჩევის სტრუქტურა **Select Case** საშუალებას გვაძლევს ოპერატორთა რამდენიმე ჯგუფიდან ერთ-ერთი ამოვირჩიოთ და შევასრულოთ, იმის მიხედვით, თუ რა მნიშვნელობა ექნება სპეციალურ მმართველ ცვლადს ან გამოსახულებას. აღნიშნული ოპერატორის ჩაწერის სინტაქსი შემდეგია:

Select Case მმართველი გამოსახულება
Case მნიშვნელობა-1
ოპერატორი-1
Case მნიშვნელობა-2
ოპერატორი-2
Case მნიშვნელობა-3
ოპერატორი-3
. . . . . .
Case Else
ოპერატორი *n*

End Select

<u>მაგალითი</u>

VB-ის დაპროექტების სისტემაში **Select Case** ოპერატორთა ბლოკის გამოყენებით შევადგინოთ შემდეგი სისტემის ამოხსნის პროგრამა:

$$A=\begin{cases} bx^2+d\cdot\sin^2 x, & \text{თუ } 0<d\cdot x<1, \\ \dfrac{b^x+d}{2}, & \text{თუ } d\cdot x\le 0 \\ \dfrac{d}{\sqrt{x^2+bx}}, & \text{თუ } d\cdot x\ge 1 \end{cases}$$

ამოხსნა

```
Private Sub Command1_Click( )
Dim A As Double, b As Single, d As Single, x As Single
x = InputBox("Input x:", "Data")
b = InputBox("Input b:", "Data")
d = InputBox("Input d:", "Data")
Select Case d * x
Case Is >= 1
A = d / Sqr(x ^ 2 + b * x)
Case Is <= 0
A = (b ^ x + d) / 2
Case Else
A = b * x ^ 2 + d * (Sin(x)) ^ 2
End Select
MsgBox A, 64, "Message"
End Sub
```

GoTo ოპერატორი უპირობო გადასვლის ოპერატორია, რომელიც საშუალებას იძლევა დავარღვიოთ ოპერატორთა შესრულების რიგითობა და ნებისმიერი ადგილიდან განვაგრძოთ პროგრამის შესრულება.

ზემოაღნიშნული ოპერატორის ჩაწერის სინტაქსი შემდეგია:

GoTo *n*

სადაც ***n*** არის ჭდე. ის თავსდება ცალკე სტრიქონში იმ ოპერატორის წინ, რომელზეც მიმართვა ხდება. ჭდის ჩაწერა აუცილებლად ორწერტილით (:) უნდა დასრულდეს.

MsgBox ფუნქცია გამოიყენება ინფორმაციის ეკრანზე გამოტანის მიზნით. ის თავსდება მინიჭების ოპერატორის მარჯვენა ნაწილში და ჩაწერის შემდეგი სინტაქსი გააჩნია:

ცვლადი = MsgBox(შეტყობინება, რიცხვითი კოდები, "სათაური")

აღნიშნული ფუნქციის არგუმენტების დანიშნულება იგივეა, რაც შესაბამისი პარამეტრების დანიშნულებები **MsgBox** ოპერატორში, იმ განსხვავებით, რომ ისინი ფრჩხილებშია მოთავსებული.

MsgBox ფუნქციის მნიშვნელობა არის იმ ღილაკის კოდი (რიცხვი), რომელზეც დავაჭერთ შეტყობინების ფანჯარაში. ეს მნიშვნელობა ოპერატორ-ფუნქციის მარცხენა ნაწილში მდგომ ცვლადს მიენიჭება.

1-ელ ცხრილში მოყვანილია **MsgBox** ფუნქციის მნიშვნელობები, რომელსაც ის გვაძლევს სხვადასხვა ღილაკებზე დაჭერისას:

ცხრილი 1

დაჭერილი ღილაკის დასახელება	ფუნქციის მნიშვნელობა
OK	**1 (vbOk)**
Cancel	**2 (vbCancel)**
Abort	**3 (vbAbort)**
Retry	**4 (vbRetry)**
Ignore	**5 (vbIgnore)**
Yes	**6 (vbYes)**
No	**7 (vbNo)**

დავალება

VB-ის დაპროგრამების ენაზე შეადგინეთ ამა თუ იმ სისტემაში რეგისტრაციის ამოცანა. პროგრამაში შეიტანეთ მომხმარებლის სახელი და პაროლი, რომელთა მნიშვნელობები შესაბამის ცვლადებს მიენიჭება. წარუმატებელი რეგისტრაციის შემთხვევაში შეკითხვის ტიპის ფანჯარა უნდა გაიხსნას, რომელიც საშუალებას მოგცემთ პროგრამის გაგრძელების ერთ-ერთი მიმართულება აირჩიოთ - ხელახალი რეგისტრაცია ან პროგრამის დასრულება. პროგრამულ კოდს ქვემოთ წარმოდგენილი სახე უნდა ჰქონდეს, ხოლო ფორმის გრაფიკული ინტერფეისის აგება და მასზე ობიექტების განთავსება დამოუკიდებლად მოახდინეთ.

```
Private Sub Command1_Click( )
Dim UserName As String
Dim password As String, k As Integer
M1:
UserName = InputBox("User Name", "Registration")
password = InputBox("Enter Password", "Password")
If UserName = "Lela" And password = "GTU" Then
MsgBox "Welcome, Lela!", 64, "Registration"
GoTo Finish
End If
MsgBox "Unknoun User", 64, "Registration"
k = MsgBox("Try Again", 32 + 4, "Registration")
If k = 6 Then GoTo M1
Finish:
End
End Sub
```

