ლ. გაჩეჩილაძე, ლ. ნონიკაშვილი

დაპროგრამების საფუძვლები
(ალგორითმული ენა C++)

I ნაწილი

მეთოდური მითითებები
ლაბორატორიული სამუშაოების შესასრულებლად

თბილისი
2013

განხილულია 15 ლაბორატორიული სამუშაოს შესრულების მეთოდიკა. ყოველი სამუშაოსთვის აღწერილია პროგრამული კოდის ჩაწერისა და C++ პროექტის შესრულების პროცედურები.

მოცემული მეთოდური მითითებანი განკუთვნილია დაპროგრამების საფუძვლების ასათვისებლად.

რეცენზენტი სრ. პროფ. ო. კოტრიკაძე

ISBN 978-9941-0-5569-0 (ორივე ნაწილის)
ISBN 978-9941-0-5570-6 (პირველი ნაწილის)

შესავალი

დაპროგრამების ენა **C++** , როგორც **C** ენის გაფართოება, შემუშავებული იქნა ბერნ სტრაუსტროპის მიერ გასული საუკუნის 80-იან წლებში.

C++-ში დაპროგრამების საფუძველს შეადგენს კლასები, რომელთა ბაზაზე იქმნება ახალი ობიექტები. **C++** ერთგვარი "ჰიბრიდული" ენაა, რომელიც დაპროგრამების საშუალებას იძლევა როგორც **C**-ს, ასევე ობიექტურ-ორიენტირებად სტილში.

ელექტრონულ გამომთვლელ მანქანაზე დამოკიდებულების მიხედვით დაპროგრამების ენები შეიძლება დაიყოს სამ ძირითად ტიპად:

1. მანქანური ენები;
2. ასემბლერის ენები;
3. მაღალი დონის დაპროგრამების ენები.

მანქანური ენები კონკრეტული ტიპის კომპიუტერზე დამოკიდებულ ალგორითმულ ენებს მიეკუთვნება. მანქანურ ენაზე შექმნილი პროგრამა წარმოადგენს ციფრული სახით ჩაწერილ მანქანურ ბრძანებათა ერთობლიობას.

ელექტრონულ გამომთვლელ მანქანაზე ორიენტირებადი ენები (ასემბლერი, ავტოკოდი) ახლოა მანქანურ ენებთან, მაგრამ მათგან იმით განსხვავდება, რომ აქ პროგრამები წარმოდგენილია არა ციფრულ კოდში, არამედ სიმბოლური სახით.

დაპროგრამების პროცესის დაჩქარების მიზნით პროგრამისტების მიერ შემუშავებულ იქნა მაღალი დონის დაპროგრამების ენები, რომლებიც არ არიან დამოკიდებული ამა თუ იმ კომპიუტერის კონკრეტულ ტიპზე.

მაღალი დონის დაპროგრამების ენებს მიეკუთვნება: **C**, **C++**, **FORTRAN**, **PASCAL**, **COBOL** და ა.შ.

ლაბორატორიული სამუშაო №1
მონაცემთა ტიპები, ოპერაციები და ფუნქციები დაპროგრამების ენა C++-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.

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.

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

თავისი ფიზიკური არსით ცვლადი მეხსიერების არეა, რომელსაც გარკვეული სახელი (იდენტიფიკატორი) გააჩნია. ცვლადის სახელი უნდა იწყებოდეს ლათინური ანბანის ასოთი; მასში დასაშვებია ასოების, ციფრების და ხაზგასმის ("_") სიმბოლოთა გამოყენება. ცვლადის სახელში დაუშვებელია (**Space**) ინტერვალის გამოყენება. ცვლადების სახელებად არ შეიძლება დაპროგრამების ენის ოპერატორების და საკვანძო (კოდური) სიტყვების გამოყენება.

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

C++-ში არსებობს ცვლადების შემდეგი ძირითადი ტიპები:

1. ***მთელი ტიპის*** ცვლადები. ისინი წარმოადგენენ მთელ რიცხვებს წილადი ნაწილის გარშე და არსებობენ სახით: ***int*** და ***long***.

- ***int*** ტიპის ცვლადში ინახება მთელი რიცხვები, რომელთა დასაშვები მნიშვნელობები მოთავსებულია დიაპაზონში: -32768-დან +32767-მდე.
- ***long*** (გრძელი) ტიპის ცვლადში ინახება მთელი რიცხვები, რომელთა დასაშვები მნიშვნელობები მოთავსებულია დიაპაზონში: -2147483648-დან +2147483647-მდე.

2. ცვლადები ***მცოცავი წერტილით***. ისინი არსებობენ სახით: ***float*** და ***double***.

- ***float*** ტიპის ცვლადში ინახება დადებითი და უარყოფითი წილადი რიცხვები, რომელთა დასაშვები აბსოლუტური მნიშვნელობები მოთავსებულია დიაპაზონში: $3,4X10^{-38}$-დან $3,4X10^{38}$-მდე.
- ***double*** (ორმაგი) ტიპის ცვლადში ინახება დადებითი და უარყოფითი წილადი რიცხვები, რომელთა დასაშვები აბსოლუტური მნიშვნელობები მოთავსებულია დიაპაზონში: $1,7X10^{-308}$-დან $1,7X10^{308}$-მდე.

3. ***სიმბოლოები***, იგივე ***char*** ტიპის სიმბოლური ცვლადები, სადაც ინახება ამა თუ იმ ანბანურ-ციფრული სიმბოლოს შესაბამისი მთელი რიცხვი. ***char*** ტიპის ცვლადები შეიძლება გამოყენებულ იქნას მთელი ტიპის ცვლადების სახით.

გარდა ცვლადების ზემოთ აღნიშნული ტიპებისა, არსებობს რიცხვითი ცვლადების კიდევ ერთი დამატებითი ტიპი: ***unsigned***-უნიშნო (მხოლოდ დადებითი) რიცხვები, რომლის მეშვეობით ორმაგდება რიცხვითი ცვლადების დასაშვები დადებით რიცხვთა დიაპაზონი. მაგ: ***unsigned int*** ტიპის ცვლადების დასაშვები რიცხვითი დიაპაზონია: 0-დან 65535-მდე.

კომპიუტერის მეხსიერების ეკონომიის მიზნით გამოიყენება ცვლადების მეორე დამატებითი ტიპი ***short*** (მოკლე), რომელიც ახორციელებს კომბინაციებს ცვლადების ზემოთ განხილულ მონაცემთა ტიპებთან. შესაძლებელია ***unsigned*** და ***short*** ტიპების ერთობლივი გამოყენებაც.

ნებისმიერი ცვლადი პროგრამაში გამოყენებამდე უნდა გაცხადდეს (აღიწეროს), რადგან ამ პროცესში ადგილი აქვს კომპიუტერის მეხსიერებაში წინასწარ უჯრათა გარკვეული უბნის გამოყოფას მოცემული ტიპის ცვლადების მნიშვნელობათა შენახვის მიზნით. ცვლადების გაცხადება ხდება შემდეგი წესის გათვალისწინებით:

პირველ რიგში ეთითება ცვლადის ტიპი, შემდეგ მისი იდენტიფიკატორი. როდესაც გაცხადების პროცესში ცვლადს ენიჭება კონკრეტული მნიშვნელობა, აღნიშნულ პროცესს ცვლადის ***ინიციალება*** ეწოდება. თუმცა შესაძლებელია ცვლადს კონკრეტული მნიშვნელობა მოგვიანებითაც მიენიჭოს.

თუ პროგრამაში რამდენიმე ერთიდაიმავე ტიპის ცვლადია, ისინი შეიძლება აღიწეროს ერთ სტრიქონზე ერთიმეორისგან მძიმის გამოყოფით. მაგ: **int *a*=5, *b*, *c*=10;**

აღნიშნულ შემთხვევაში წარმოებს ***a*** და ***c*** ცვლადების ინიციალება და ***b*** ცვლადის აღწერა.

კონსტანტა (მუდმივა) ეწოდება სიდიდეს, რომლის მნიშვნელობის შეცვლა პროგრამის შესრულების პროცესში დაუშვებელია.

კონსტანტები პროგრამაში გამოყენებამდე უნდა გაცხადდეს, რისთვისაც გამოიყენება საკვანძო (კოდური) სიტყვა ***const***, რომელსაც მოსდევს კონსტანტას ტიპი, მისი იდენტიფიკატორი და კონკრეტული მნიშვნელობა. რადგან კონსტანტა მუდმივი სიდიდეა, ამიტომ გაცხადების პროცესშივე უნდა მოხდეს მისი ინიციალება.

კონსტანტების აღწერის ნიმუშები:

ა) ***const float pi=3.1416***; ბ) ***const int A=5***.

პირველ შემთხვევაში ხდება ***float*** ტიპის ***pi*** იდენტიფიკატორის მქონე კონსტანტას ინიციალება, რის შედეგადაც მას ენიჭება 3.1416-ის ტოლი მნიშვნელობა; ხოლო მეორე შემთხვევაში ***int*** (მთელი) ტიპის ***A*** იდენტიფიკატორის მქონე კონსტანტა იღებს ხუთის ტოლ მნიშვნელობას.

დაპროგრამების ენა **C++**-ში გამოიყენება შემდეგი არითმეტიკული ოპერაციები:

შეკრება "+", გამოკლება "-", გამრავლება "*", გაყოფა "/" და მთელრიცხვული გაყოფით მიღებული ნაშთის პოვნა "%".

C++-ში მთელრიცხვული გაყოფის შედეგი მთელი რიცხვია, ანუ თუ გაყოფის ოპერაციის ორივე ოპერანდი (გასაყოფი და გამყოფი) მთელი ტიპის მონაცემებია, რეზულტატიც მთელი ტიპისაა, წილადი ნაწილი უბრალოდ იკარგება (ავტომატურად იხსნება) და რჩება მხოლოდ მთელი ნაწილი.

"%" ოპერაცია გამოიყენება მხოლოდ მთელრიცხვა ოპერანდების შემთხვევაში. არითმეტიკული ოპერაციები **C++**-ში სრულდება იმავე თანმიმდევრობით, როგორც მათემატიკაში (ოპერაციათა პრიორიტეტების გათვალისწინებით). არსებულ კანონზომიერებას არღვევს მხოლოდ ფრჩხილები (ისევე, როგორც ეს ხდება მათემატიკაში).

C++-ში არსებული შედარების ოპერაციათა ნიმუშები წარმოდგენილია პირველ ცხრილში.

ლოგიკური ოპერაციები გამოიყენება გამოსახულებაში ორი ან მეტი პირობის გაერთიანების მიზნით.

C++-ში არსებობს შემდეგი ლოგიკური ოპერაციები:

1. "&&" - ლოგიკური "და" ანუ კონიუნქცია, რომელიც მოითხოვს თავისი ყველა ოპერანდის ჭეშმარიტობას.

2. " || " ლოგიკური "ან" ანუ დიზიუნქცია, რომელიც მოითხოვს თავისი ოპერანდებიდან ერთ-ერთის ჭეშმარიტობას.
3. " ! " - ლოგიკური "არა" (უარყოფა), რომელიც ცვლის თავისი ოპერანდის მნიშვნელობას საწინააღმდეგო მნიშვნელობით.

ცხრილი 1

ოპერაციის სახე მათემატიკაში	ოპერაციის სახე **C++**-ში
a=b	***a= =b***
a≠b	***a!=b***
a>b	***a>b***
a≥b	***a>=b***
a<b	***a<b***
a≤b	***a<=b***

C++-ში მინიჭების ოპერაცია საშუალებას იძლევა არითმეტიკული ოპერაციები ჩაიწეროს კომპაქტურად. ეს ოპერაციები წარმოდგენილია მე-2 ცხრილში. (სადაც *a* ცვლადის მნიშვნელობა ხუთის ტოლია. *a*=5).

ცხრილი 2

ოპერაცია	ნიმუში	განმარტება	შედეგი
+=	a+=7	a=a+7	12
-=	a-=2	a=a-2	3
***=**	a*=3	a=a*3	15
/=	a/=2	a=a/2	2
%=	a%=3	a=a%3	2

ოპერაციას, რომლის დროსაც ცვლადის მნიშვნელობა იზრდება ერთით და მიღებული შედეგი ამავე ცვლადში ინახება, ***ინკრემენტის ოპერაცია*** ეწოდება. მაგ: *x*++ ან ++*x*.

ოპერაციას, რომლის დროსაც ცვლადის მნიშვნელობა მცირდება ერთით და მიღებული შედეგი ამავე ცვლადში ინახება, ***დეკრემენტის ოპერაცია*** ეწოდება. მაგ: *x*-- ან –*x*.

ალგორითმულ ენა **C++**-ში არსებობს ერთადერთი ტერნერული **(?:)** - პირობითი ოპერაცია, რომელიც სამი ოპერანდისაგან შედგება. მისი ჩაწერის ზოგადი სახე შემდეგია:

პირობითი გამოსახულება? მოქმედება-1 : მოქმედება-2

ტიპიური მათემატიკური გამოთვლების განსახორციელებლად **C++**-ში გამოიყენება მათემატიკური ფუნქციების ბიბლიოთეკა **<math>**, სადაც თავმოყრილია მათემატიკურ ფუნქციათა უმრავლესობა. არგუმენტების სახით ფუნქციებს გააჩნიათ ***double*** ტიპის მნიშვნელობები და ამავე ტიპის შედეგებსაც იძლევიან, თუმცა შესაძლებელია როგორც არგუმენტების, ისე საბოლოო შედეგების სახით გამოყენებულ იქნას ***float*** ტიპის მნიშვნელობები.

ეს მათემატიკური ფუნქციები წარმოდგენილია მე-3 ცხრილში.

ცხრილი 3

ფუნქციის ჩანაწერი C++-ში	**განმარტება**
sin(x)	*sinx* (*x* არგუმენტი მოიცემა რადიანებში)
cos(x)	*cosx* (*x* არგუმენტი მოიცემა რადიანებში)
tan(x)	*tgx* (*x* არგუმენტი მოიცემა რადიანებში)
asin(x)	*arcsinx* (*x* არგუმენტი მოიცემა რადიანებში)
acos(x)	*arccosx* (*x* არგუმენტი მოიცემა რადიანებში)
atan(x)	*arctgx* (*x* არგუმენტი მოიცემა რადიანებში)
sqrt(x)	$\sqrt{x}$ კვადრატული ფესვი *x*-დან
pow(x,y)	x^y *x* აყვანილი *y* ხარისხში
exp(x)	e^x ექსპონენციალური ფუნქცია

fabs(x)	$\|x\|$ *x*-ის მოდული (აბსოლუტური მნიშვნელობა)
log(x)	*lnx* ნატურალური ლოგარითმი (*e*-ს ფუძით)
log10(x)	*lgx* ათობითი ლოგარითმი(10-ს ფუძით)
ceil(x)	*x*-ის დამრგვალება მეტობით მომდევნო მთელ რიცხვამდე (არანაკლებ *x*-ისა)
floor(x)	*x*-ის დამრგვალება ნაკლებობით მომდევნო მთელ რიცხვამდე (არაუმეტეს *x*-ისა)
fmod(x,y)	*x*-ის *y*-ზე გაყოფის შედეგად მიღებული ნაშთი (ათწილადი რიცხვი)

მაგალითი 1. შევადგინოთ პროგრამა, რომელიც გამოთვლის *C* ცვლადის მნიშვნელობას, თუ:

$$c = by^2; \quad y = \frac{5\sin x/2 + xe^{x-1}}{4 + \cos(A+1)};$$ სადაც A=1,5 x=2.