---------- ----------

(ქულა) (ხელმოწერა)

ლაბორატორიული სამუშაო №8
ციკლის ოპერატორები

ციკლური პროცესების ორგანიზების მიზნით **VB**-ის დაპროგრამების ენაში შემდეგი ოპერატორები გამოიყენება:

1. **Do While(Until) – Loop;**
2. **Do – Loop While(Until);**
3. **For-Next.**

ოპერატორი **Do While – Loop** და **Do Until – Loop** ახდენს ციკლის პირობის შემოწმებას მის დასაწყისში და ჩაწერის შემდეგი სტრუქტურა გააჩნია:

Do While პირობა
ოპერატორი
. . . . . **ციკლის ტანი**
ოპერატორი
Loop

Do Until პირობა
ოპერატორი
. . . . . **ციკლის ტანი**
ოპერატორი
Loop

ციკლებში მითითებული პირობა ლოგიკური გამოსახულებაა, რომლის შემოწმების შედეგი შეიძლება ჭეშმარიტი (True) ან მცდარი (False) იყოს. ზემოთ წარმოდგენილი ოპერატორების ორი ფორმა ერთმანეთისგან იმით განსხვავდება, რომ პირველ შემთხვე-

ვაში (**Do While – Loop**) პირობის ქვეშ ციკლის ტანში არსებული ოპერატორების გამეორების პირობა იგულისხმება, ხოლო მეორეში (**Do Until – Loop**) – ციკლიდან გამოსვლის; ამასთან პირველ შემთხვევაში ციკლის ტანში არსებული ოპერატორები მანამდე სრულდება, სანამ ციკლის პირობა ჭეშმარიტია, ხოლო მეორე შემთხვევაში – პირობა მცდარია.

<u>დავალება 1</u>

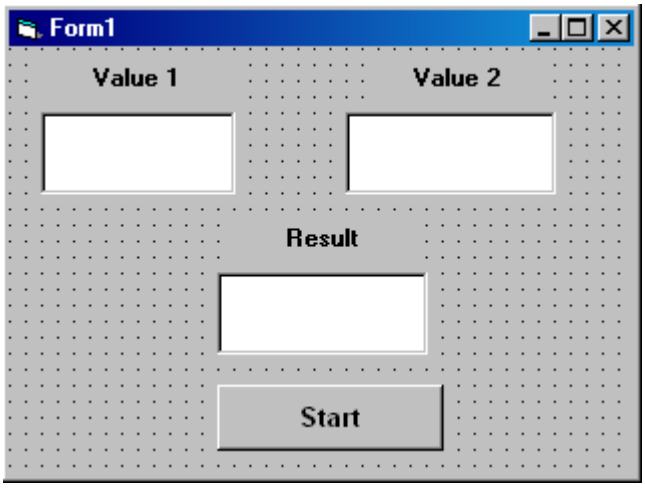
VB-ის დაპროგრამების ენაზე შეადგინეთ ორი ***a*** და ***b*** რიცხვის უდიდესი საერთო გამყოფის განსაზღვრის პროგრამა. ამოცანა გადავწყვიტეთ ორ ვარიანტად:

1) **Do While – Loop**;
2) **Do Until – Loop** ციკლების გამოყენებით.

ამოცანის ამოხსნის პირველ ვარიანტს შემდეგი სახე აქვს:

```
Private Sub Command1_Click( )
Dim a As Integer, b As Integer
a = Text1.Text
b = Text2.Text
Do While a < > b
If a > b Then
a = a - b
Else
b = b - a
End If
Loop
Text3.Text = a
End Sub
```

ამოცანის გადაწყვეტის დროს ფორმაზე განთავსებული უნდა იყოს 1-ელ ნახაზზე წარმოდგენილი ობიექტები:



ნახ. 1.

ოპერატორი **Do – Loop While** და **Do – Loop Until** ახდენს ციკლის პირობის შემოწმებას მის დასასრულს და გააჩნია ჩაწერის შემდეგი სტრუქტურა:

Do
ოპერატორი
. . . . . .
ოპერატორი
Loop While პირობა

} **ციკლის ტანი**

ციკლის დასასრულს, როდესაც პირობა მოწმდება, მიუხედავად მისი ჭეშმარიტების ან მცდარობისა, ციკლის ტანში არსებული ოპერატორები ერთხელ მაინც სრულდება. სწორედ ამით განსხვავდება ერთმანეთისგან ციკლი: **Do While(Until) – Loop** და **Do – Loop While(Until)**. საზოგადოდ, მკვეთრი გამიჯვნა ზემოაღნიშნული ოპერატორების გამოყენებისას არ არსებობს.

დავალება 2

VB-ის დაპროგრამების ენაზე შეადგინეთ $y = a \cdot x^2 + b$ ფუნქციის ყველა მნიშვნელობის გამოთვლის პროგრამა. ამოცანის საწყისი პირობებია:
$a = 2,\ b = 4,\ x \in [1;10],\ h = 0,5,$ სადაც h არის x არგუმენტის ცვლილების ბიჯი მოცემულ დიაპაზონში. ამოცანა გადაწყვიტეთ ორ ვარიანტად:

1) **Do – Loop While**;
2) **Do – Loop Until** ციკლების გამოყენებით.

ამოცანის ამოხსნის მეორე ვარიანტს შემდეგი სახე აქვს:

```
Private Sub Command1_Click( )
Dim a As Integer, b As Integer
Dim x As Single, h As Single
```

```
Dim y As Single
a = 2: b = 4: h = 0.5: x = 1
Do
y = a * x ^ 2 + b
Print "x="; x, "y="; y
x = x + h
Loop Until x > 10
End Sub
```

I დავალების პროგრამული კოდი და მიღებული შედეგები:	II დავალების პროგრამული კოდი და მიღებული შედეგები:

---------- (ქულა)

---------- (ხელმოწერა)

ლაბორატორიული სამუშაო №9
ციკლის ოპერატორი For-Next

ოპერატორი **For-Next** იმ შემთხვევაში გამოიყენება, როდესაც წინასწარაა განსაზღვრული ციკლის გამეორებათა რაოდენობა.

აღნიშნული ოპერატორის ზოგადი სტრუქტურა შემდეგია:

For i=n_1 T n_2 Step n_3
ოპერატორი
. . . . . .
ოპერატორი
Next i

} **ციკლის ტანი**

i - განმეორებათა მთვლელია, რომელსაც ციკლის მთვლელს ან მმართველ პარამეტრს უწოდებენ;

n_1 – ციკლის მთვლელის საწყისი მნიშვნელობაა;

n_2 – ციკლის მთვლელის საბოლოო მნიშვნელობაა;

n_3 – ციკლის მთვლელის ცვლილების ბიჯია (n_1; n_2) შუალედში.

n_1, n_2, n_3 სიდიდეები შეიძლება რიცხვებს, არითმეტიკულ გამოსახულებებს ან ცვლადებს წარმოადგენდეს. ჩვეულებრივ, თუ $n_2 > n_1$, მაშინ ციკლის მთვლელის ცვლილების n_3 ბიჯი დადებითია, ხოლო თუ $n_2 < n_1$, მაშინ ციკლის მთვლელის ცვლილების n_3 ბიჯი უარყოფითია. თუ ციკლის პარამეტრის (მთვლელის) ცვლილების ბიჯი 1-ის ტოლია, n_3 შეიძლება საერთოდ არ მივუთითოთ.

<u>დავალება 1</u>

VB-ის დაპროგრამების ენაზე შეადგინეთ $y = 2 \cdot x^2 + 1$ ფუნქციის ყველა მნიშვნელობის გამოთვლის პროგრამა. ამოცანის საწ-

ყისი პირობებია: $x \in [a;b]$, $h = 1,5$, სადაც h არის x არგუმენტის ცვლილების ბიჯი მოცემულ დიაპაზონში.

ამოხსნა:

```
Private Sub Command1_Click( )
Dim x As Single, y As Single, h As Single
Dim a As Integer, b As Integer
a = InputBox("Enter a:", "Data")
b = InputBox("Enter b:", "Data")
h = InputBox("Enter h:", "Data")
For x = a To b Step h
y = 2 * x ^ 2 + 1
Print "x="; x, "y="; y
Next x
End Sub
```

წარმოადგინეთ პროგრამის შესრულებაზე გაშვების შედეგად შემთხვევისათვის, როდესაც $a = 1$ და $b = 15$, მიღებული რეზულტატები.

I დავალების შედეგები:

For-Next ციკლიდან შესაძლებელია ალტერნატიული გამოსვლა. ამ პროცესს **Exit For** ოპერატორი ემსახურება**.** ის იმ შემთხვევაში გამოიყენება, როდესაც რაიმე პირობის შესრულების

შემდეგ ციკლის გამეორებას აზრი აღარ აქვს. როგორც წესი, ეს ოპერატორი თავსდება პირობითი **If** ოპერატორის რომელიმე შტოში.

If პირობა Then Exit For

დავალება 2

VB-ის დაპროგრამების ენაზე შეადგინეთ $y=\frac{\sqrt{x^2+1}}{2}$ გამოსახულების მნიშვნელობების გამოთვლის პროგრამა, სადაც არგუმენტი $x\in[2;20]$ აღნიშნულ დიაპაზონში იცვლება $h=2$-ის ტოლი ბიჯით. თუ y ფუნქციის მნიშვნელობა 7-ზე მეტი აღმოჩნდება, გამოთვლითი პროცესი შეწყვიტეთ.