ამოხსნა:

```
// c ცვლადის მნიშვნელობის გამოთვლა
#include <iostream>
#include <math>
main( )
{
  double A=1.5, x=2, c, b, y;
cout<<"b=";
cin>>b;
y=(5*sin(x/2)+x*exp(x-1))/(4+cos(A+1));
c=b*pow(y,2);
cout<<"y="<<y<<endl;
cout<<"c="<<c<<endl;
return 0;
}
```

დავალება

შეადგინეთ ქვემოთ წარმოდგენილი მაგალითებისა და ამოცანების გადაწყვეტის პროგრამები:

1. $z = y^2 + 1;$ $y = \dfrac{\sin(x^2 + b) + 2}{\sqrt{x^3 - 0{,}1}} e^{x+c} A;$ თუ A=4,5 b=0,25 x=4,5.
2. $z = \ln(y + 5{,}5);$ $y = \dfrac{\sqrt{x^2 + b}e^{-(x+c)}}{2(\sin(x + c) + 1)} A;$ თუ A=7,5 b=3,64 c=0,6 x=0,5.
3. $z = \dfrac{y^2 + 1}{x - A};$ $y = -\dfrac{1}{\sqrt{x^2 - A^2}} + \dfrac{2A^2 + 1}{3\sqrt{x^2 - A^2}};$ თუ A=0,3 x=0,7.
4. $z = y^3 - 1;$ $y = (A - 1{,}1)e^{x^2} \dfrac{1 + \sin x}{1 + \cos^2 x};$ თუ A=5 x=3.
5. $z = y^2 + Ax;$ $y = \left(\dfrac{\cos Ax}{\ln(x^2 + 2)} + \sqrt{Ax} \right)^3;$ თუ A=0,5 x=1,2.
6. $z = Ay \sin y;$ $y = \sqrt{\dfrac{e^x + 1}{e^x + 2} + \sin x};$ თუ A=10 x=1,5.
7. $z = \left| x^2 - y^2 \right|;$ $y = \dfrac{\sqrt{\ln(A + B) + x}}{e^x + 1};$ თუ A=2,8 x=1,5 B=-1.
8. შეადგინეთ პროგრამა, რომელიც გამოთვლის კვადრატის პერიმეტრსა და ფართობს, თუ მისი გვერდის სიგრძეა: a=2,5 სმ.
9. შეადგინეთ პროგრამა, რომელიც გამოთვლის ABC მართკუთხა სამკუთხედში A კუთხის გრადუსულ ზომას, თუ $\angle C$=35°.
10. შეადგინეთ პროგრამა, რომელიც ჰერონის ფორმულის გამოყენებით გამოთვლის ABC სამკუთხედის ფართობს, თუ მისი გვერდების სიგრძეებია: a=2სმ, b=3სმ, c=4სმ.

პროგრამული კოდი და მიღებული შედეგები:

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

ლაბორატორიული სამუშაო №2
განშტოებადი ალგორითმების დაპროგრამება

C++-ში განშტოებადი ალგორითმების დაპროგრამების მიზნით შემდეგი პირობითი და უპირობო გადასვლის ოპერატორები გამოიყენება: ***if***, ***if/else***, ***switch***, ***goto***.

if ოპერატორის ჩაწერის სინტაქსი შემდეგია:

```
if (პირობა)
{
  ბრძანებები;
}
```

საკვანძო სიტყვა ***if*** ნიშნავს "თუ"-ს. პირობის ქვეშ იგულისხმება ნებისმიერი შედარების თუ ლოგიკური ოპერაცია. აღნიშნულ ოპერატორში პირობის შემდეგ დაუშვებელია წერტილმძიმის დასმა, რადგან ეს სინტაქსურ შეცდომად ითვლება. ოპერატორის ტანში არსებული ბრძანებები, რომლებიც მოთავსებულია ღია და დახურულ ფიგურულ ფრჩხილებს "{}" შორის, სრულდება მხოლოდ იმ შემთხვევაში, თუ პირობა ჭეშმარიტია. წინააღმდეგ შემთხვევაში (თუ პირობა მცდარია), ***if*** ოპერატორის ტანში არსებული ბრძანებები არ შესრულდება და მართვა გადაეცემა ***if*** ოპერატორის (დახურული ფიგურული ფრჩხილის შემდგომ) მომდევნო ოპერატორს პროგრამაში. თუ ***if*** ოპერატორის ტანი შეიცავს მხოლოდ ერთ ბრძანებას, ფიგურული ფრჩხილების გამოყენება სავალდებულო არ არის.

მაგალითი 2.1. შევადგინოთ პროგრამა, რომელიც განსაზღვრავს წარმოადგენს თუ არა ნებისმიერი მთელი ტიპის x ცვლადი უარყოფით რიცხვს.

ამოხსნა:

```
#include <iostream>
main( ){
int x;
```

```
cout<<”x=”;
cin>>x;
if(x<0)
cout<<” x is a negative number.”<<endl;
cout<<”The problem is over.”<<endl;
 return 0;
}
```

if/else ოპერატორის ჩაწერის სინტაქსი შემდეგია:

```
if (პირობა)
{
  ბრძანებები-1;
}
else
{
  ბრძანებები-2;
}
```

if/else ოპერატორის მოქმედების პრინციპი მდგომარეობს შემდეგში:

მოწმდება პირობა, რომელიც მოსდევს ***if*** ოპერატორს და თუ ის ჭეშმარიტია, სრულდება "ბრძანებები-1" და არ სრულდება "ბრძანებები-2". თუ პირობა მცდარია, სრულდება "ბრძანებები-2" და არ სრულდება "ბრძანებები-1". ნებისმიერ შემთხვევაში მართვა გადაეცემა აღნიშნული ბლოკის მომდევნო ოპერატორს პროგრამაში.

თუ "ბრძანებები-1" და "ბრძანებები-2" შედგება თითო ბრძანებისგან, ფიგურული ფრჩხილების გამოყენება სავალდებულო არ არის. აქაც, ისევე როგორც წინა შემთხვევაში, პირობის შემდეგ წერტილ-მძიმის დასმა დაუშვებელია და ითვლება სინტაქსურ შეცდომად, რასაც მივყავართ დასმული ამოცანის არასწორ შედეგამდე.

მაგალითი 2.2. შევადგინოთ $ax=b$ წრფივი განტოლების ამოხსნის პროგრამა a და b კოეფიციენტების ნებისმიერი მნიშვნელობების დროს.

ამოხსნა:

```
//წრფივი განტოლების ამოხსნა
#include <iostream>
main( )
{
float a, b, x;
cout<<”a=”;
cin>>a;
cout<<”b=”;
cin>>b;
if(a!=0)
 {
 x=b/a;
 cout<<”x=”<<x<<endl;
 }
 else
 {
 if(b==0)
 cout<<”Any solution”<<endl;
 else
 cout<<”No solution”<<endl;
 }
return 0;
 }
```

switch ოპერატორის ჩაწერის სინტაქსი შემდეგია:

```
switch (გამოსახულება)
{
 გასაღებთა სია;
}
```

გამოსახულების ქვეშ აქ იგულისხმება ნებისმიერი გამოსახულება, რომლის შედეგს წარმოადგენს მთელი ტიპის (***int, long*** ან ***char***) მნიშვნელობა.

გასაღებთა სია შედგება გასაღებთა ნებისმიერი რაოდენობისგან, სადაც ყოველ მათგანს გააჩნია ჩაწერის შემდეგი ფორმა:

***case* მთელი ტიპის მნიშვნელობა:**
ბრძანებები;
break;

ყოველ გასაღებს გააჩნია განსაზღვრული მუდმივი მნიშვნელობა და ბრძანებათა ჩამონათვალი, რომელიც სრულდება ***break*** (შეწყვეტა) ოპერატორით.

break ოპერატორის გამოტოვების შემთხვევაში ბრძანებები სრულდება მომდევნო ***break*** ოპერატორამდე პროგრამაში.

switch ოპერატორს გააჩნია მთელი რიგი ჭდეებისა (***case***) და არასავალდებულო ***default*** ჭდე.

ჭდე შედგება კოდური სიტყვისაგან ***case***, რომელსაც მოსდევს მთელი ტიპის მნიშვნელობა და ორწერტილი.

switch ოპერატორის მუშაობის პრინციპი მდგომარეობს შემდეგში:

1. გამოითვლება გამოსახულება და მისი მნიშვნელობა (მთელი ტიპის) თანმიმდევრულად შედარდება გასაღებთა მნიშვნელობებს (რომლებიც მითითებულია ***case*** ჭდეების შემდეგ).
2. თუ გამოსახულების შედეგი დაემთხვევა რომელიმე გასაღების მნიშვნელობას, შესრულდება ის ბრძანებები, რომლებიც აღნიშნულ ჭდეს მოსდევს ***break*** ოპერატორამდე. კოდური სიტყვა ***default*** გამოიყენება იმ შემთხვევაში, როდესაც გამოსახულების მნიშვნე-

ლობა არ ემთხვევა გასაღებთა არცერთ მნიშვნელობას.

default ჭდის გამოყენება ბლოკში სავალდებულო არ არის, თუმცა მისი მოთავსება ***switch*** მმართველ სტრუქტურაში დაპროგრამების კარგ სტილად ითვლება. ***default*** ჭდე შეიძლება ჩაიწეროს აღნიშნული ბლოკის ნებისმიერ ადგილას, მაგრამ სასურველია, თუ ის მოთავსდება ბლოკის ბოლოს.

მაგალითი 2.3. მოცემულია მონეტა, რომელსაც ვაგდებთ 50-ჯერ. საჭიროა შევადგინოთ პროგრამა, რომელიც გამოთვლის მონეტის თითოეული მხარის მოსვლის ხდომილობას.

ამოხსნა:

```
#include <iostream>
#include <iomanip>
#include <stdlib>
main( )
{
srand(time(0));
int face;
int x1=0, x2=0;
for(int i=1; i<=50; i++)
{
face=1+rand( )%2;
switch(face)
{
case 1:
++x1;
break;
case 2:
++x2;
break;
}}
cout<<"Face"<<setw(10)<<"count"<<endl;
cout<<" x1="<<setw(10)<<x1<<endl;
cout<<" x2="<<setw(10)<<x2<<endl;
```

```
return 0;
}
```

goto ოპერატორის ჩაწერის სინტაქსს აქვს შემდეგი სახე:

goto ჭდე;

აღნიშნული ოპერატორის შესრულების შედეგად მართვა გადაეცემა პროგრამაში იმ პირველ ოპერატორს, რომლის ჭდეც მითითებულია ***goto*** ოპერატორში. ჭდე ამ შემთხვევაში წარმოადგენს იდენტიფიკატორს, რომელიც ბოლოვდება ორწერტილით (:).

მაგალითი 2.4. ***goto*** ოპერატორის გამოყენებით შევადგინოთ პროგრამა, რომელიც ბეჭდავს ერთიდან ათამდე მნიშვნელობებს.

ამოხსნა:

```
// goto ოპერატორის გამოყენების მარტივი პროგრამა
#include <iostream>
main( )
{
 int count=1;
start:                  //ჭდე
if(count>10)
goto end;
cout<<count<<"  ";
++count;
goto start;
end:                    //ჭდე
cout<<endl;
return 0;
}
```

დავალება

შეადგინეთ ქვემოთ წარმოდგენილი მაგალითებისა და ამოცანების გადაწყვეტის პროგრამები:

1. $x = e^{\sqrt{z+1}} \sin z \cos(2{,}6z)$; სადაც z=2,17.

$$y=\begin{cases} arctgx + 4, & \text{თუ } x\le -1; \\ x+4, & \text{თუ } x>1; \\ 1. & \end{cases}$$

2. $A = e^{\sin x}$; $z = \frac{a}{3} tg\left(\frac{\cos(A-1)}{2}\right)$; თუ x=12,56.

$$y=\begin{cases} \sqrt{1+x^2}, & \text{თუ } z>0; \\ (1+x)\ln x^2, & \text{თუ } z<0; \\ \cos(x+1), & \text{თუ } z=0. \end{cases}$$

3. $B=A^2-c^3\cos(5x)$; თუ A=1,2 c=1,81 x=12.

$$y=\begin{cases} 100, & \text{თუ } B<1; \\ Bx^3+c, & \text{თუ } 1\le B<2; \\ x+B, & \text{თუ } B\ge 2. \end{cases}$$

4. $y = \frac{\sin(2-x)+1}{\sqrt{e^{A-x}}}$; თუ x=2, A=2,8, B=-4,2 c=8

$$z=\begin{cases} Ax^3-2, & \text{თუ } y<1; \\ Ax^2+Bx+c, & \text{თუ } y=1; \\ c. \end{cases}$$

5. შეადგინეთ პროგრამა, რომელიც განსაზღვრავს კლავიატურიდან აკრეფილი ნებისმიერი x რიცხვის ნიშანს.
6. შეადგინეთ პროგრამა, რომელიც გამოთვლის $y=sinx/x$ გამოსახულების მნიშვნელობას ნებისმიერი x ცვლადისათვის. (გაითვალისწინეთ პირობა, რომ ნულზე გაყოფა დაუშვებელია.)
7. შეადგინეთ პროგრამა, რომელიც n ცვლადის მნიშვნელობის გათვალისწინებით გამოთვლის:
   - ცილინდრის მოცულობას, თუ $n=1$;
   - კონუსის მოცულობას, თუ $n\neq1$.

   საწყისი მონაცემებია: r=5სმ, h=12სმ.
8. განსაზღვრეთ რა სიჩქარით გაირბენს მორბენალი დისტანციას, თუ საწყისი პირობებია: მოძრაობის დრო 3წთ და 25წმ, ხოლო დისტანციის სიგრძე 1000 მეტრია.
9. შეადგინეთ პროგრამა, რომელიც ნებისმიერ სამ a, b და c რიცხვებს შორის მოძებნის მაქსიმალური (მინიმალური) მნიშვნელობის მქონე რიცხვს.
10. შეადგინეთ პროგრამა, რომელიც კამათლის 25-ჯერ გაგორების შემთხვევაში განსაზღვრავს თვითოეული მხარის მოსვლის ხდომილობას.

პროგრამული კოდი და მიღებული შედეგები:

(ქულა)

(ხელმოწერა)

ლაბორატორიული სამუშაო №3
ციკლის ოპერატორები

ციკლური პროცესების ორგანიზების მიზნით **C++** დაპროგრამების ენაში გამოიყენება ოპერატორები: ***while***, ***do/while***, ***for***.

while ციკლის ჩაწერის სინტაქსი შემდეგია:

```
while (პირობითი გამოსახულება)
{
  ოპერატორები;
}
```

while ციკლში პირობითი გამოსახულება წარმოადგენს ნებისმიერ შედარების ან ლოგიკურ გამოსახულებას, რომელიც ამავე დროს ციკლის პირობაც არის.

აღნიშნულ ციკლს გააჩნია ტანი, რომელიც მოთავსებულია ღია და დახურულ ფიგურულ ფრჩხილებს შორის და მოიცავს ერთ ან რამდენიმე ოპერატორს, რომლებიც სრულდება მანამ, სანამ ციკლის პირობა ჭეშმარიტია.

თუ ციკლის ტანი შედგება ერთი ოპერატორისგან, სავალდებულო არ არის მისი ფიგურულ ფრჩხილებში მოთავსება.

while ციკლში პირობის შემდეგ დაუშვებელია წერტილ-მძიმის დასმა (ითვლება სინტაქსურ შეცდომად).

აღნიშნული ციკლის მუშაობის პრინციპი შემდეგში მდგომარეობს:

- მოწმდება ციკლის პირობა (პირობითი გამოსახულება) და თუ ის ჭეშმარიტია, სრულდება ციკლის ტანში არსებული ოპერატორები მანამდე, სანამ აღნიშნული პირობა არ გახდება მცდარი.
- თუ ციკლის პირობა თავიდანვე მცდარია, ციკლის ტანში არსებული ოპერატორები არ შესრულდება და მართვა გადაეცემა მოცემული ბლოკის (დახურული

ფიგურული ფრჩხილის შემდეგ) მომდევნო ოპერატორს პროგრამაში.

ცხადია, ***while*** ციკლში თუ პირობა არ გახდება მცდარი ალგორითმის გარკვეული ბიჯების შემდეგ, ციკლი არასდროს დასრულდება და პროგრამა ჩაიციკლება. აღნიშნულ შემთხვევაში ციკლი წარმოადგენს უსასრულო ციკლის ტიპიურ მაგალითს.

მაგალითი 3.1. შევადგინოთ პროგრამა, რომელიც განსაზღვრავს ორი მთელი არაუარყოფითი *m* და *n* რიცხვების უდიდეს საერთო გამყოფს ევკლიდეს ალგორითმის საფუძველზე.