II დავალების პროგრამული კოდი და მიღებული შედეგები:

---------- ----------
(ქულა) (ხელმოწერა)

ლაბორატორიული სამუშაო №10
HScrollBar და VScrollBar ობიექტების გამოყენება

HScrollBar და **VScrollBar** ობიექტები არის შესაბამისად ჰორიზონტალური და ვერტიკალური გადაფურცვლის ზოლები, რომელთა დანიშნულებაა გარკვეული რიცხვითი დიაპაზონიდან მნიშვნელობის ამორჩევა.

აღნიშნულ ობიექტებთან დაკავშირებულ ძირითად მოვლენას წარმოადგენს **Change** (ცვლილება) მოვლენა, რომელიც ხორციელდება გადაფურცვლის ზოლში მცოცის მდებარეობის ცვლილების დროს. ამ ობიექტების გამოსაყენებლად აუცილებელია **Change** მოვლენის შესაბამისი პროგრამული პროცედურის შედგენა.

HScrollBar და **VScrollBar** ობიექტების ძირითადი თვისებებია:

1. **Min** – თვისება, რომელიც განსაზღვრავს გადაფურცვლის ზოლის (იგივე რიცხვითი დიაპაზონის) ქვედა საზღვარს. მისი საწყისი მნიშვნელობა 0-ის ტოლია;
2. **Max** – თვისება, რომელიც განსაზღვრავს გადაფურცვლის ზოლის (იგივე რიცხვითი დიაპაზონის) ზედა საზღვარს. მისი საწყისი მნიშვნელობა 32767-ის ტოლია;
3. **Value** – თვისების მნიშვნელობა არის **Min÷Max** დიაპაზონში მცოცის მიმდინარე მდებარეობის შესაბამისი რიცხვი. პროგრამაში ეს მნიშვნელობა ენიჭება რაიმე ცვლადს. მაგალითად, **A=HScroll1.Value**. ამ თვისების საწყის მნიშვნელობას შეესაბამება **Min** თვისების მიმდინარე მნიშვნელობა;

4. **SmallChange** – თვისება განსაზღვრავს **Value** თვისების მნიშვნელობის ცვლილების მცირე ბიჯს, რომლითაც ის იცვლება გადაფურცვლის ისრებზე დაჭერისას;
5. **LargeChange** – თვისება განსაზღვრავს **Value** თვისების მნიშვნელობის ცვლილების დიდ ბიჯს, რომლითაც ის იცვლება მცოცსა და ისრებს შორის თაგვით დაჭერისას.

აღნიშნული თვისებების მნიშვნელობათა დაფიქსირება ხდება ინტერფეისის შექმნის ეტაპზე ობიექტთა თვისებების (**Properties**) ფანჯარაში, თუმცა შესაძლებელია ეს პროგრამულადაც განვახორციელოთ.

<u>დავალება</u>

შეადგინეთ პროგრამული პროდუქტი, რომელიც საგნების რაოდენობისა და ქულათა ჯამის მიხედვით განსაზღვრავს სტუდენტის საშუალო შედეგს. ამასთან, საგნების რაოდენობა და ქულათა ჯამი ამოირჩიეთ **VScroll1** და **HScroll1** ვერტიკალური და ჰორიზონტალური გადაფურცვლის ზოლების მეშვეობით. ამოცანის საწყისი მონაცემები და მიღებული შედეგი ასახეთ ტექსტურ ველებში. თვალსაჩინოების მიზნით ფორმაზე მოათავსეთ **Label** ტიპის სამი ობიექტი წარწერებით „ქულათა ჯამი“, „საგნების რაოდენობა“ და „საშუალო შედეგი“.

დასმული ამოცანის პროგრამული კოდი წარმოდგენილია 1-ელ ნახაზზე, ხოლო **VScroll1** და **HScroll1** ზოლების პარამეტრები ნაჩვენებია 1-ელ ცხრილში.

ცხრილი 1

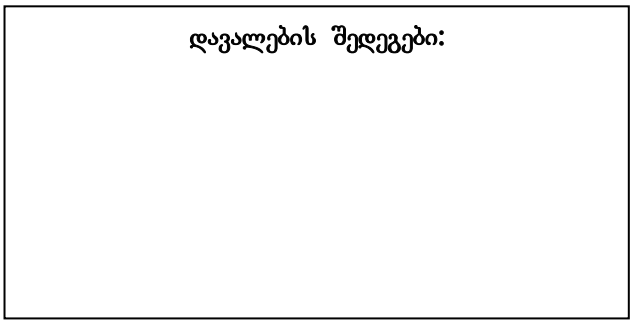
ობიექტი	Min	Max	SmallChange	LargeChange	Value
VScroll 1	50	300	1	5	50
HScroll 1	20	50	1	3	20

```
Dim sag As Integer, qula As Integer, average As Single

Private Sub HScroll1_Change()
qula = VScroll1.Value
Text1.Text = qula
sag = HScroll1.Value
Text2.Text = sag
average = qula / sag
Text3.Text = average
End Sub

Private Sub VScroll1_Change()
sag = HScroll1.Value
Text2.Text = sag
qula = VScroll1.Value
Text1.Text = qula
average = qula / sag
Text3.Text = average
End Sub
```

ნახ. 1.