ამოხსნა:

```
// ევკლიდეს ალგორითმი
#include <iostream>
main( )
{
 int m, n;
cout<<”m=”;
cin>>m;
cout<<”n=”;
cin>>n;
while(m!=n)
{
if(m>n)
m=m-n;
else
n=n-m;
}
cout<<”m=”<<m<<endl;  // შესაძლებელია აქ n ცვლადის
                      // დაბეჭდვაც, რადგან m=n.
return 0;
}
```

do/while ციკლის ჩაწერის სინტაქსს აქვს შემდეგი სახე:

```
do
{
  ოპერატორები;
}
while(პირობითი გამოსახულება);
```

კოდური სიტყვა "***do***" მიუთითებს ციკლის დასაწყისზე, ხოლო ***while*** (მანამდე სანამ), რომელსაც ახლავს ფრჩხილებში მოთავსებული პირობითი გამოსახულება ციკლის დასასრულზე. ფიგურულ ფრჩხილებში მოთავსებული ოპერატორები წარმოადგენს ციკლის ტანის ოპერატორებს. თუ ციკლის ტანი შედგება ერთი ოპერატორისგან, სავალდებულო არ არის მისი ფიგურულ ფრჩხილებში მოთავსება.

პირობითი გამოსახულების ქვეშ იგულისხმება ნებისმიერი შედარების თუ ლოგიკური ოპერაცია, რომელიც აუცილებელია დასრულდეს წერტილ-მძიმით და შემოწმდეს ციკლის ბოლოს.

do/while ციკლის მუშაობის პრინციპი შემდეგში მდგომარეობს:

- თავდაპირველად ციკლის ტანი სრულდება ერთხელ და შემდეგ მოწმდება ციკლის გაგრძელების პირობა. თუ პირობა ჭეშმარიტია, ციკლის ოპერატორები სრულდება მანამ, სანამ აღნიშნული პირობა მცდარი არ გახდება.
- თუ ციკლის გაგრძელების პირობა თავიდანვე მცდარია, ციკლის ოპერატორები ერთხელ მაინც სრულდება და შემდეგ მართვა გადაეცემა აღნიშნული ბლოკის მომდევნო ოპერატორს პროგრამაში.

მაგალითი 3.2. შევადგინოთ პროგრამა, რომელიც გამოთვლის მოცემული მწკრივის პირველი ოცი წევრის ჯამს:

$$s = 1 + x + \frac{x^2}{2} + \frac{x^3}{3} + \ldots + \frac{x^n}{n}.$$

ამოხსნა:

```
// მწკრივის პირველი ოცი წევრის ჯამის გამოთვლა
#include <iostream>
main( )
{
float x, p=1, s=1, n=20, i=1;
cout<<"x=";
cin>>x;
do
{
p*=x;
s+=p/i;
i++;
}
while(i<n);
cout<<"result="<<s<<endl;
return 0;
}
```

for ციკლის ჩაწერის სინტაქსს შემდეგი სახე აქვს:

```
for( გამოსახულება-1; გამოსახულება-2; გამოსახულება-3)
{
  ოპერატორები;
}
```

"გამოსახულება-1" წარმოადგენს ცვლადს, რომელსაც ენიჭება საწყისი მნიშვნელობა. აღნიშნული ცვლადი გვევლინება მთვლელის როლში, რომელიც მართავს ციკლს. ცხადია, მთვლელი უნდა იყოს მთელი ტიპის მონაცემი.

"გამოსახულება-1" სრულდება მხოლოდ ერთხელ ციკლის დასაწყისში. "გამოსახულება-2" წარმოადგენს ციკლის გაგრძელების პირობას, იმდენად რამდენადაც, სანამ პირობა ჭეშმარიტია, სრულდება ციკლის ტანის ოპერატორები.

"გამოსახულება-2" მოწმდება ყოველი იტერაციის წინ.

"გამოსახულება-3" წარმოადგენს ციკლის მმართველი ცვლადის (მთვლელის) ცვლილების ოპერაციას. იგი სრულდება ყოველი იტერაციის ბოლოს.

ოპერატორები, რომლებიც მოთავსებულია ღია და დახურულ ფიგურულ ფრჩხილებს შორის, შეადგენენ ციკლის ტანს.

თუ ციკლის ტანი შეიცავს ერთ ოპერატორს, მისი ფიგურულ ფრჩხილებში მოთავსება სავალდებულო არ არის.

for სტრუქტურის სამივე შემადგენელი ნაწილი: მთვლელის საწყისი მნიშვნელობა, ციკლის გაგრძელების პირობა და მთვლელის მნიშვნელობის ცვლილება ერთმანეთისგან გამოიყოფა წერტილ-მძიმით (;).

for ციკლის მუშაობის პრინციპი შემდეგში მდგომარეობს:

- მოწმდება ციკლის გაგრძელების პირობა და თუ ის ჭეშმარიტია, ციკლის ტანის ოპერატორები სრულდება მანამდე, სანამ აღნიშნული პირობა მცდარი არ გახდება.
- თუ ციკლის გაგრძელების პირობა თავიდანვე მცდარია, მართვა გადაეცემა აღნიშნული ბლოკის მომდევნო ოპერატორს პროგრამაში ისე, რომ ციკლის ტანში არსებული ბრძანებები არ შესრულდება.

მაგალითი 3.3. შევადგინოთ პროგრამა, რომელიც გამოთვლის არაუარყოფითი **N** რიცხვის ფაქტორიალის მნიშვნელობას.

ამოხსნა:

```
// არაუარყოფითი N რიცხვის ფაქტორიალის გამოთვლა
#include <iostream>
main( )
{
unsigned long factorial=1;
int number;
cout<<”number=”;
cin>>number;
```

```
for(int counter=number; counter>=1; counter--)
factorial*=counter;
cout<<number<<"!="<<factorial<<endl;
return 0;
}
```

<u>დავალება</u>

1. ***while*** ციკლის გამოყენებით შეადგინეთ პროგრამა, რომელიც გამოთვლის $y=x^2-2x+5$ ფუნქციის მნიშვნელობათა ცხრილს, თუ x არგუმენტი [1;2] ინტერვალში იცვლება $h=0,01$ ბიჯით.
2. ***for*** ციკლის გამოყენებით შეადგინეთ პირველი 15 ნატურალური რიცხვის ნამრავლისა და ჯამის გამოთვლის პროგრამა.
3. ***do/while*** ციკლის გამოყენებით შეადგინეთ $z=\sqrt{5x^2+7y^3+\cos^2 xy}$ ფუნქციის გამოთვლის პროგრამა, თუ x იცვლება 1-დან 11-მდე $h_1=0,02$ ბიჯით, ხოლო y იცვლება 5-დან $h_2=0,01$ ბიჯით.
4. მოცემულია $y=3x^4-5x^3-x^2-6x-3$ ფუნქცია, რომელიც გამოითვლება [-2;5,8] მონაკვეთზე $h=0,02$ ბიჯით. შეადგინეთ პროგრამა, რომელიც განსაზღვრავს ამ ფუნქციის დადებითი მნიშვნელობების საშუალო არითმეტიკულს.
5. შეადგინეთ შემდეგი მწკრივის ელემენტების ჯამის გამოთვლის პროგრამა: $s=1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{100}$.
6. შეადგინეთ $y=\frac{\cos x}{x}$ ფუნქციის მნიშვნელობათა ცხრილის გამოთვლის პროგრამა, თუ x არგუმენტი [-10;25] ინტერვალში იცვლება $h=0,1$ ბიჯით.

7. მოცემულია $y = \left(\frac{x}{x^2 + 1} - e^{-x} \right) \sin x$ ფუნქცია, რომელიც გამოითვლება [-2,3;3] მონაკვეთზე h=0,005 ბიჯით. შეადგინეთ პროგრამა, რომელიც გამოთვლის y ფუნქციის არაუარყოფითი მნიშვნელობების ნამრავლს.
8. შეადგინეთ პროგრამა, რომელიც მონეტის 20-ჯერ აგდების შემთხვევაში გამოთვლის თვითოეული მხარის მოსვლის ხდომილობას და შედეგებს წარმოადგენს ცხრილის სახით, რომელიც ოთხ სტრიქონსა და ხუთ სვეტს შეიცავს.
9. შეადგინეთ პროგრამა, რომელიც გამოთვლის ათი მოსწავლისგან შემდგარ კლასში მოსწავლეთა მოსწრების საშუალო შედეგს 100-ქულიანი სისტემით. (შეფასება წარმოებს 1-დან 100-მდე ქულათა დიაპაზონში მთელი ტიპის რიცხვითი მონაცემების გამოყენებით).
10. შეადგინეთ პროგრამა, რომელიც დაბეჭდავს მნიშვნელობათა შემდეგ ცხრილს:

№	10№	100№	1000№
1	10	100	1000
2	20	200	2000
3	30	300	3000
4	40	400	4000
5	50	500	5000

პროგრამული კოდი და მიღებული შედეგები:

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

ლაბორატორიული სამუშაო №4
ჩალაგებული პირობები და ციკლები. ციკლებიდან ალტერნატიული გამოსვლის შესაძლებლობები

ხშირად იმ მიზნით, რომ გადავწყვიტოთ, თუ რომელი მოქმედების შესრულებაა საჭირო, აუცილებელი ხდება რამდენიმე პირობის შემოწმება. **C++**-ში ეს ხდება მაშინ, როდესაც ერთი ***if*** (ან ***if/else***) მმართველი სტრუქტურის შიგნით თავსდება მეორე ***if*** (ან ***if/else***) მმართველი სტრუქტურა. შესაბამისად მიიღება ერთმანეთში ჩალაგებული პირობები, სადაც შიდა ***if*** (ან ***if/else***) სტრუქტურა სრულდება მხოლოდ იმ შემთხვევაში, თუ გარე ***if*** სტრუქტურის პირობა ჭეშმარიტია.

ჩალაგებული პირობების ერთგვარ ანალოგიას წარმოადგენს ჩალაგებული ციკლები, სადაც ერთი (გარე) ციკლის შიგნით თავსდება ერთი ან რამდენიმე შიდა ციკლი. ჩალაგებული ციკლების შემთხვევაში გარე ციკლის ტანის შესასრულებლად საჭიროა შესრულდეს შიდა ციკლის ყველა გამეორება.

ჩალაგებული პირობები და ციკლები რთული პირობებისა და ციკლების კერძო შემთხვევას წარმოადგენს.

მაგალითი 4.1. შევადგინოთ პროგრამა, რომელიც გამოთვლის მთელი ტიპის ნებისმიერი ათი დადებითი კენტი რიცხვის ჯამს.

ამოხსნა:

```
// მთელი ტიპის ნებისმიერი დადებითი კენტი რიცხვების
//ჯამის გამოთვლა
#include <iostream>
main( )
{
int x, Sum=0;
for(int i=1; i<=10; i++)
{
```

```
cout<<”x=”;
cin>>x;
if(x>0)
{
if(x%2= =1)
Sum+=x;
}}
cout<<”Sum=”<<Sum<<endl;
return 0;
}
```

მაგალითი 4.2. შევადგინოთ პროგრამა, რომელიც ჩალაგებული ციკლების გამოყენებით მონიტორზე დაბეჭდავს შემდეგ სურათს:

```
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
```

ამოხსნა:

```
#include <iostream>
main( )
{
for(int i=1; i<=4; i++)
{
for(int j=1; j<=8; j++)
cout<<"*"<<" ";
cout<<endl;
for(int k=1; k<=8; k++)
cout<<" *"<<"";
cout<<endl;
```

```
}
return 0;
}
```

break და ***continue*** ოპერატორები ცვლიან პროგრამებში მართვის ნაკადს. როდესაც ***break*** ოპერატორი სრულდება ***while***, ***for*** ***do/while*** ან ***switch*** სტრუქტურებში, ადგილი აქვს სტრუქტურებიდან დაუყოვნებლივ გამოსვლას; ე.ი. ***break*** ოპერატორი წარმოადგენს ციკლებიდან ალტერნატიული გამოსვლის შესაძლებლობას. ეს კი ნიშნავს, რომ ციკლში მიმდინარე პროცესები წყდება მასში ყველა იტერაციის დასრულებამდე და მართვა გადაეცემა ბლოკის მომდევნო ოპერატორს პროგრამაში.

continue ოპერატორი ***while***, ***for*** ან ***do/while*** სტრუქტურებში იწვევს აღნიშნული ციკლების ტანის ნაწილის გამოტოვებას და შემდეგი იტერაციების შესრულებას. მაშასადამე, ***continue*** ოპერატორი ციკლურ სტრუქტურებში წყვეტს მხოლოდ მიმდინარე იტერაციას და ახორციელებს შემდეგი იტერაციების გაგრძელებას.

მაგალითი 4.3. შევადგინოთ პროგრამა, რომელიც გამოთვლის მოცემული $y=x^5+3x^4-12$ ფუნქციის მნიშვნელობათა ცხრილს, თუ x იცვლება [2;11] შუალედში h=0,1 ბიჯით. ამასთან, როცა y-ის მნიშვნელობა გადააჭარბებს 6000-ს, დავასრულოთ გამოთვლები.

ამოხსნა:

```
// y=x^5+3x^4-12 ფუნქციის მნიშვნელობათა ცხრილის გამოთვლა
#include <iostream>
#include <math>
main( )
{
float h=0.1, x=2, y;
do{
```

```
if(y>6000)
break;
y=pow(x,5)+3*pow(x,4)-12;
cout<<"x="<<x<<endl;
cout<<"y="<<y<<endl;
x+=h;
}while(x<=11);
return 0;
}
```

მაგალითი 4.4. შეავდგინოთ პროგრამა, რომელიც ბეჭდავს ყველა ციფრს 1-დან 9-ის ჩათვლით გარდა 5-სა.

ამოხსნა:

```
// 1-დან 9-ის ჩათვლით 5-ის გარდა ციფრების დაბეჭდვა
#include <iostream>
#include <iomanip>
main( )
{
for(int x=1; x<=9; x++)
{
if(x==5)
continue;
cout<< setw(5)<<x;
}
return 0;
 }
```

<u>დავალება</u>

1. შეადგინეთ პროგრამა, რომელიც ბეჭდავს შემდეგ სურათს:

```
* * * * * * * * * *
 * * * * * * * * * *
* * * * * * * * *
```

```
* * * * * * *
 * * * * * *
  * * * * *
   * * * *
    * * *
     * *
      *
```

2. შეადგინეთ პროგრამა, რომელიც გამოთვლის კლავიატურიდან შეყვანილი 15 დადებითი რიცხვის ჯამიდან მიღებული კვადრატული ფესვის მნიშვნელობას. თუ კლავიატურიდან მოხდება უარყოფითი რიცხვის აკრეფა, გამოთვლითი პროცესი შეწყვიტეთ.
3. შეადგინეთ პროგრამა, რომელიც მონიტორზე დაბეჭდავს 10-დან 20-მდე მთელი ტიპის რიცხვითი მონაცემების კვადრატების მნიშვნელობებს გარდა 15-ის და 18-ის კვადრატისა.
4. შეადგინეთ + = 0 წრფივი განტოლების ამოხსნის პროგრამა *a* და *b* კოეფიციენტების ნებისმიერი მნიშვნელობების დროს.
5. შეადგინეთ + + = 0 კვადრატული განტოლების ამოხსნის პროგრამა *a*, *b* და *c* კოეფიციენტების ნებისმიერი მნიშვნელობების დროს.

6. შეადგინეთ პროგრამა, რომელიც ბეჭდავს შემდეგ სურათს:

```
      *
     * *
    * * *
   * * * *
  * * * * * *
```

```
      * * * * * *
     * * * * * * *
    * * * * * * * *
   * * * * * * * * *
  * * * * * * * * * *
```

7. შეადგინეთ პროგრამა, რომელიც გამოთვლის მთელი ტიპის ნებისმიერი რვა უარყოფითი ლუწი რიცხვის საშუალო არითმეტიკულს.
8. შეადგინეთ პროგრამა, რომელიც სამკუთხედის *a*, *b* და *c* გვერდების მოცემული მნიშვნელობების დროს განსაზღვრავს არის თუ არა *abc* სამკუთხედი ტოლგვერდა.
9. შეადგინეთ პროგრამა, რომელიც განსაზღვრავს კლავიატურიდან თქვენ მიერ შეტანილი ნებისმიერი რიცხვითი მონაცემის ნიშანს (თუ მნიშვნელობა ნულის ტოლია, პროგრამამ გამოიტანოს სათანადო შეტყობინება).
10. შეადგინეთ პროგრამა, რომელიც წარმოადგენს კამათლის 15-ჯერ გაგორების შედეგებს, მაგრამ თუ გაგორების შედეგად კამათელზე აღმოჩნდება ციფრი "ერთი", შეწყვეტს პროგრამას და გამოიტანს შეტყობინებას: "თქვენ წააგეთ", ხოლო ექვსიანის შემთხვევაში შეწყვეტს პროგრამას და წარმოადგენს შეტყობინებას: "თქვენ მოიგეთ".