---------- (ქულა)

---------- (ხელმოწერა)

ლაბორატორიული სამუშაო №11
ობიექტები Frame და OptionButton

Frame ობიექტს დამოუკიდებელი ფუნქცია არ გააჩნია. ის არის ჩარჩო, რომელიც საშუალებას გვაძლევს რამდენიმე ობიექტი ერთ ჯგუფად გავაერთიანოთ. ჩარჩოში, როგორც წესი, საერთო დანიშნულების მქონე ობიექტებს ათავსებენ, მაგალითად, ალმებს ან გადამრთველებს.

Frame ობიექტის ეგზემპლარებს ფორმაზე გადატანის შემდეგ ენიჭებათ სახელები: **Frame1, Frame2,…**

ამ ობიექტებისთვის მოვლენათა ანალიზი არ ხდება, რადგან მომხმარებელს მუშაობა უწევს მასში მოთავსებულ ობიექტებთან.

OptionButton ობიექტი (გადამრთველი ღილაკი) წარმოადგენს წრის ფორმის ღილაკს და შესაძლო ოპერაციათა ერთობლიობიდან მხოლოდ ერთის ამორჩევას ემსახურება. მას ორი მდგომარეობა გააჩნია ჩართული და გამორთული. გადამრთველი ღილაკის ჩართვა-გამორთვა მასზე თაგვის დაჭერით ხორციელდება. მოცემულ მომენტში შესაძლებელია მხოლოდ ერთი ღილაკის ჩართვა, რომელიც იწვევს მანამდე ჩართული ღილაკის გამორთვას.

OptionButton ობიექტის ეგზემპლარები ფორმაზე თავსდება სახელებით: **Option1, Option2, …**

Value ობიექტის მთავარი თვისებაა, რომლის შესაძლო მნიშვნელობებია: 1 (True) – ღილაკი ჩართულია ან 0 (False) – ღილაკი გამორთულია.

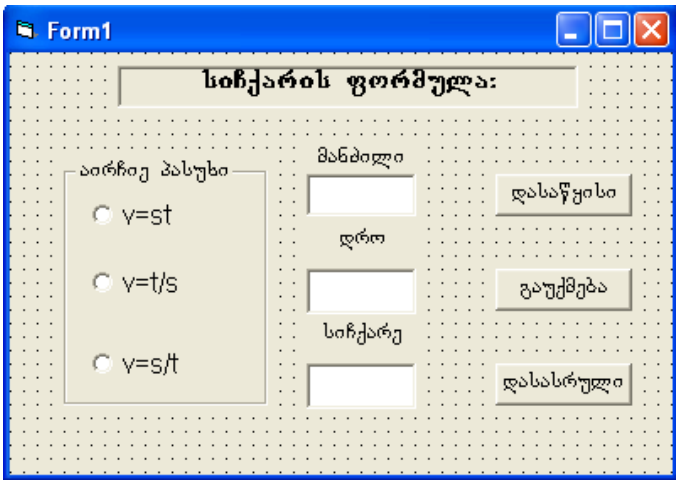
Caption თვისება განსაზღვრავს იმ ოფციის დასახელებას, რომელსაც ღილაკი ემსახურება. ის თავსდება ობიექტის მარჯვნივ.

Click ობიექტის მთავარი მოვლენაა, რომელიც მდგომარეობს თაგვის დაჭერით ალმის ჩართვა-გამორთვაში.

დავალება 1

შეადგინეთ პროგრამული პროდუქტი, რომელიც **Frame** და **OptionButton** ობიექტების გამოყენებით სტუდენტის ცოდნას შეამოწმებს. კერძოდ, ფიზიკის კურსიდან ცნობილი ფორმულის თანახმად, გამოთვლის მგზავრის სიჩქარეს. არასწორი პასუხის შემთხვევაში **MsgBox** ფანჯარა სათანადო შეტყობინების გამოტანას მოახდენს.

საწყისი ფორმა წარმოდგენილია 1-ელ ნახაზზე:



ნახ. 1.

პროგრამულ კოდს შემდეგი სახე აქვს:

```
Dim v As Single, s As Single, t As Single
Private Sub Command1_Click( )
s = InputBox("Input S:", "Data")
t = InputBox("Input t:", "Data")
Text1.Text = s
Text2.Text = t
If Option3.Value = True Then
```

```
v = s / t
Text3.Text = v
Else
MsgBox "Answer is FALSE - Click OK"
End If: End Sub
Private Sub Command2_Click( )
Text1.Text = Clear
Text2.Text = Clear
Text3.Text = Clear
End Sub
Private Sub Command3_Click( )
End: End Sub
```

მიღებული შედეგები წარმოდგენილია მე-2 ნახაზზე:

ნახ. 2.

დავალება 2

შეადგინეთ პროგრამა, რომელიც **Frame** და **OptionButton** ობიექტების გამოყენებით სწორედ ამორჩეული ჰერონის ფორმულის საფუძველზე გამოთვლის სამკუთხედის ფართობს, ხოლო არასწორი პასუხის შემთხვევაში **MsgBox** ფანჯარა სათანადო შეტყობინებას გამოიტანს.

II დავალების პროგრამული კოდი და მიღებული შედეგები:

---------- ----------
(ქულა) (ხელმოწერა)

ლაბორატორიული სამუშაო №12
ობიექტი ListBox

ListBox (სიის ფანჯარა) ობიექტი საშუალებას გვაძლევს შევადგინოთ გარკვეული ელემენტების სია (ჩამონათვალი) და შემდგომ ამოვირჩიოთ სიიდან სასურველი ელემენტები. აღნიშნული ობიექტი მონაცემთა შეტანა-გამოტანის საშუალებებს მიეკუთვნება. მისი ცალკეული ეგზემპლარი ფორმაზე მოთავსებისას აღინიშნება სახელებით: **List1, List2, List3,…**

სიის ყოველ ელემენტს ინდექსი აქვს, რომელიც სიაში მის მდებარეობას განსაზღვრავს. ინდექსაცია 0-დან იწყება. სიაში ნებისმიერ მომენტში შესაძლებელია ახალი ელემენტის დამატება ან უსარგებლო ელემენტის ამოშლა.

Click ობიექტთან დაკავშირებული მთავარი **მოვლენაა**, რომელიც სიის ელემენტზე თაგვის დაჭერით ხორციელდება და გულისხმობს სიიდან ელემენტის ამორჩევას.

პროგრამიდან სიის ელემენტებთან მუშაობა ამ ობიექტთან დაკავშირებული **მეთოდების** საშუალებით ხორციელდება.

AddItem მეთოდი სიის შევსებას და არსებულ სიაში ახალი ელემენტის დამატებას ემსახურება. მისი ჩაწერის სინტაქსი შემდეგია:

ობიექტი.AddItem ელემენტი, ინდექსი

ობიექტი - **ListBox** ტიპის ობიექტის სახელია, ელემენტი – სიაში შესატანი მონაცემი ან მისი შესაბამისი ცვლადის დასახელება, ხოლო ინდექსი განსაზღვრავს სიაში ელემენტის ჩამატების პოზიციას. იგი მთელი რიცხვია. თუ ინდექსი მითითებული არ არის, ელემენტი სიის ბოლოს ჩაიწერება.

თუ სიის ელემენტები სრულად ვერ თავსდება მოცემული ზომის **ListBox** ფანჯარაში, ობიექტს ავტომატურად გაუჩნდება

გადაფურცვლის ზოლები, რომელთა გამოყენება შესაძლებელია მხოლოდ პროგრამის შესრულების პროცესში.

RemoveItem მეთოდი სიიდან ელემენტების წასაშლელად გამოიყენება. მას შემდეგი სახე აქვს:

ობიექტი.RemoveItem ინდექსი

ინდექსი - წასაშლელი ელემენტის ინდექსია. მაგალითად ბრძანება: **List1.RemoveItem 5** სიიდან ამოშლის მეექვსე ელემენტს (რადგან ინდექსაცია 0-დან იწყება).

Clear მეთოდი სიიდან ყველა ელემენტის წაშლას მოახდენს. მისი ჩაწერის სინტაქსი შემდეგია:

ობიექტი.Clear

ListBox ობიექტი შემდეგი ძირითადი **თვისებებით** ხასიათდება:

1. **Text -** ობიექტის ყველაზე მნიშვნელოვანი თვისებაა, რომელიც სიიდან ამორჩეული ელემენტის მნიშვნელობას მიიღებს. მისი გამოყენება შესაძლებელია მხოლოდ ამ ობიექტისთვის შედგენილ (**Click** მოვლენით გამოწვეულ) პროგრამულ პროცედურაში. მაგალითად, **name=List1.Text** ოპერატორით **name** ცვლადს სიაში ამორჩეული ელემენტის მნიშვნელობა ენიჭება.
2. **List** (ინდექსი) – თვისება საშუალებას გვაძლევს მივიღოთ სიის ელემენტის მნიშვნელობა მისი ინდექსის მიხედვით. მაგალითად, **S=List1.List(5)** ოპერატორი **S** ცვლადს მიანიჭებს სიის მეექვსე ელემენტის მნიშვნელობას. **Text** თვისებისაგან განსხვავებით, რომლის მნიშვნელობა ელემენტზე თაგვის დაჭერით განისაზღვრება, **List** თვისება ციკლში ელემენტთა ავტომატურად ამორჩევისთვის გამოიყენება. ინ-

დექსის მნიშვნელობა არის ცვლადი, რომელიც ამავე დროს ციკლის მმართველი პარამეტრის როლში გვევლინება.