პროგრამული კოდი და მიღებული შედეგები:

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

ლაბორატორიული სამუშაო №5
ერთგანზომილებიანი მასივები და მათი გამოყენების ნიმუშები

მასივი წარმოადგენს კომპიუტერის მეხსიერებაში არსებული მიმდევრობითი უჯრედების ჯგუფს, რომელთაც გააჩნიათ საერთო სახელი და ერთიდაიგივე ტიპი. ყოველი მასივი ხასიათდება ზომით და განზომილებით.

მასივში შემავალი ელემენტების რაოდენობა განსაზღვრავს მის ზომას. თითოეულ ელემენტს შეიძლება გააჩნდეს ერთი ან მეტი ინდექსი, რაც განაპირობებს მასივის განზომილების განსაზღვრას.

თუ მასივში შემავალ ყოველ ელემენტს გააჩნია თითო ინდექსი, მას ერთგანზომილებიანი მასივი ანუ ***ვექტორი*** ეწოდება.

დაპროგრამების ენა C++-ში ყოველ მასივს გააჩნია ნულოვანი ელემენტი, ანუ როგორც წესი, ნებისმიერი მასივის პირველი ელემენტის ინდექსი ნულის ტოლია. მასივის ელემენტის ინდექსი მთელი ტიპის რიცხვია, ან გამოსახულება, რომლის შედეგი ასევე მთელი ტიპისაა.

მასივი, მსგავსად ნებისმიერი სხვა ობიექტისა, პროგრამაში გამოყენებამდე უნდა იქნას გაცხადებული, რათა კომპილატორმა წინასწარ მოახდინოს მისთვის კომპიუტერის მეხსიერებაში საჭირო მოცულობის გამოყოფა. ამ მიზნით პროგრამისტმა აუცილებელია მიუთითოს მასივში შემავალი ელემენტების ტიპი, მათი რაოდენობა და მასივის იდენტიფიკატორი. შესაძლებელია ერთი და იმავე ტიპის მქონე ელემენტებისგან შემდგარი რამდენიმე მასივის ერთდროულად აღწერაც. მაგალითად:

int b[100], a[25];

ამ შემთხვევაში აღიწერება ასი მთელრიცხვა ელემენტისგან შემდგარი **b** მასივი და ამავე ტიპის ოცდახუთი ელემენტისგან შემდგარი **a** მასივი.

მასივის ელემენტებს საწყისი მნიშვნელობები შეიძლება მიენიჭოს მისი გაცხადების დროს. ამ პროცესს ელემენტების ***ინიციალება*** ეწოდება. აღნიშნულ შემთხვევაში მასივის ელემენტების მნიშვნელობები თავსდება ფიგურულ ფრჩხილებში და ერთიმეორისგან მძიმით გამოიყოფა. მაგალითად:

int A[7]={1, 2, 5, 10, 0, -5, 8};

თუ მასივში არსებული ელემენტების რაოდენობა აღმოჩნდება მეტი მნიშვნელობათა რაოდენობაზე, დარჩენილი ელემენტები ავტომატურად იღებენ საწყის ნულოვან მნიშვნელობებს.

როგორც ზემოთ ავღნიშნეთ, მასივში შემავალი ელემენტების რიცხვი განსაზღვრავს მის ზომას. ამდენად, მასივის ზომა მუდმივი სიდიდეა და სასურველია პროგრამებში ის აღიწეროს მასივთა გაცხადებამდე. ასე მაგალითად:

```
const int size=10;     // მასივის ზომა განისაზღვრა, როგორც
                       // მთელი ტიპის მუდმივი სიდიდე.
int A[size];           // აღიწერა მთელრიცხვა მასივი "A"
                       // ზომით size.
```

მაგალითი 5.1. შევადგინოთ პროგრამა, რომელიც ათი მთელი ტიპის მნიშვნელობის მქონე ელემენტისგან შემდგარ **A** მასივის ელემენტებს მიანიჭებს საწყის ლუწ მნიშვნელობებს, გამოთვლის მათ საშუალო არითმეტიკულს, ნამრავლს და დაბეჭდავს საწყის მასივსა და მიღებულ შედეგებს.

ამოხსნა:

```
// ჯამისა და ნამრავლის დაგროვება მასივში
#include <iostream>
#include <iomanip>
 main( )
{
 const int size=10;  // მასივის ზომა
 int  A[size];       // მასივის გაცხადება
 int  S=0;           // S ცვლადში ინახება ელემენტების ჯამი
```

```
 unsigned long M=1; // M ცვლადში ინახება ელემენტების
                    // ნამრავლი
for(int i=0; i<size; i++)
{
 A[i]=2+2*i;
  S+=A[i];
  M*=A[i];
}
 S/=size;
 cout<<"Current array: "<<endl;
 for(int i=0; i<size; i++)
 cout<<setw(5)<<A[i];
 cout<<endl<<"Average="<<S<<endl;
 cout<<"Multiple="<<M<<endl;
 return 0;
}
```

მაგალითი 5.2. შეავდგინოთ პროგრამა, რომელიც ოცი ელემენტისგან შემდგარ **B** მასივის ელემენტებს მიანიჭებს ნებისმიერ მთელ მნიშვნელობებს, გამოთვლის მოცემული მასივის კენტინდექსიანი და ამავდროულად ლუწი მნიშვნელობის მქონე ელემენტების კვადრატების ჯამს, დაბეჭდავს საწყის მასივს და მიღებულ შედეგს.

ამოხსნა:

```
#include <iostream>
#include <iomanip>
#include <math>
 main( )
{
 const int size=20;
 int  B[size];
 long S=0;
 for(int i=0; i<size; i++)
{
 cout<<"B["<<i<<"]=";
```

```
 cin>>B[i];
 if (i%2= =1 && B[i]%2= =0)
 S+=pow(B[i], 2);
}
 cout<<endl<<"Current array: "<<endl;
 for(int i=0; i<size; i++)
 cout<<setw(5)<<B[i];
 cout<<endl<<"S="<<S;
 cout<<endl;
 return 0;
}
```

მაგალითი 5.3. შევადგინოთ პროგრამა, რომელიც წაიკითხავს მოცემული ათელემენტიანი **A** მასივიდან რიცხვით მნიშვნელობებს და წარმოადგენს მასივს შესაბამის ჰისტროგრამასთან ერთად.

ამოხსნა:

```
//ჰისტროგრამის ბეჭდვის პროგრამა
#include <iostream>
#include <iomanip>
main( )
{
const int size=10;
int A[size];
for(int i=0; i<size; i++)
A[i]=i+1;
cout<<"Item"<<setw(8)<<"Count"<<setw(10)<<"Histro"<<
<<endl;
 for(int i=0;i<size;i++)
 {
 cout<<setw(3)<<i<<setw(8)<<A[i]<<" ";
 for(int j=1; j<=A[i]; j++)
 cout<<setw(4)<<'*';
 cout<<endl;}
 return 0;
 }
```

დავალება

1. შეადგინეთ პროგრამა, რომელიც **A=(A_0, A_1,…,A_{10})** მასივის ელემენტებს მიანიჭებს ნებისმიერ მნიშვნელობებს, გამოთვლის მასივის დადებითი ელემენტების ნახევარჯამს და უარყოფითი მნიშვნელობის მქონე ელემენტების საშუალო არითმეტიკულს.
2. შეადგინეთ პროგრამა, რომელიც **B=(B_0, B_1,…,B_{17})** მასივის ელემენტებს მიანიჭებს ერთიდან დაწყებული კენტ მნიშვნელობებს ზრდადობის მიხედვით და დაბეჭდავს მიღებულ მასივს შესაბამის ჰისტროგრამასთან ერთად.
3. შეადგინეთ პროგრამა, რომელიც **A=(A_0, A_1,…,A_{19})** მასივის დადებით ელემენტებს შორის მოძებნის უდიდესი მნიშვნელობის მქონე ელემენტს და უარყოფითი მნიშვნელობის მქონე ელემენტებს შორის - უმცირეს ელემენტს.
4. შეადგინეთ პროგრამა, რომელიც გამოთვლის **M=(M_0, M_1,…,M_{12})** მასივის ლუწინდექსიანი ელემენტების ნამრავლსა და კენტინდექსიანი ელემენტების კუბების ჯამს.
5. შეადგინეთ პროგრამა, რომელიც გამოთვლის **N=(N_0, N_1,…,N_{20})** მასივის მოდულით უდიდესი და მოდულით უმცირესი ელემენტების ნამრავლს.
6. შეადგინეთ პროგრამა, რომელიც გამოთვლის **C=(C_0, C_1,…,C_{14})** მასივის ლუწინდექსიანი და ამავდროულად არაუარყოფითი მნიშვნელობების მქონე ელემენტების საშუალო არითმეტიკულს.
7. შეადგინეთ პროგრამა, რომელიც **b=(b_0, b_1,…,b_{19})** მასივის დადებით ელემენტებს შორის მოძებნის უმცირეს ელემენტსა და მის ინდექსს.

8. შეადგინეთ პროგრამა, რომელიც მასივისა და ჩალაგებული ციკლების გამოყენებით წარმოადგენს შემდეგ ნახაზს:

   * * * * * * *

   * * * * * *

   * * * * *

   * * * *

   * * *

   * *

   *

9. მოცემულია ორი ვექტორი: **A=(A_0, A_1,…,A_9)** და **B=(B_0, B_1,…,B_9)**. შეადგინეთ პროგრამა, რომელიც მოძებნის ამ ვექტორთა უმცირესი მნიშვნელობების ელემენტებს შორის უდიდესს.
10. შეადგინეთ პროგრამა, რომელიც ათელემენტიან მთელრიცხვა მასივში გამოთვლის ყველა იმ ელემენტის ნამრავლს, რომელთა მნიშვნელობა 10-ზე მეტია და ყველა იმ ელემენტის კვადრატების ჯამს, რომელთა მნიშვნელობა 10-ზე ნაკლებია. მიღებულ შედეგებს კი შეადარებს ერთმანეთს.

პროგრამული კოდი და მიღებული შედეგები:

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

ლაბორატორიული სამუშაო №6
მასივის ელემენტების დახარისხება

მონაცემთა დახარისხება განსაზღვრული წესით ზრდადობის ან კლებადობის მიხედვით თავისი არსით კომბინატორული ამოცანების კლასს მიეკუთვნება. სპეციალისტთა აზრით, კომპიუტერული დროის დაახლოებით 25% სისტემატურად იხარჯება დახარისხების ამოცანებზე. ამიტომ აღნიშნული ალგორითმები განსაკუთრებულ ყურადღებას იმსახურებენ.

მასივთა დახარისხების კლასიკური მეთოდებიდან ჩვენ განვიხილავთ ბუშტისებრი დახარისხების (იგივე "ჩაძირვის"), მარტივი გადანაცვლებისა და კომბინირებული დახარისხების მეთოდებს. აღნიშნული მეთოდებიდან უმარტივესია "ჩაძირვის" მეთოდი, რომელიც საჭიროებს ერთით ნაკლები რაოდენობის ეტაპის შესრულებას ვიდრე მასივის ელემენტების რაოდენობაა და ყოველ ეტაპზე ერთით ნაკლები შედარების ოპერაციის განხორციელებას მასივის ელემენტების საერთო რაოდენობასთან შედარებით. სწორედ ეს უკანასკნელი წარმოადგენს ამ მეთოდის ნაკლს, რადგან შესაძლოა, მასივი პირველ ან თუნდაც მეორე, მესამე ეტაპზე დახარისხდეს და ზედმეტი ეტაპების განხორციელება კომპიუტერული დროის ზედმეტ დანახარჯებს გამოიწვევს. აღნიშნული ნაკლის აღმოსაფხვრელად შემუშავებულ იქნა მარტივი გადანაცვლების მეთოდი, სადაც შემოტანილია ერთგვარი მთვლელი, რომლის შიგთავსი ყოველი ეტაპის ბოლოს მოწმდება და მთვლელის ნულოვნი მნიშვნელობა მიგვითითებს იმაზე, რომ მასივი დახარისხებულია და პროცესი შეგვიძლია შევწყვიტოთ.

დახარისხების კომბინირებული მეთოდი ზემოთ აღწერილი ორივე მეთოდის კომბინაციას წარმოადგენს. ამასთან, ის ელემენტები, რომელთაც გარანტირებულად დაიკავეს თავიანთი ადგილები მასივის ბოლოს (ზრდადობით დახარისხების შემთხვე-

ვაში), შემდგომ ეტაპებზე აღარ განიხილება, რაც მნიშვნელოვნად ზრდის დახარისხების პროცესის სწრაფქმედებას. მეთოდში გათვალისწინებულია ციკლური პროცესებიდან ალტერნატიული გამოსვლის შესაძლებლობაც, თუ მთვლელში აღმოჩნდება ნულოვანი მნიშვნელობა.

მაგალითი 6.1. შევადგინოთ პროგრამა, რომელიც კომბინირებული მეთოდით ზრდადობით დაახარისხებს ათელემენტიანი მთელრიცხვა მასივის ელემენტებს.

ამოხსნა:

```
//კომბინირებული დახარისხების მეთოდი
#include <iostream>
#include <iomanip>
main( )
{
const int n=10;
float A[n], temp;
int i, l, k;
for(i=0; i<n; i++)
{
cout<<"A["<<i<<"]=";
cin>>A[i];
}
cout<<endl<<"Current array:"<<endl;
for(i=0; i<n; i++)
cout<<setw(5)<<A[i];
for(l=1; l<n; l++)
{
k=0;
for(i=0; i<n-l; i++)
{
if(A[i]>A[i+1])
{
temp=A[i];
A[i]=A[i+1];
```

```
A[i+1]=temp;
k++;
}}
if(k= =0)
break;
}
cout<<endl<<"Sorted array:"<<endl;
for(i=0; i<n; i++)
cout<<setw(5)<<A[i];
cout<<endl;
return 0;
}
```

დავალება

1. მოცემულია მასივი **A=(A_0, A_1, …, A_{19})**. შეადგინეთ პროგრამა, რომელიც მარტივი გადანაცვლების მეთოდის გამოყენებით მოახდენს მასივის ელემენტების დახარისხებას კლებადობით.
2. მოცემულია მასივი **B=(B_0, B_1, …, B_9)**. შეადგინეთ პროგრამა, რომელიც ბუშტისებრი დახარისხების მეთოდის გამოყენებით მოახდენს მასივის ელემენტების დახარისხებას კლებადობით.
3. მოცემულია მასივი **C=(C_0, C_1, …, C_{14})**. შეადგინეთ პროგრამა, რომელიც მოახდენს მასივის ელემენტების დახარისხებას კლებადობის მიხედვით კომბინირებული დახარისხების მეთოდის გამოყენებით.
4. მოცემულია მასივი **A=(A_0, A_1, …, A_9)**. შეადგინეთ პროგრამა, რომელიც მასივის ელემენტებს ერთიდან დაწყებული ზრდადობის მიხედვით მიანიჭებს კენტ მნიშვნელობებს და "ჩაძირვის" მეთოდით" დაახარისხებს მათ კლებადობით.