3. **ListIndex –** თვისების მნიშვნელობაა სიიდან ამორჩეული ელემენტის ინდექსი. თუ სიიდან არცერთი ელემენტი ამორჩეული არ არის, მისი მნიშვნელობა ერთის ტოლია. მაგალითად, ***n*=List1.ListIndex** ოპერატორი ***n*** ცვლადს მიანიჭებს სიაში ამორჩეული ელემენტის ინდექსის მნიშვნელობას.
4. **ListCount –** თვისების მნიშვნელობა განსაზღვრავს სიაში ელემენტთა საერთო რაოდენობას. მაგალითად, **A=List1.ListCount.**

ზემოთ განხილული თვისებები მხოლოდ პროგრამულ კოდში გამოიყენება. ახლა განვიხილოთ იმ თვისებათა ჩამონათვალი, რომელთა შერჩევა ობიექტის თვისებების (**Properties**) ფანჯარაში მოხდება:

1. **List –** თვისების მნიშვნელობებია სიის ელემენტები. ის საშუალებას გვაძლევს სიის შევსება განვახორციელოთ **Properties** ფანჯარაში. ამისათვის საჭიროა გავხსნათ ამ თვისების მნიშვნელობათა ველის ჩამონათვალი და შევიტანოთ ელემენტების მნიშვნელობები (ყოველი ელემენტის შეტანის შემდეგ კლავიატურაზე უნდა დავაწვეთ **Enter** კლავიშს).
2. **Sorted –** თვისება განსაზღვრავს სიაში ელემენტების დალაგების წესს. მისი საწყისი მნიშვნელობაა **False**, რომელიც სიის ელემენტებს დაალაგებს შეტანის რიგითობის მიხედვით. **True** მნიშვნელობა ელემენტებს დააწყობს ანბანის მიხედვით.
3. **Columns –** თვისება სიის ელემენტების სვეტებად დალაგებას მოახდენს. თუ აღნიშნული თვისების შესაბამის

ველში 0-ის ნაცვლად სხვა რიცხვს ჩავწერთ, ეს უკანასკნელი **ListBox** ფანჯარაში სვეტების რაოდენობას განსაზღვრავს, რომლებშიც სიის ელემენტები განთავსდება.

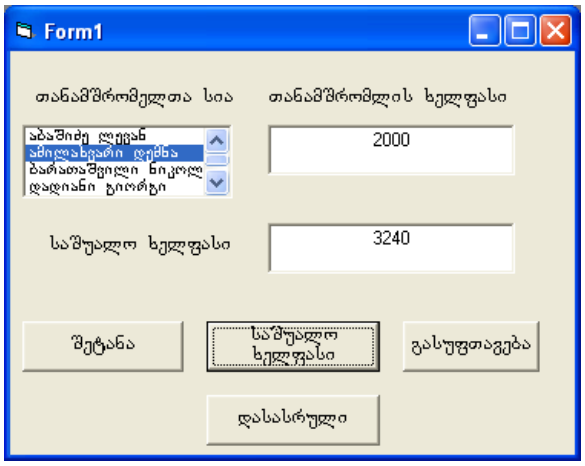
დავალება

შეადგინეთ პროგრამული პროდუქტი, რომელიც **ListBox** ობიექტის გამოყენებით საშუალებას მოგცემთ შეიტანოთ ათი თანამშრომლისგან შემდგარი სია, თითოეულ თანამშრომელზე გასაცემი ხელფასის ოდენობა და საშუალო ხელფასის მნიშვნელობა განსაზღვროთ.

დასმული ამოცანის გადაწყვეტის პროგრამულ კოდს ქვემოთ წარმოდგენილი საზე აქვს, ხოლო საბოლოო შედეგები ნაჩვენებია 1-ელ ნახაზზე.

```
Dim g(10) As String, x(10) As Integer, i As Integer
Dim S As Single, ind As Integer
Private Sub Command1_Click( )
For i = 1 To 10
g(i) = InputBox("Gvari", , i)
x(i) = InputBox("Xelpasi", , i)
List1.AddItem g(i)
Next i
End Sub
Private Sub Command2_Click( )
S = 0
For i = 1 To 10
S = S + x(i)
Next i
S = S / 10
Text2.Text = S
End Sub
```

```
Private Sub Command3_Click( )
List1.Clear
Text1.Text = Clear
Text2.Text = Clear
End Sub
Private Sub Command4_Click( )
End
End Sub
Private Sub List1_Click()
ind = List1.ListIndex
Text1.Text = x(ind + 1)
End Sub
```



ნახ. 1.

---------- ----------
(ქულა) (ხელმოწერა)

ლაბორატორიული სამუშაო №13
ობიექტი ComboBox

ComboBox (კომბინირებული ფანჯარა) ობიექტი **TextBox** და **ListBox** ობიექტების ერთობლიობაა. სიის ფანჯარასთან ერთად იგი ტექსტურ ველსაც შეიცავს, რომელშიც შესაძლებელია სიაში ჩასართავი ელემენტების შეტანა და რედაქტირება. აღნიშნულ ობიექტს იმ შემთხვევაში იყენებენ, როდესაც სიაში შესატანი ელემენტების მნიშვნელობები წინასწარ ცნობილი არ არის.

Click ობიექტის მთავარი მოვლენაა. მას ვიყენებთ სიაში არსებული სასურველი ელემენტის ამორჩევის დროს. ამ მოვლენისთვის პროგრამული პროცედურა იწერება, რომელშიც სიიდან ამორჩეული ელემენტის დამუშავება მოხდება.