5. მოცემულია მასივი **A=(A_0, A_1, …, A_{19})**. შეადგინეთ პროგრამა, რომელიც მოახდენს **A** მასივის მარტივი გადანაცვლების მეთოდით დახარისხებას კლებადობის მიხედვით და განსაზღვრავს შუა ელემენტების მნიშვნელობებს საწყის და დახარისხებულ მასივებში.
6. მოცემულია მასივი **B=(B_0, B_1, …, B_{14})** შეადგინეთ პროგრამა, რომელიც მოახდენს **B** მასივის მარტივი გადანაცვლების მეთოდით დახარისხებას კლებადობის მიხედვით და განსაზღვრავს ბოლო ელემენტების მნიშვნელობებს საწყის და დახარისხებულ მასივებში.
7. მოცემულია მასივი **A=(A_0, A_1, …, A_{19})**. შეადგინეთ პროგრამა, რომელიც დაახარისხებს მასივის ელემენტებს ზრდადობით, განსაზღვრავს დახარისხებული მასივის შუა ელემენტის მნიშვნელობასა და ინდექსს.
8. მოცემულია მასივი **C=(C_0, C_1, …, C_9)** შეადგინეთ პროგრამა, რომელიც მოახდენს **C** მასივის კომბინირებული დახარისხების მეთოდით კლებადობის მიხედვით დახარისხებას და განსაზღვრავს ნულოვანი ელემენტების მნიშვნელობებს საწყის და დახარისხებულ მასივებში.
9. მოცემულია მასივი **X=(X_0, X_1, …, X_{13})**. შეადგინეთ პროგრამა, რომელიც მოახდენს **X** მასივის ბუშტისებრი დახარისხების მეთოდით კლებადობის მიხედვით დახარისხებას და გამოთვლის შუა ელემენტების მნიშვნელობათა ნახევარჯამს საწყის და დახარისხებულ მასივებში.
10. მოცემულია მასივი **D=(D_0, D_1, …, D_{17})**. შეადგინეთ პროგრამა, რომელიც მოახდენს **D** მასივის ბუშტისებრი დახარისხების მეთოდით ზრდადობის მიხედვით დახარისხებას და გამოთვლის შუა ელემენტების მნიშვნელობათა კვადრატების ჯამს საწყის და დახარისხებულ მასივებში.

პროგრამული კოდი და მიღებული შედეგები:

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

(ქულა) (ხელმოწერა)

ლაბორატორიული სამუშაო №7
ძებნის მეთოდების გამოყენება მასივებში

ხშირ შემთხვევაში პროგრამისტებს მუშაობა უწევთ მონაცემთა დიდ მოცულობასთან, რომელიც წარმოდგენილია მასივების სახით. ზოგჯერ საჭიროა განისაზღვროს, შეიცავს თუ არა მასივი ამათუიმ მნიშვნელობის მქონე ელემენტს. აღნიშნულ პროცესს ეწოდება ***ძებნა*** მასივში. არსებობს საჭირო მნიშვნელობის მქონე ელემენტის ძიების ორი მეთოდი:

1. მარტივი წრფივი ძებნის მეთოდი;
2. ორმაგი ძებნის ანუ დიხოტომიის მეთოდი.

წრფივი ძებნის მეთოდის არსი მდგომარეობს შემდეგში: წინასწარ მოიცემა განსაზღვრული მნიშვნელობის მქონე ცვლადი და ის თანმიმდევრობით შედარდება მასივის ყველა ელემენტის მნიშვნელობას. თუ მასივის რომელიმე ელემენტის მნიშვნელობა დაემთხვევა მოცემული ცვლადის მნიშვნელობას, პროგრამა გამოიტანს შესაბამისი ცვლადის ინდექსს; წინააღმდეგ შემთხვევაში პროგრამა დასრულდება შეტყობინებით: "**We didn't find this item**", რაც ნიშნავს, რომ საჭირო ელემენტი მასივში ვერ მოიძებნა.

ამგვარად, ძებნის პროცესი აღნიშნული მეთოდის გათვალისწინებით გრძელდება მანამდე, სანამ არ შემოწმდება მასივის ყველა ელემენტი და არ მოიძებნება მათ შორის ისეთი, რომლის მნიშვნელობაც ტოლი იქნება მოცემული ცვლადის მნიშვნელობის (ცხადია, თუ ასეთი ელემენტი მასივში საერთოდ არსებობს).

აღნიშნული მეთოდი მარტივია დაპროგრამების თვალსაზრისით, მაგრამ დიდი ზომის მასივების შემთხვევაში ხასიათდება არაეფექტურობით, იმდენად, რამდენადაც ძებნის პროცესი ხორციელდება ხანგრძლივი დროის მანძილზე.

ორმაგი ძებნის ალგორითმი ითვალისწინებს მასივის შუა ელემენტის მოძებნას, რომლის მნიშვნელობაც უნდა შედარდეს წინასწარ მოცემული ცვლადის მნიშვნელობას; თუ მათი მნიშვნელობები დაემთხვა, ეს ნიშნავს, რომ საჭირო ელემენტი მოიძებნა მასივში და ძებნის პროცესი წყდება. წინააღმდეგ შემთხვევაში, თუ ცვლადის მნიშვნელობა აღმოჩნდება ნაკლები მასივის შუა ელემენტის მნიშვნელობაზე, ძიება წარმოებს მასივის პირველ ნახევარში, ხოლო თუ ცვლადის მნიშვნელობა გადააჭარბებს მასივის შუა ელემენტის მნიშვნელობას, ძიება ხორციელდება საწყისი მასივის მეორე ნახევარში.

ამდენად, ძებნის პროცესის პირველივე ეტაპზე ადგილი აქვს მასივის განახევრებას, შემდეგ მიღებული ნახევრებიდან ერთ-ერთის იგნორირების გზით და მასივის მეორე ნახევრის შუაზე გაყოფით ძიება წარმოებს საწყისი მასივის მეოთხედში და ა.შ. პროცესი გრძელდება მანამდე, სანამ მასივის ერთ-ერთი ნახევრის შუა ელემენტის მნიშვნელობა არ დაემთხვევა მოცემული ცვლადის მნიშვნელობას, ან სანამ საჭირო ელემენტი არ მოიძებნება.

ამგვარად, მართალია, დიხოტომიის მეთოდი წინასწარ მოითხოვს მასივთა დახარისხებას, მაგრამ სანაცვლოდ ძებნის პროცესს აჩქარებს ყოველ ეტაპზე საწყისი და მიღებული მასივების განახევრების გზით.

<u>დავალება</u>

1. მოცემულია მასივი **A=(A_0, A_1, …, A_9)** და **k** ცვლადი. შეადგინეთ პროგრამა, რომელიც მასივის ელემენტებს ზრდადობის მიხედვით მიანიჭებს ორიდან დაწყებული ლუწ მნიშვნელობებს და წრფივი ძებნის მეთოდით მოძებნის **k** ცვლადის მნიშვნელობის ტოლი მნიშვნელობის მქონე ელემენტს მასივში (თუ კი ასეთი არსებობს).

2. მოცემულია მასივი **A=(A_0, A_1, …, A_{19})** და **k** ცვლადი. შეადგინეთ პროგრამა, რომელიც მოახდენს **A** მასივის დახარისხებას კლებადობის მიხედვით "ჩაძირვის" მეთოდით და დიხოტომიის მეთოდით მოძებნის **k** ცვლადის მნიშვნელობის მქონე ელემენტს მასივში.
3. მოცემულია მასივი **B=(B_0, B_1, …, B_{14})** და **p** ცვლადი. შეადგინეთ პროგრამა, რომელიც დაახარისხებს მასივის ელემენტებს ზრდადობით დახარისხების კომბინირებული მეთოდის გამოყენებით და ორმაგი ძებნის მეთოდით მოძებნის **p** ცვლადის მნიშვნელობის მქონე შესაბამის ელემენტს მასივში.
4. შეადგინეთ პროგრამა, რომელიც ათელემენტიან მთელრიცხვა მასივში წრფივი ძებნის მეთოდით განსაზღვრავს საჭირო მნიშვნელობის ელემენტებს და დათვლის მათ რაოდენობას.
5. შეადგინეთ პროგრამა, რომელიც თორმეტელემენტიან მთელრიცხვა მასივში წრფივი ძებნის მეთოდით განსაზღვრავს საჭირო მნიშვნელობის ელემენტებს და თუ მათი რაოდენობა აღმოჩნდა კენტი, გამოთვლის ლუწი მნიშვნელობის ელემენტების ნამრავლს.
6. შეადგინეთ პროგრამა, რომელიც ოცელემენტიან მთელრიცხვა მასივში წრფივი ძებნის მეთოდით განსაზღვრავს საჭირო მნიშვნელობის ელემენტებს და თუ მათი რაოდენობა აღმოჩნდა ლუწი, გამოთვლის კენტინდექსიანი ელემენტების კუბების ჯამს.
7. შეადგინეთ პროგრამა, რომელიც თხუთმეტელემენტიან მთელრიცხვა მასივში წრფივი ძებნის მეთოდით განსაზღვრავს საჭირო მნიშვნელობის ელემენტებს და თუ მათი რაოდენობა აღმოჩნდა ლუწი, გამოთვლის ლუწინდექსიანი ელემენტების ჯამიდან კვადრატული ფესვის მნიშვნელობას.

8. მოცემულია მასივი **X=(X_0, X_1, …, X_{18})** და **k** ცვლადი. შეადგინეთ პროგრამა, რომელიც დაახარისხებს მასივის ელემენტებს კლებადობით დახარისხების კომბინირებული მეთოდის გამოყენებით და დიხოტომიის მეთოდით მოძებნის **k** ცვლადის მნიშვნელობის მქონე შესაბამის ელემენტს მასივში.
9. მოცემულია მასივი **S=(S_0, S_1, …, S_{17})** და **a** ცვლადი. შეადგინეთ პროგრამა, რომელიც დაახარისხებს მასივის ელემენტებს ზრდადობით "ჩაძირვის" მეთოდის გამოყენებით და ორმაგი ძებნის მეთოდით მოძებნის **a** ცვლადის მნიშვნელობის მქონე შესაბამის ელემენტს მასივში.
10. შეადგინეთ პროგრამა, რომელიც ცამეტელემენტიან მთელრიცხვა მასივში წრფივი ძებნის მეთოდით განსაზღვრავს საჭირო მნიშვნელობის ელემენტებს და გამოთვლის მოძებნილი ელემენტების ინდექსების ნამრავლს.

პროგრამული კოდი და მიღებული შედეგები:

---------- (ქულა)

---------- (ხელმოწერა)

ლაბორატორიული სამუშაო №8
ტიპიური ამოცანები მატრიცებზე

დაპროგრამების ენა **C++**-ში მასივებს შეიძლება გააჩნდეს რამდენიმე ინდექსი. მრავალგანზომილებიანი მასივების ტიპიური წარმომადგენელია ერთგვარ მნიშვნელობათა ცხრილები, რომლებიც ინფორმაციას მოიცავენ სტრიქონებსა და სვეტებში.

იმისათვის, რომ ამ ცხრილებში განისაზღვროს ესათუის ელემენტი, საჭიროა მისი ორი ინდექსის მითითება. პირველი მათგანი სტრიქონის ნომერია, ხოლო მეორე იმ სვეტის ნომერი, რომელთა გადაკვეთაზეც იმყოფება ელემენტი.

იმ მასივებს ან ცხრილებს, რომელთა ცალკეულ ელემენტზე მისამართად საჭიროა ორი ინდექსის გამოყენება, ორგანზომილებიანი მასივები ანუ ***მატრიცები*** ეწოდებათ. აღსანიშნავია, რომ მრავალგანზომილებიან მასივებში ელემენტებს შეიძლება გააჩნდეს ორზე მეტი ინდექსიც.

a მასივის ყოველი ელემენტი განისაზღვრება მისი სახელით და ჩაიწერება შემდეგი სახით: ***a[i][j]***, სადაც ***a*** - მასივის იდენტიფიკატორია, ხოლო ***i*** და ***j*** - ინდექსები, რომლებიც ერთმნიშვნელოვნად განსაზღვრავენ ***a*** მასივის ყოველ ელემენტს. უნდა აღინიშნოს, რომ მატრიცის რიგით პირველ ელემენტს, მსგავსად ვექტორებისა, გააჩნია ნულოვანი ინდექსები.

მრავალგანზომილებიანი მასივების ელემენტები ისევე იღებენ საწყის მნიშვნელობებს, როგორც ერთგანზომილებიანი მასივის ელემენტები მათი გაცხადების დროს.

მაგალითი 8.1. მოცემულია ათი სტუდენტისგან შემდგარი ჯგუფი, რომელთაგან თვითოეული სტუდენტი აბარებს ოთხ გამოცდას. მათი შეფასება წარმოებს ასქულიანი სისტემით. შევადგინოთ პროგრამა, რომელიც სტუდენტებს ანიჭებს ყოველ გამოცდაში კონკრეტულ ქულებს, ითვლის თვითოეული სტუ-

დენტის საშუალო შედეგს, ჯგუფში პოულობს საუკეთესო და ყველაზე ცუდ შედეგებს და მათ მნიშვნელობებს.

ამოხსნა:

```
//ჯგუფური მაჩვენებლების ამოცანა
#include <iostream>
#include <iomanip>
main( )
{
const int students=10;
const int exams=4;
int array[students][exams];
int max, min, k, p, m, n;
float average[students];
for(int i=0; i<students; i++)
{
for(int j=0; j<exams; j++)
{
cout<<"array["<<i<<"]["<<j<<"]=";
cin>>array[i][j];
}}
cout<<endl<<"Current array:"<<endl;
for(int i=0; i<students; i++)
{
for(int j=0; j<exams; j++)
cout<<setw(5)<<array[i][j];
cout<<endl;
}
max=array[0][0];
min=array[0][0];
for(int i=0; i<students; i++)
{
average[i]=0;
for(int j=0; j<exams; j++)
{
average[i]+=array[i][j];
if(max<=array[i][j])
```

```
{
max=array[i][j];
k=i;
p=j;
}
if(min>=array[i][j])
{
min=array[i][j];
m=i;
n=j;
}}
cout<<"average["<<i<<"]="<<average[i]/exams<<endl;
}
cout<<"max="<<max<<endl;
cout<<"Maximum's indices are
array["<<k<<"]["<<p<<"]"<<
endl;
cout<<"min="<<min<<endl;
cout<<"Minimum's indices are
array["<<m<<"]["<<n<<"]"<<
endl;
return 0;
}
```

მაგალითი 8.2. მოცემულია კვადრატული მატრიცა ***a*[4x4]**. შევადგინოთ პროგრამა, რომელიც მოახდენს მატრიცის ტრანსპონირებას და დაბეჭდავს საწყის და მიღებულ მატრიცებს.

ამოხსნა:

```
//მატრიცის ტრანსპონირება
#include <iostream>
#include <iomanip>
main( )
{
const int rows=4;
const int columns=4;
int a[rows][columns], temp;
```

```
cout<<"Input array:"<<endl;
for(int i=0; i<rows; i++)
{

for(int j=0; j<columns; j++)
{
cout<<"a["<<i<<"]["<<j<<"]=";
cin>>a[i][j];}}
cout<<endl<<"Current array:"<<endl;
for(int i=0; i<rows; i++)
{
for(int j=0; j<columns; j++)
cout<<setw(5)<<a[i][j];
cout<<endl;}
for(int i=0; i<rows; i++)
{
for(int j=i+1; j<columns; j++)
{
temp=a[i][j];
a[i][j]=a[j][i];
a[j][i]=temp;}}
cout<<endl<<"Transposed array:"<<endl;
for(int i=0; i<rows; i++)
{
for(int j=0; j<columns; j++)
cout<<setw(5)<<a[i][j];
cout<<endl;}
return 0;
}
```

დავალება

1. მოცემულია მატრიცა **A[5x5]**. შეადგინეთ პროგრამა, რომელიც მატრიცის უარყოფითი მნიშვნელობის მქონე ელემენტებს შეცვლის ერთით და გამოთვლის მიღებული მატრიცის ელემენტების საშუალო არითმეტიკულს.

2. მოცემულია კვადრატული მატრიცა **B[4x4]**. შეადგინეთ პროგრამა, რომელიც დაბეჭდავს მატრიცის მოდულით უდიდესი მნიშვნელობის მქონე ელემენტს და ამ ელემენტის შემცველ სვეტს.
3. მოცემულია მატრიცა **C[2x4]**. შეადგინეთ პროგრამა, რომელიც გამოთვლის მატრიცის დადებითი ელემენტების ჯამს, უარყოფითი მნიშვნელობის მქონე ელემენტების ნამრავლს და მოძებნის მათ შორის უმცირეს შედეგს.
4. მოცემულია კვადრატული მატრიცა **m[3x3]**. შეადგინეთ პროგრამა, რომელიც ადგილებს შეუცვლის იმ სვეტსა და სტრიქონს, რომელთა გადაკვეთაზეც იმყოფება მატრიცის უდიდესი მნიშვნელობის მქონე ელემენტი.
5. მოცემულია კვადრატული მატრიცა **N[4x4]**. შეადგინეთ პროგრამა, რომელიც გამოთვლის მატრიცის ელემენტების ჯამებს სვეტების მიხედვით და დაბეჭდავს იმ სვეტის ელემენტებს, რომელთა ჯამიც მაქსიმალურია.
6. მოცემულია კვადრატული მატრიცა **B[3x3]**. შეადგინეთ პროგრამა, რომელიც გამოთვლის მატრიცის ელემენტების ნამრავლს სტრიქონების მიხედვით და დაბეჭდავს იმ სტრიქონის ელემენტებს, რომელთა ნამრავლიც მინიმალურია.
7. მოცემულია კვადრატული მატრიცა **C[3x3]**. შეადგინეთ პროგრამა, რომელიც მოახდენს მატრიცის ტრანსპონირებას და დაბეჭდავს $\mathbf{C}^t$ მატრიცას.
8. მოცემულია მატრიცა **B[3x4]**. შეადგინეთ პროგრამა, რომელიც ადგილებს შეუცვლის მატრიცის მეორე სვეტსა და მოდულით უმცირესი მნიშვნელობის მქონე ელემენტის შემცველ სვეტს.
9. მოცემულია მატრიცა **M[3x5]**. შეადგინეთ პროგრამა, რომელიც გამოთვლის მატრიცის კენტინდექსებიანი ელემენტების

საშუალო არითმეტიკულს და დანარჩენი ელემენტების ნამრავლს.

10. მოცემულია კვადრატული მატრიცა ***a*[4x4]**. შეადგინეთ პროგრამა, რომელიც გამოთვლის მატრიცის ლუწი მნიშვნელობის მქონე ელემენტების კუბების ჯამს და დანარჩენი ელემენტების ნახევარჯამს.

პროგრამული კოდი და მიღებული შედეგები:

---------- (ქულა)

---------- (ხელმოწერა)

ლაბორატორიული სამუშაო №9
კომბინირებული ამოცანები ვექტორებსა და მატრიცებზე

როგორც ცნობილია, $x = (x_1, x_2, \ldots x_m)$ ვექტორ-სტრიქონის ნამრავლი ***A[mxn]*** მატრიცაზე ეწოდება ***n*** განზომილების ისეთ $c = (c_1, c_2, \ldots c_n)$ ვექტორ-სტრიქონს, რომლის ელემენტები წარმოადგენს ***x*** ვექტორ-სტრიქონის ელემენტების ***A*** მატრიცის შესაბამის სვეტებში განლაგებულ ელემენტებზე ნამრავლთა ჯამებს.

ვექტორ-სტრიქონის მატრიცაზე გამრავლება შესაძლებელია იმ შემთხვევაში, როცა მატრიცის სტრიქონების რაოდენობა ვექტორის კომპონენტების რიცხვის ტოლია.

მაგალითი 9.1. მოცემულია ვექტორი $x = (x_0, x_1, x_2)$ და მატრიცა: $a = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \\ a_{20} & a_{21} \end{bmatrix}$. შევადგინოთ პროგრამა, რომელიც გამოთვლის ***x*** ვექტორის ***a*** მატრიცაზე ნამრავლს.

ამოხსნა: მათემატიკურად ***a*** მატრიცის ***x*** ვექტორზე ნამრავლი მიიღებს შემდეგ სახეს:

$$c = (x_0 a_{00} + x_1 a_{10} + x_2 a_{20},\ x_0 a_{01} + x_1 a_{11} + x_2 a_{21})$$

```
//მატრიცის ნამრავლი ვექტორ-სტრიქონზე
#include <iostream>
#include <iomanip>
main( )
{
const int rows=3;
const int columns=2;
float x[rows], a[rows][columns], c[columns];
```

```
cout<<"Input matrix a:"<<endl;
for(int i=0; i<rows; i++){
for(int j=0; j<columns; j++)
{
cout<<"a["<<i<<"]["<<j<<"]=";
cin>>a[i][j];}}
cout<<endl<<"Input vector x:"<<endl;
for(int i=0; i<rows; i++)
{
cout<<"x["<<i<<"]=";
cin>>x[i];
}
for(int j=0; j<columns; j++)
{
c[j]=0;
for(int i=0; i<rows; i++)
c[j]+=x[i]*a[i][j];
}
cout<<endl<<"Current matrix a:"<<endl;
for(int i=0; i<rows; i++)
{
for(int j=0; j<columns; j++)
cout<<setw(5)<<a[i][j];
cout<<endl; }
cout<<"Current vector x:"<<endl;
for(int i=0; i<rows; i++)
cout<<setw(5)<<x[i];
cout<<endl<<"Result:"<<endl;
for(int j=0; j<columns; j++)
cout<<setw(5)<<c[j];
cout<<endl;
return 0;}
```

A*[*m*x*n*]** მატრიცის ნამრავლი ***n-განზომილებიან ვექტორ-სვეტზე ეწოდება შემდეგი სახის ***m***-განზომილებიან ვექტორ-სვეტს:

$$\begin{bmatrix} a_{11} & a_{12}...a_{1n} \\ a_{21} & a_{22}...a_{2n} \\ ..... & ..... \\ a_{m1} & a_{m2}...a_{mn} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ . \\ u_n \end{bmatrix} = \begin{bmatrix} a_{11}u_1 + a_{12}u_2 + ... + a_{1n}u_n \\ a_{21}u_1 + a_{22}u_2 + ... + a_{2n}u_n \\ ..... \\ a_{m1}u_1 + a_{m2}u_2 + ... + a_{mn}u_n \end{bmatrix}$$

მატრიცის ვექტორ-სვეტზე გამრავლება შესაძლებელია იმ შემთხვევაში, როცა მატრიცის სვეტების რაოდენობა ვექტორის კომპონენტების რიცხვის ტოლია.

მაგალითი 9.2. მოცემულია მატრიცა ***a*[2x3]** და ვექტორ-სვეტი:

$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$ შევადგინოთ პროგრამა, რომელიც გამოთვლის ***a*** მატრიცის ***x*** ვექტორ-სვეტზე ნამრავლს.

ამოხსნა:

```
//მატრიცის ნამრავლი ვექტორ-სვეტზე
#include <iostream>
#include <iomanip>
main( )
{
const int rows=2;
const int columns=3;
float x[columns], a[rows][columns], c[rows];
cout<<"Input matrix  a:"<<endl;
for(int i=0; i<rows; i++){

for(int j=0; j<columns; j++)
{
cout<<"a["<<i<<"]["<<j<<"]=";
cin>>a[i][j];
}}
cout<<endl<<"Input vector  x:"<<endl;
for(int j=0; j<columns; j++)
```

```
{
cout<<"x["<<j<<"]=";
cin>>x[j];
}
for(int i=0; i<rows; i++)
{
c[i]=0;
for(int j=0; j<columns; j++)
c[i]+=a[i][j]*x[j];
}
cout<<endl<<"Current matrix  a:"<<endl;
for(int i=0; i<rows; i++)
{
for(int j=0; j<columns; j++)
cout<<setw(5)<<a[i][j];
cout<<endl; }
cout<<"Current vector  x:"<<endl;
for(int j=0; j<columns; j++)
cout<<setw(5)<<x[j];
cout<<endl<<"Result:"<<endl;
for(int i=0; i<rows; i++)
cout<<setw(5)<<c[i]<<endl;
return 0;}
```

<u>დავალება</u>

1. მოცემულია თხუთმეტი ელემენტისგან შემდგარი ორი ვექტორი: $x = (x_0, x_1, \ldots x_{14})$ და $y = (y_0, y_1, \ldots y_{14})$. შეადგინეთ პროგრამა, რომელიც გამოთვლის ამ ვექტორთა ნამრავლს.
2. მოცემულია **p** რიცხვი და მატრიცა **B[3x4]**. შეადგინეთ პროგრამა, რომელიც გამოთვლის **B** მატრიცის **p** რიცხვზე ნამრავლს.
3. მოცემულია ორი მატრიცა: **A[4x5]** და **B[5x3]**. შეადგინეთ პროგრამა, რომელიც გამოთვლის **A** და **B** მატრიცების ნამრავლს.

4. მოცემულია $x=(x_0, x_1, x_2, x_3)$ ვექტორ-სტრიქონი და მატრიცა **B[4x4]**. შეადგინეთ პროგრამა, რომელიც გამოთვლის ***x*** ვექტორ-სტრიქონის ნამრავლს **B** მატრიცაზე.

5. მოცემულია U= $\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{bmatrix}$ ვექტორ-სვეტი და მატრიცა **C[3x4]**. შეადგინეთ პროგრამა, რომელიც გამოთვლის **C** მატრიცის **U** ვექტორ-სვეტზე ნამრავლს.

6. მოცემულია კვადრატული მატრიცა **B[3x3]**. შეადგინეთ პროგრამა, რომელიც გამოთვლის მატრიცის ელემენტების კუბების ჯამს გარდა იმ ელემენტებისა, რომლებიც განლაგებულია მატრიცის მცირე დიაგონალზე.

7. მოცემულია კვადრატული მატრიცა **A[4x4]**. შეადგინეთ პროგრამა, რომელიც გამოთვლის მატრიცის მოდულით უდიდესი და მოდულით უმცირესი ელემენტების ჯამს და მიღებულ შედეგებს შეადარებს მატრიცის მთავარ დიაგონალზე მოთავსებული ელემენტების მოდულების ჯამს.

8. მოცემულია კვადრატული მატრიცა **M[4x4]**. შეადგინეთ პროგრამა, რომელიც გამოთვლის მატრიცის მცირე დიაგონალის ზემოთ მოთავსებული ელემენტების კვადრატების ჯამს და დანარჩენი ელემენტების საშუალო არითმეტიკულს.

9. მოცემულია კვადრატული მატრიცა **N[5x5]**. შეადგინეთ პროგრამა, რომელიც მოძებნის მატრიცის მთავარ დიაგონალზე მოთავსებულ უდიდეს და მცირე დიაგონალზე მოთავსებულ უმცირეს ელემენტებს.

10. მოცემულია ორი მატრიცა: ***a*[2x3]** და ***b*[2x3]**. შეადგინეთ პროგრამა, რომელიც გამოთვლის ამ მატრიცების არაუარყოფითი ელემენტების ჯამებს და დაბეჭდავს მათ შორის უდიდესს.

პროგრამული კოდი და მიღებული შედეგები:

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

(ქულა) (ხელმოწერა)

ლაბორატორიული სამუშაო №10
მომხმარებელთა მიერ შექმნილი ფუნქციები

ფუნქცია წარმოადგენს ბრძანებათა ჯგუფს, რომელსაც აქვს კონკრეტული სახელი, ანუ იდენტიფიკატორი, ხოლო ამ ჯგუფში მითითებული ბრძანებების შესრულება შესაძლებელია მხოლოდ ფუნქციის გამოძახების შედეგად.

C++-ში განასხვავებენ მომხმარებლის მიერ შექმნილი ფუნქციების ორ ძირითად სახეს:

1. ფუნქციები, რომლებიც ასრულებს რა გამოთვლით ოპერაციებს, მიღებულ შედეგებს აბრუნებს პროგრამის ძირითად ნაწილში (როგორც წესი, ***main*** ფუნქციაში);
2. ფუნქციები, რომლებიც ასრულებს რა სხვადასხვა სახის მანიპულაციებს, მიღებულ შედეგებს არ აბრუნებს პროგრამის ძირითად ნაწილში, რადგან საჭიროების შემთხვევაში თავად შეუძლია მათი დაბეჭდვა.

ფუნქციის განსაზღვრა (***function definition***) ეწოდება მისი პროგრამული კოდის იმ ნაწილს, რომელშიც აღიწერება შესასრულებელი მოქმედებები. იგი მოიცავს სათაურსა და ფუნქციის ტანს. ზოგად შემთხვევაში ფუნქციის განსაზღვრას აქვს შემდეგი სახე:

ფუნქციის სათაური
{
ფუნქციის ტანი;
}

ფუნქციის სათაური (***function header***) მოიცავს:

- დასაბრუნებელი შედეგის ტიპს, ხოლო ფუნქციებისთვის, რომლებიც მიღებულ შედეგებს არ აბრუნებს, ამ შემთხვევაში მიეთითება ტიპი ***void***, რაც ნიშნავს ცარიელს, დაუკავებელს;

- ფუნქციის სახელს (იდენტიფიკატორს), ასევე ერთმანეთისგან მძიმეებით გამოყოფილი პარამეტრებისა და მათი ტიპების სიას, რაც თავსდება მრგვალ ფრჩხილებში. ამასთან, ყოველი პარამეტრის წინ იწერება მისი ტიპი (უპარამეტრებო ფუნქციის შემთხვევაში მრგვალ ფრჩხილებში არსებული სივრცე ცარიელი რჩება).

ფუნქციის ტანი (***function body***) მოიცავს კოდს, რომელიც სრულდება ფუნქციის გამოძახების დროს. ეს არის ერთი ან რამდენიმე ბრძანების შემცველი პროგრამული კოდის ფრაგმენტი, რომლის საზღვრები ღია და დახურულ ფიგურულ ფრჩხილებს შორის თავსდება.

ფუნქციის განსაზღვრისას აღწერილი ცვლადები განიხილება როგორც ლოკალური ცვლადები, ანუ მათი განსაზღვრის არე თავად ფუნქციაა, რაც ნიშნავს, რომ ყველა ეს ცვლადი ცნობილია მხოლოდ იმ ფუნქციისთვის, რომელშიც მოხდა მათი აღწერა.

ფუნქციის არგუმენტების სიაში არსებული მონაცემების ტიპები და რაოდენობა სავსებით უნდა შეესაბამებოდეს ფუნქციის პარამეტრების სიის მონაცემთა ტიპებსა და რაოდენობას. ფუნქციის გამოძახება უმეტეს შემთხვევაში ხორციელდება ***main*** ფუნქციაში, სადაც წინასწარ უნდა იქნეს აღწერილი ფუნქციის პარამეტრების შესაბამისი არგუმენტები და ის ცვლადები, რომელთაც ენიჭება ფუნქციის მიერ დაბრუნებული შედეგები (ცხადია, ეს იმ შემთხვევაში, თუ კი ფუნქცია ახდენს მიღებული რეზულტატების დაბრუნებას).

იმ ფუნქციათა ტანი, რომლებიც ***main*** ფუნქციაში აბრუნებს გამოთვლის შედეგებს, აუცილებლად უნდა შეიცავდეს ერთს მაინც ***return*** ოპერატორს, რომელიც გვაძლევს მიღებულ რეზულტატს.

მაგალითი 10.1. შევადგინოთ პროგრამა, რომელიც გამოთვლის $s = 1 + x + \frac{x^2}{2} + \frac{x^3}{3} + \ldots + \frac{x^n}{n}$ უსასრულო მწკრივის პირველი n წევრის ჯამს, სადაც n=10 (პროგრამა წარმოვადგინოთ ორ ვარიანტად).

ამოხსნა (I ვარიანტი)

```
// უსასრულო მწკრივის პირველი n წევრის ჯამის გამო-
   თვლა
#include <iostream>
double function(float x, int n)  // function header
{
float p=1, s=1;
for(int i=1; i<n; i++)
{
p*=x;    // function body
s+=p/i;
}
return s;
}
main( )
{
double k;
float y;
int m;
cout<<"x=";
cin>>y;
cout<<"n=";
cin>>m;
k=function(y,m);   // function call
cout<<"\n  result="<<k<<endl;
return 0;
}
```

ამოხსნა (II ვარიანტი)

```
//უსასრულო მწკრივის პირველი n წევრის ჯამის გამოთვლა
```

```
#include <iostream>
void function(float x, int n)   // function header
{
float p=1, s=1;                 // function body
for(int i=1; i<n; i++)
{
p*=x;
s+=p/i;
}
cout<<"\n result="<<s<<endl;
}
main( )
{
float y;
int m;
cout<<"x=";
cin>>y;
cout<<"n=";
cin>>m;
function(y,m);    // function call
return 0;}
```

<u>დავალება</u>

1. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ფუნქციის საშუალებით y ფუნქციის ყველა მნიშვნელობას გამოთვლის შემდეგი გამოსახულების საფუძველზე:

$$y = 0,3x^3 + 2(\cos x + 1)x^2 - 1,2x - 3,$$

სადაც x არგუმენტი [-3;3] დიაპაზონში იცვლება h=0,4 ბიჯით (პროგრამა წარმოადგინეთ ორ ვარიანტად ფუნქციის პროტოტიპის გამოყენებით).

2. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ფუნქციების საშუალებით გამოთვლის შემდეგი გამოსახულების მნიშვნელობას:

$$\max\{C-D,\ \ C^2 * D\},$$

$$\text{თუ}\quad C=\ln(|y|+p^3),\quad D=\frac{\sqrt{\sin^2(y+ap)}}{\sqrt{e^{|p|+a^2}}},$$

$$y=|p|x^5+\sqrt[5]{a},\quad \text{სადაც}\quad a=2{,}5;\ \ x=0{,}691;\ \ p=-7{,}98$$

(პროგრამა ორ ვარიანტად წარმოადგინეთ ფუნქციის პროტოტიპის გამოყენებით).

3. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ფუნქციის საშუალებით y ფუნქციის მნიშვნელობას გამოთვლის შემდეგი გამოსახულების საფუძველზე:

$$z=x*B*tgx;\quad y=\begin{cases}\ln z, & \text{თუ}\ \ z>0\\ \sin z^2, & \text{თუ}\ \ z\le 0,\end{cases}$$

სადაც x=4,5; B=15 (პროგრამა ორ ვარიანტად წარმოადგინეთ ფუნქციის პროტოტიპის გამოყენებით).

4. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ფუნქციის გამოყენებით შემთხვევითი რიცხვების გენერაციის გზით გამოიმუშავებს მთელი ტიპის სამ შემთხვევით რიცხვს ერთიდან ორმოცამდე დიაპაზონში და მოძებნის მათ შორის მინიმალურ მნიშვნელობას.

5. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ფუნქციის გამოყენებით გამოთვლის $y=1+\frac{a}{1!}+\frac{a^2}{2!}+\frac{a^3}{3!}+\ldots+\frac{a^n}{n!}$ უსასრულო მწკრივის პირველი n წევრის ჯამს, სადაც n=15 (პროგრამა ორ ვარიანტად წარმოადგინეთ).

6. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ფუნქციის საშუალებით გამოთვლის ნებისმიერი ორი მთელი არაუარყოფითი a და b რიცხვის უმცირეს საერთო ჯერადს (პროგრამა წარმოადგინეთ ორ ვარიანტად ფუნქციის პროტოტიპის გამოყენებით).

7. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ფუნქციის საშუალებით ნებისმიერ სამ ცვლადს შორის განსაზღვრავს შუალედური მნიშვნელობის ცვლადს.
8. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ფუნქციის საშუალებით გამოთვლის ნებისმიერ სამ ცვლადს შორის უდიდესი და უმცირესი მნიშვნელობის ცვლადების ნამრავლს.
9. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ფუნქციის საშუალებით გამოთვლის $ax=b$ წრფივი განტოლების ფესვებს a და b კოეფიციენტების ნებისმიერი მნიშვნელობების დროს.
10. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ფუნქციის საშუალებით წარმოადგენს ერთიდან ათის ჩათვლით მთელი ტიპის რიცხვითი მონაცემების კვადრატების მნიშვნელობებს.

პროგრამული კოდი და მიღებული შედეგები:

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

ლაბორატორიული სამუშაო №11
რეკურსიული ფუნქციები

არის შემთხვევები, როდესაც პროგრამებში სასარგებლოა ისეთი ფუნქციების გამოყენება, რომელთაც შეუძლია საკუთარი თავის გამოძახება. აღნიშნული სახის ფუნქციებს, ***რეკურსიული*** ეწოდება.

რეკურსიული ამოცანა ცალკეულ ეტაპებად იყოფა. ამოცანის გადასაწყვეტად ხდება რეკურსიული ფუნქციის გამოძახება, რომელმაც "იცის", თუ როგორ ამოხსნას ამოცანის უმარტივესი ნაწილი, ე.წ. ***საბაზო ამოცანა***. თუ ფუნქციის გამოძახება ხდება მხოლოდ საბაზო ამოცანის გადაწყვეტის მიზნით, მაშინ იგი უბრალოდ აბრუნებს მიღებულ შედეგს, ხოლო უფრო რთული ამოცანის გადასაწყვეტად, იგი ამოცანას ორ ნაწილად ყოფს. ამასთან, ერთი ნაწილის ამოხსნა ფუნქციამ იცის, ხოლო მეორისა არა. რეკურსია რომ შესრულდეს, ამოცანის მეორე ნაწილი საწყისი ამოცანის მსგავსი უნდა იყოს, მაგრამ უფრო გამარტივებული. რადგან ეს ახალი ამოცანა საწყისი ამოცანის მსგავსია, ფუნქცია ახდენს საკუთარი თავის ახალი ასლის გამოძახებას იმ მიზნით, რომ მუშაობა დაიწყოს შედარებით მარტივ პრობლემაზე. ამ პროცესს ***რეკურსიული გამოძახება***, ანუ ***რეკურსიის ბიჯი*** ეწოდება და იგი აუცილებლად უნდა შეიცავდეს ***return*** ოპერატორს, ხოლო მის მიერ დაბრუნებული შედეგი საჭიროა გაერთიანდეს ამოცანის იმ ნაწილთან, რომლის ამოხსნაც ფუნქციას შეუძლია. შემდეგ კი ფორმირდება საბოლოო შედეგი და იგი უბრუნდება ფუნქციის გამოძახების საწყის ადგილს (ხშირად ***main*** ფუნქციას). რეკურსიის ბიჯმა შესაძლოა გამოიწვიოს არაერთი რეკურსიული გამოძახება, რადგან ფუნქცია ყოველ ახალ ქვეამოცანას ორ ნაწილად დაყოფს. რეკურსიის პროცესის დასრულებისათვის აუცილებელია საწყისი ამოცანა დაყვანილ იქნეს საბაზომდე.

მაგალითი 11.1. რეკურსიული გზით გამოვთვალოთ ფიბონაჩის რიცხვითი მიმდევრობის ნებისმიერი *n* წევრის მნიშვნელობა.

ამოხსნა

მათემატიკიდან ცნობილია, რომ ფიბონაჩის რიცხვითი მიმდევრობა 0-ით და 1-ით იწყება და ხასიათდება იმ თვისებით, რომ მიმდევრობის ყოველი მომდევნო წევრის მნიშვნელობა წინა ორი წევრის მნიშვნელობათა ჯამის ტოლია. ბუნებაში ფიბონაჩის რიცხვითი მიმდევრობა სპირალის ფორმას აღწერს და მათემატიკურად აქვს შემდეგი სახე:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, . . .

რეკურსიული გზით ფიბონაჩის რიცხვითი მიმდევრობა შეიძლება განისაზღვროს შემდეგნაირად:

Fibonacci(0)=0;

Fibonacci(1)=1;

Fibonacci(n)= Fibonacci(n-1)+ Fibonacci(n-2).

```
// ფიბონაჩის მიმდევრობის n წევრის მნიშვნელობის განსაზღვრა.
#include <iostream>
unsigned long Fibonacci(unsigned long);  // prototipe
main( )
{ unsigned long result, number;
cout<<"n=";
cin>>number;
result=Fibonacci(number);
cout<<"\n Result="<<result<<endl;
return 0;
}
unsigned long Fibonacci(unsigned long n)
{
if(n==0 || n==1)
return n;
```

```
else
return Fibonacci(n-1)+Fibonacci(n-2);
}
```

<u>დავალება</u>

1. შეადგინეთ პროგრამა, რომელიც რეკურსიული გზით გამოთვლის ნებისმიერი მთელი არაუარყოფითი რიცხვის ფაქტორიალის მნიშვნელობას (იგივე ამოცანა გადაწყვიტეთ იტერაციული ხერხით).
2. შეადგინეთ პროგრამა, რომელიც რეკურსიული გზით გამოთვლის ფიბონაჩის რიცხვითი მიმდევრობის პირველი თხუთმეტი წევრის მნიშვნელობას.
3. შეადგინეთ პროგრამა, რომელიც რეკურსიული გზით გამოთვლის *a*-დან *b*-მდე მთელი ტიპის დადებითი მნიშვნელობის მქონე რიცხვების კვადრატების ჯამს (იგივე ამოცანა გადაწყვიტეთ იტერაციული ხერხით).
4. შეადგინეთ პროგრამა, რომელიც რეკურსიული გზით გამოთვლის *m*-დან *n*-მდე მთელი ტიპის დადებითი მნიშვნელობის რიცხვების ნამრავლს (იგივე ამოცანა გადაწყვიტეთ იტერაციული ხერხით).
5. შეადგინეთ პროგრამა, რომელიც რეკურსიული გზით გამოთვლის a^x გამოსახულების მნიშვნელობას.
6. შეადგინეთ პროგრამა, რომელიც რეკურსიული გზით გამოთვლის *x*-დან *y*-მდე მთელი ტიპის დადებითი მნიშვნელობის მქონე რიცხვების ჯამს (იგივე ამოცანა გადაწყვიტეთ იტერაციული ხერხით).
7. შეადგინეთ პროგრამა, რომელიც რეკურსიული გზით გამოთვლის *k*-დან *p*-მდე მთელი ტიპის დადებითი მნიშვნელობის მქონე რიცხვების ნამრავლს (იგივე ამოცანა გადაწყვიტეთ იტერაციული ხერხით).

8. შეადგინეთ პროგრამა, რომელიც რეკურსიული გზით განსაზღვრავს ორი ნატურალური რიცხვის უდიდესი საერთო გამყოფის მნიშვნელობას.
9. შეადგინეთ პროგრამა, რომელიც რეკურსიული გზით გამოთვლის *a*-დან *b*-მდე მთელი ტიპის დადებითი მნიშვნელობის მქონე რიცხვების ჯამს, თუ *a* და *b* კენტი რიცხვებია, წინააღმდეგ შემთხვევაში გამოთვლის ამ რიცხვების ნამრავლს.
10. შეადგინეთ პროგრამა, რომელიც რეკურსიული გზით გამოთვლის ერთიდან ათამდე ნატურალური რიცხვების ფაქტორიალის მნიშვნელობებს (იგივე ამოცანა გადაწყვიტეთ იტერაციული ხერხით).

პროგრამული კოდი და მიღებული შედეგები:

(ქულა)

(ხელმოწერა)

ლაბორატორიული სამუშაო №12

inline ფუნქციები

ცნობილია, რომ ნებისმიერი ფუნქციის გამოძახება დაკავშირებულია დამატებით დროით დანახარჯებთან. ამ სიტუაციის თავიდან აცილების მიზნით **C++**-ში გამოიყენება ერთგვარი შესაძლებლობა ***inline*** ფუნქციების სახით. სპეციფიკაცია ***inline***, რომელიც თავსდება ფუნქციის მიერ დასაბრუნებელი შედეგის ტიპის მითითებამდე ფუნქციის განსაზღვრის დროს, კომპილატორს "ურჩევს" მოახდინოს ფუნქციის კოდის ასლის გენერირება პროგრამის საჭირო ადგილზე, ამით თავიდან აიცილოს ფუნქციის გამოძახება და მასთან დაკავშირებული დროითი დანახარჯები.

აღსანიშნავია, რომ ***inline*** სპეციფიკაციას მიზანშეწონილია მივმართოთ მხოლოდ მცირე ზომისა და ხშირად გამოყენებადი ფუნქციების შემთხვევაში, რადგან, მართალია იგი ამცირებს პროგრამის შესრულების დროს, მაგრამ სამაგიეროდ შეუძლია მნიშვნელოვნად გაზარდოს მისი ზომა. მხედველობაშია მისაღები ის ფაქტიც, რომ კომპილატორმა შეიძლება იგნორირება გაუკეთოს აღნიშნულ სპეციფიკაციას და, როგორც წესი, იგი ხშირად ასეც იქცევა, გარდა იმ შემთხვევისა, როდესაც საქმე გვაქვს მცირე ზომის ფუნქციებთან.

მაგალითი 12.1. ***inline*** სპეციფიკაციის მქონე ფუნქციის გამოყენებით შევადგინოთ პროგრამა, რომელიც გამოთვლის კუბის მოცულობას.

ამოხსნა

```
#include <iostream>
inline float Cube( float  S)
{
return S*S*S;
```

```
}
main ( )
{
float side, size;
cout<<"S=";
cin>>side;
size=Cube(side);
cout<<"\n Size="<<size;
cout<<endl;
return 0;
}
```

<u>დავალება</u>

1. შეადგინეთ პროგრამა, რომელიც ***inline*** სპეციფიკაციის მქონე ფუნქციის გამოყენებით გამოთვლის შემდეგი გამოსახულების მნიშვნელობას:

$$min\{B - D,\ |D|\},\quad \text{თუ}$$

$$B = \sin(c+1) + \frac{1}{\cos(cx+3)},\quad D = ln\left(|c| + c^3\right),$$

სადაც $c = y^2x^3 + (a+12)^2,\ x = 2{,}8;\ y = 7{,}66;\ a = 9{,}012.$

2. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ***inline*** სპეციფიკაციის მქონე ფუნქციის გამოყენებით გამოთვლის $ax + b = 0$ წრფივი განტოლების ფესვებს a და b კოეფიციენტების ნებისმიერი მნიშვნელობების დროს.
3. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ***inline*** სპეციფიკაციის მქონე ფუნქციის გამოყენებით გამოთვლის მართკუთხა პარალელეპიპედის მოცულობას.
4. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ***inline*** სპეციფიკაციის მქონე ფუნქციის გამოყენებით გამოთვლის $ax^2 + bx + c = 0$ კვადრატული განტოლების ფეს-

ვებს *a*, *b* და *c* კოეფიციენტების ნებისმიერი მნიშვნელობების დროს.

5. გამოთვალეთ $S = 1 + \frac{y}{1!} + \frac{y^2}{2!} + \ldots + \frac{y^m}{m!}$ შემდეგი მწკრივის პირველი 10 წევრის ჯამის მნიშვნელობა მომხმარებლის მიერ შექმნილი ***inline*** სპეციფიკაციის მქონე ფუნქციის გამოყენებით.
6. მოცემულია სამი ***x***, ***y***, და ***z*** მთელი ტიპის ცვლადი. გამოთვალეთ მათ შორის მაქსიმალური და მინიმალური მნიშვნელობის მქონე ცვლადების საშუალო არითმეტიკული (გამოიყენეთ მომხმარებლის მიერ შექმნილი ***inline*** სპეციფიკაციის მქონე ფუნქციები).
7. მოცემულია ***float*** ტიპის სამი ***a***, ***b***, და ***c*** ცვლადი. გამოთვალეთ მათ შორის მაქსიმალური და მინიმალური მნიშვნელობის მქონე ცვლადების ნამრავლი (გამოიყენეთ მომხმარებლის მიერ შექმნილი ***inline*** სპეციფიკაციის მქონე ფუნქციები).
8. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ***inline*** სპეციფიკაციის მქონე ფუნქციის გამოყენებით გამოთვლის ტრაპეციის ფართობს.
9. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ***inline*** სპეციფიკაციის მქონე ფუნქციის გამოყენებით ჰერონის ფორმულის საფუძველზე გამოთვლის სამკუთხედის ფართობს.
10. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ***inline*** სპეციფიკაციის მქონე ფუნქციის გამოყენებით გამოთვლის პირამიდის მოცულობას.

პროგრამული კოდი და მიღებული შედეგები:

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

ლაბორატორიული სამუშაო №13
დამისამართებით გადაცემული პარამეტრები. ფუნქციათა გადატვირთვა. ფუნქციის შაბლონი.

დაპროგრამების მრავალ ენაში არსებობს ფუნქციებზე მიმართვის ორი შესაძლებლობა:

1. ფუნქციის გამოძახება პარამეტრების მნიშვნელობების მიხედვით;
2. ფუნქციის გამოძახება დამისამართებით გადაცემული პარამეტრების მიხედვით.