აღნიშნული ობიექტის ეგზემპლარები ფორმაზე მოთავსდება სახელებით: **Combo1, Combo2, Combo3,…**

ComboBox ობიექტი ყველა იმ თვისებითა და მეთოდით ხასიათდება, რომელიც **ListBox** ობიექტს აქვს. ამას გარდა, მას კიდევ ორი მნიშვნელოვანი თვისება გააჩნია.

Style მხოლოდ ამ ობიექტისთვის დამახასიათებელი თვისებაა. ის განსაზღვრავს მის გარეგნულ სახეს და ფუნქციონირების წესს. თვისების შესაძლო მნიშვნელობებია: 0 – ობიექტი არის ტექსტური ველი ჩამონათვლის ისრით, რომელზე დაჭერითაც იხსნება ელემენტთა სია; 1 – ობიექტში მუდმივად ჩანს როგორც ტექსტური ველი, ასევე ელემენტთა სია; 2 – ობიექტში წარმოდგენილია მხოლოდ სია. ის ტექსტურ ველს არ შეიცავს. თვისების საწყისი მნიშვნელობაა 0.

Tetx თვისება **TextBox** ობიექტის ამავე თვისების ანალოგიურია. მისი მნიშვნელობა არის **ComboBox** ობიექტის ტექსტური ველის შემცველობა. ეს თვისება გამოიყენება სხვა ობიექტების

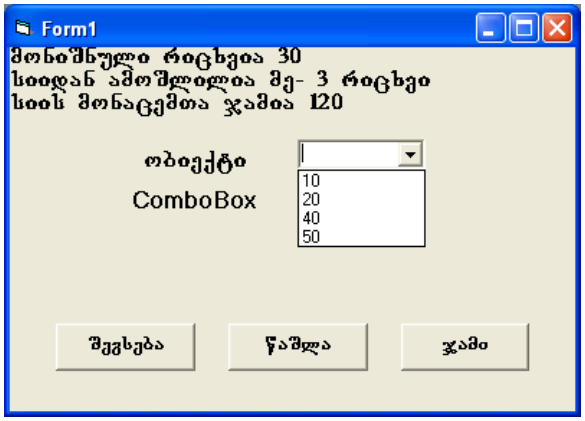
პროცედურებში, მაშინ, როდესაც **ListBox** ობიექტის ამავე დასახელების თვისება გამოიყენება **ComboBox** ობიექტის პროცედურაში და სიიდან მიიღებს ამორჩეული ელემენტის მნიშვნელობას.

დავალება

შეადგინეთ პროგრამული პროდუქტი, რომელიც **ComboBox** ობიექტის გამოყენებით სიის შევსებას მოახდენს (ამ შემთხვევაში რიცხვითი მნიშვნელობებით), მისი შემადგენელი ელემენტების ჯამის გამოთვლას და სიიდან ამორჩეული ელემენტების წაშლას. ამოცანის გადაწყვეტის პროგრამული კოდი ქვემოთაა წარმოდგენილი, ხოლო მიღებულ შედეგებს 1-ელ ნახაზზე ნაჩვენები სახე უნდა ჰქონდეს.

```
Dim X As Single, S As Single
Dim ind As Integer, k As Integer
Private Sub Command1_Click( )
X = Combo1.Text
Combo1.AddItem X
End Sub
Private Sub Combo1_Click( )
Form1.Font = "AcadNusx"
Form1.FontBold = True
Form1.FontSize = 12
X = Combo1.Text
Print "moniSnuli ricxvia"; X
End Sub
Private Sub Command2_Click( )
ind = Combo1.ListIndex
Combo1.RemoveItem ind
Print "siidan amoSlilia me-"; ind + 1; "ricxvi"
```

```
End Sub
Private Sub Command3_Click( )
k = Combo1.ListCount: S = 0
For i = 0 To k - 1
S = S + X
Next i
Print "siis monacemTa jamia"; S
End Sub
```



ნახ. 1.

---------- ----------

(ქულა) (ხელმოწერა)

ლიტერატურა:

1. Гарнаев А. “Visual Basic 6.0. Разработка приложений”. Москва, 2006 г. 448 с.
2. Петрусос Е. “Visual Basic 6.0. Руководство разработчика в двух томах”. “Ирина”, BHV, Киев, 2000 г.
3. თ. მაჭარაძე, ზ. წვერაიძე “ინფორმატიკის საფუძვლები”. სტუ, თბილისი, 2003. 320 გვ.
4. ზ. მაჩაიძე, ნ. პავლიაშვილი “დაპროგრამების საფუძვლები”. თბილისი, 2002. 330 გვ.

სარჩევი

გვ.

რედაქტორი ნ. დოლიძე

გადაეცა წარმოებას 22.01.2010. ხელმოწერილია დასაბეჭდად 18.03.2010. ქაღალდის ზომა 60X84 1/16. პირობითი ნაბეჭდი თაბახი 4,5. ტირაჟი 100 ეგზ.

საგამომცემლო სახლი ,,ტექნიკური უნივერსიტეტი'', თბილისი, კოსტავას 77