როდესაც ამა თუ იმ ფუნქციის გამოძახებისას ადგილი აქვს არგუმენტის გადაცემას შესაბამისი პარამეტრის მნიშვნელობის მიხედვით, იქმნება არგუმენტის ასლი, რომლის ცვლილება, ბუნებრივია, გამოძახების ოპერატორში არ მოქმედებს არგუმენტის ორიგინალის მნიშვნელობაზე. ფუნქციის ასეთი გამოძახების ნაკლი კი ისაა, რომ თუ ფუნქციაში გადაიცემა მონაცემთა დიდი რაოდენობა, მაშინ მათი ასლების შექმნამ შესაძლოა გამოიწვიოს მნიშვნელოვანი დროითი დანახარჯები. ამიტომ, როდესაც აუცილებელია აღნიშნული სახის დანახარჯების თავიდან აცილება, ფუნქციაში საჭიროა პარამეტრები გადავცეთ დამისამართებით. ამ შემთხვევაში ფუნქციის გამოძახების ოპერატორი მას საშუალებას აძლევს პირდაპირი გზით მიმართოს ფუნქციაში გადაცემულ მონაცემებს და საჭიროების შემთხვევაში შეცვალოს კიდეც მათი ორიგინალების მნიშვნელობები.

ამგვარად, დამისამართებით გადაცემული პარამეტრი წარმოადგენს შესაბამისი არგუმენტის ფსევდონიმს. როდესაც ესა თუ ის პარამეტრი დამისამართებით გადაიცემა ფუნქციაში, მისი პროტოტიპისა და თავად ფუნქციის განსაზღვრის დროს დამისამართებით გადაცემული პარამეტრის ტიპის მითითების შემდეგ იწერება ამპერსენდის (**&**) სიმბოლო. ასე, მაგალითად:

int &count;

აღნიშნული ჩანაწერი მიუთითებს, რომ ცვლადი ***count*** წარმოადგენს დამისამართებას ***int*** (მთელი) ტიპის მონაცემზე. ფუნქციის გამოძახების შემთხვევაში კი საკმარისია შესაბამისი ცვლადის სახელის მითითება და იგი გადაიცემა დამისამართებით. ყოველივე ზემოთ თქმულის გათვალისწინებით, განვიხილოთ მაგალითი.

მაგალითი 13.1. შევადგინოთ პროგრამა, რომელიც გამოთვლის ნებისმიერი მთელი ტიპის მნიშვნელობის რიცხვის კვადრატს. ამასთან, პროგრამაში გამოვიყენოთ ორი ფუნქცია, რომელთაგან პირველ ფუნქციაში პარამეტრი გადაიცემა მნიშვნელობით, ხოლო მეორეში დამისამართებით.

ამოხსნა

```
#include <iostream>
void function1(int);       //prototipe
void function2(int &);   //prototipe
main( )
{
int a, b;
cout<<"a="; cin>>a;
cout<<"b="; cin>>b;
function1(a);
function2(b);
cout<<"\n  Real situation:"<<endl;
cout<<"a="<<a<<endl;
cout<<"b="<<b<<endl;
return 0;
}
void function1(int x)
{
x=x*x;
cout<<"\n  x="<<x<<endl;
}
```

```
void function2(int &y)
{
y=y*y;
cout<<"\n  y="<<y<<endl;
}
```

თუ დავუშვებთ, რომ *a*=5 და *b*=7, მაშინ პროგრამის შესრულებაზე გაშვების შედეგად მივიღებთ:

```
a=5
b=7
x=25
y=49
Real situation:
a=5
b=49
```

C++ საშუალებას გვაძლევს ამა თუ იმ პროგრამაში განვსაზღვროთ ერთი და იმავე სახელის მქონე რამდენიმე ფუნქცია, ოღონდ მხოლოდ იმ შემთხვევაში, თუკი ფუნქციებს ექნება პარამეტრების განსხვავებული ნაკრები (სულ მცირე, პარამეტრების განსხვავებული ტიპები). აღნიშნულ თვისებას, ***ფუნქციათა გადატვირთვა*** ეწოდება. იმისათვის, რომ კომპილატორმა განსაზღვროს, გადატვირთვის შემთხვევაში რომელი ფუნქცია გამოიძახოს, იგი იძულებულია გაანალიზოს არგუმენტების ტიპები, რაოდენობა და მათი მიმდევრობა.

მაგალითი 13.2. შევადგინოთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ფუნქციების გამოყენებით გამოთვლის კვადრატის ფართობს. ამასთან, გამოვიყენოთ ფუნქციათა გადატვირთვის შესაძლებლობა.

ამოხსნა

```
#include <iostream>
void Square(int x)
{
cout<<"Result1="<<x*x<<endl;
```

```
}
void Square(double y)
{
cout<<"Result2="<<y*y<<endl;
}
main( )
{
int a;
double b;
cout<<"x="; cin>>a;
cout<<"y="; cin>>b;
Square(a);
Square(b);
return 0;
}
```

ფუნქციათა გადატვირთვა, როგორც წესი, გამოიყენება მსგავსი ოპერაციების შესასრულებლად სხვადასხვა ტიპის მონაცემებზე, ხოლო, თუ ოპერაციები მონაცემთა ყოველი ტიპისათვის იდენტურია, ისინი შესაძლებელია გაცილებით კომპაქტურად განვახორციელოთ ფუნქციათა შაბლონების გამოყენებით. განვიხილოთ მისი არსი: პროგრამისტი განსაზღვრავს ფუნქციის ერთადერთ შაბლონს. ეყრდნობა რა ფუნქციის გამოძახების დროს გამოყენებულ არგუმენტთა ტიპებს, C++ ავტომატურად ახორციელებს სხვადასხვა ფუნქციის გენერირებას მონაცემთა თითოეული ტიპის დამუშავების მიზნით. ამგვარად, ერთადერთი შაბლონის აღწერა იწვევს გადაწყვეტილებათა მთელი ჯგუფის განსაზღვრას.

ფუნქციის შაბლონის განსაზღვრა იწყება კოდური სიტყვით ***template*** (შაბლონი), რომელსაც მოყვება ფუნქციის პარამეტრების ფორმალური ტიპების სია. ეს უკანასკნელი თავსდება "< >" ფრჩხილებში. ყოველ ფორმალურ ტიპს წინ უსწრებს კოდური სიტყვა ***class***. პარამეტრების ფორმალური ტიპები წარმოადგენს

მონაცემთა შედგენილ ან მომხმარებლის მიერ განსაზღვრულ ტიპებს. ისინი გამოიყენება ფუნქციის არგუმენტებისა და დასაბრუნებელი შედეგის ტიპების მოსაცემად, ასევე ფუნქციის ტანში საჭირო ცვლადების აღწერის მიზნით. შაბლონის შემდეგ, როგორც წესი, განისაზღვრება თავად ფუნქცია. შაბლონის აღწერისას გამოყენებული ყოველი ფორმალური პარამეტრი ერთხელ მაინც უნდა იქნეს წარმოდგენილი ფუნქციის პარამეტრების სიაში. ამასთან, ფორმალური პარამეტრის ყოველი სახელი უნდა იყოს უნიკალური (სხვებისგან განსხვავებული). შეგვიძლია დავასკვნათ, რომ შაბლონები რეალურად ასრულებს ფუნქციის კოდის გენერირების საშუალებათა როლს.

მაგალითი 13.3. შევადგინოთ პროგრამა, რომელიც ფუნქციის შაბლონის გამოყენებით გამოთვლის ნებისმიერი ტიპის სამი ცვლადის კვადრატების ჯამს.

ამოხსნა

```
#include  <iostream>
#include  <math>
template <class T>
T amount (T value1, T value2, T value3)
{
T sum=pow(value1,2)+ pow(value2,2)+pow(value3,2);
return sum;
}
main( )
{
int a,b,c,d;
float m,n,k,p;
double x,y,z,f;
cout<<" Enter numbers of type int:";
cin>>a>>b>>c;
cout<<" Enter numbers of type float:";
cin>>m>>n>>k;
cout<<" Enter numbers of type double:";
```

```
cin>>x>>y>>z;
d=amount(a,b,c);
p=amount(m,n,k);
f=amount(x,y,z);
cout<<"\n Result1="<<d<<endl;
cout<<" Result2="<<p<<endl;
cout<<" Result3="<<f<<endl;
return 0;}
```

<u>დავალება</u>

1. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი უპარამეტრებო ფუნქციის გამოყენებით გამოთვლის $y = e^{x^2} - e^{-x}$ ფუნქციის დადებითი მნიშვნელობების ნამრავლს, სადაც x არგუმენტი [-0,6;1,7] დიაპაზონში იცვლება h=0.25 ბიჯით.
2. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ფუნქციის გამოყენებით გამოთვლის სამკუთხედის ფართობს ჰერონის ფორმულის საფუძველზე. ამასთან, ფუნქციაში ნებისმიერი ორი პარამეტრი გადაეცით დამისამართებით, ხოლო დარჩენილი პარამეტრი მნიშვნელობით.
3. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ფუნქციის გამოყენებით გამოთვლის $ax + b = 0$ წრფივი განტოლების ფესვებს a და b კოეფიციენტების ნებისმიერი მნიშვნელობების დროს. ამასთან, a პარამეტრი ფუნქციაში გადაეცით მნიშვნელობით, ხოლო b პარამეტრი დამისამართებით.
4. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ფუნქციის გამოყენებით გამოთვლის შემდეგი გამოსახულების მნიშვნელობას:

$$z = \sin(R\pi - 5);\ \ R = \frac{x^2 + 25}{(x+5)},\ \text{ სადაც } x = 2{,}19.$$

ამასთან, ფუნქცია გამოიძახეთ ჯერ არგუმენტების გარეშე, ხოლო შემდეგ – ერთი არგუმენტით.

5. შეადგინეთ პროგრამა, რომელიც მომხმარებლის მიერ შექმნილი ფუნქციის გამოყენებით გამოთვლის შემდეგი გამოსახულების მნიშვნელობას:

$$\min\{A,\ D,\ C\},\quad \text{თუ}$$

$$A=\sin\left(\frac{z+b^2z}{|b|+1}\right),\ D=\sqrt[5]{z^3+\ln b},\ C=e^{b^2}-2^{z+b},$$

$$\text{სადაც}\quad b=1{,}57;\ z=12{,}5.$$

ამასთან, ფუნქცია გამოიძახეთ ჯერ არგუმენტების გარეშე, ხოლო შემდეგ – ერთი და ორი არგუმენტით.

6. შეადგინეთ პროგრამა, რომელიც ფუნქციათა გადატვირთვის გამოყენებით გამოთვლის ნებისმიერი სამი რიცხვის კუბების ჯამიდან კვადრატული ფესვის მნიშვნელობას.
7. შეადგინეთ პროგრამა, რომელიც ფუნქციათა გადატვირთვის გამოყენებით გამოთვლის შემთხვევითი რიცხვების გენერაციის გზით ერთიდან ასამდე დიაპაზონში გენერირებული ნებისმიერი ოთხი მთელი რიცხვის საშუალო არითმეტიკულს.
8. შეადგინეთ პროგრამა, რომელიც ფუნქციის შაბლონის გამოყენებით გამოთვლის ნებისმირი სამი რიცხვის ნამრავლიდან კვადრატული ფესვის მნიშვნელობას. ფუნქცია გამოიძახეთ ***int***, ***long*** და ***double*** ტიპის მონაცემებისთვის.
9. შეადგინეთ პროგრამა, რომელიც ფუნქციის შაბლონის გამოყენებით გამოთვლის მართკუთხედის პერიმეტრს ***float*** და ***double*** ტიპის მონაცემებისთვის.
10. შეადგინეთ პროგრამა, რომელიც ფუნქციის შაბლონის გამოყენებით განსაზღვრავს მინიმალური მნიშვნელობის ცვლადს ნებისმიერ სამ ცვლადს შორის. ფუნქცია გამოიძახეთ ***int***, ***float*** და ***char*** ტიპის მონაცემებისთვის.

პროგრამული კოდი და მიღებული შედეგები:

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

ლაბორატორიული სამუშაო №14
თანაბარი ალბათობით განაწილებული შემთხვევითი რიცხვების გენერირება მასივებში

დაპროგრამების ნებისმიერი ენის კომპილატორს გააჩნია შემთხვევითი რიცხვების გენერატორი, რომელიც სასურველი დიაპაზონიდან თანაბარი ალბათობით ახდენს შემთხვევითი რიცხვების გამომუშავებას (გენერირებას).

ასეთ გენერატორს C++ დაპროგრამების ენაში წარმოადგენს rand() ფუნქცია, რომელიც გამოიმუშავებს თანაბარი ალბათობით განაწილებულ ფსევდო-შემთხვევით რიცხვებს, ხოლო srand() ფუნქცია კვაზი-შემთხვევით რიცხვებს. ორივე ფუნქცია მიეკუთვნება stdlib ბიბლიოთეკას. ამიტომ აღნიშნული ფუნქციების გამოყენების დროს აუცილებელია პროგრამაში ჩავრთოთ დირექტივა #include <stdlib>.

მაგალითი 14.1. შევადგინოთ პროგრამა, რომელიც:

- int a[10] მასივის ელემენტებს შემთხვევითი რიცხვების გენერირების გზით [1;80] დიაპაზონიდან მიანიჭებს მთელი ტიპის კვაზი-შემთხვევით მნიშვნელობებს;
- გამოთვლის მასივის კენტი მნიშვნელობის ელემენტების კვადრატების ჯამს;
- გამოთვალის მასივის ლუწი მნიშვნელობის ელემენტების კვადრატების ჯამს;
- შეადარებს ერთმანეთს წინა ორ ეტაპზე მიღებულ შედეგებს.

ამოხსნა

```
#include  <iostream>
#include <iomanip>
#include <stdlib>
#include <math>
main( )
```

```
{
const int size=10;
int a[size];
long s1=0, s2=0, max;
srand(time(0));
for(int i=0; i<size; i++)
{
 a[i]=1+rand()%80;
cout<<setw(4)<<a[i];
if(a[i]%2==1)
s1+=pow(a[i],2);
if(a[i]%2==0)
s2+=pow(a[i],2);
}
max=s1;
if(max<s2)
max=s2;
cout<<"\ns1="<<s1<<endl;
cout<<"s2="<<s2<<endl;
cout<<"max="<<max;
return 0;
}
```

<u>დავალება</u>

C++ ალგორითმულ ენაზე შეადგინეთ შემდეგი ამოცანების გადაწყვეტის პროგრამები:

1. მოცემულია ერთგანზომილებიანი მასივი **int B[10]**.
   - მასივის ელემენტებს შემთხვევითი რიცხვების გენერირების გზით [1;90] დიაპაზონიდან მიანიჭეთ მთელი ტიპის კვაზი-შემთხვევითი მნიშვნელობები;
   - გამოთვალეთ მასივის კენტი მნიშვნელობის ელემენტების საშუალო არითმეტიკული;
   - გამოთვალეთ მასივის ლუწი მნიშვნელობის ელემენტების საშუალო არითმეტიკული;

- შეადარეთ ერთმანეთს წინა ორ ეტაპზე მიღებული შედეგები.

2. მოცემულია ერთგანზომილებიანი მასივი **int X[15]**.
   - მასივის ელემენტებს შემთხვევითი რიცხვების გენერირების გზით [1;100] დიაპაზონიდან მიანიჭეთ მთელი ტიპის კვაზი-შემთხვევითი მნიშვნელობები;
   - განსაზღვრეთ მასივის მაქსიმალური მნიშვნელობის ელემენტი;
   - განსაზღვრეთ მასივის მინიმალური მნიშვნელობის ელემენტი;
   - შეადარეთ ერთმანეთს წინა ორ ეტაპზე მიღებული შედეგები.
3. მოცემულია ერთგანზომილებიანი მასივი **int A[17]**.
   - მასივის ელემენტებს შემთხვევითი რიცხვების გენერირების გზით [-15;20] დიაპაზონიდან მიანიჭეთ მთელი ტიპის კვაზი-შემთხვევითი მნიშვნელობები;
   - გამოთვალეთ მასივის დადებითი ელემენტების ნახევარჯამი;
   - გამოთვალეთ მასივის უარყოფითი ელემენტების ნამრავლი;
   - შეადარეთ ერთმანეთს წინა ორ ეტაპზე მიღებული შედეგები.
4. მოცემულია ერთგანზომილებიანი მასივი **int C[12]**.
   - მასივის ელემენტებს შემთხვევითი რიცხვების გენერირების გზით [-15;25] დიაპაზონიდან მიანიჭეთ მთელი ტიპის კვაზი-შემთხვევითი მნიშვნელობები;
   - გამოთვალეთ მასივის დადებითი ელემენტების ჯამიდან კვადრატული ფესვის მნიშვნელობა;

- გამოთვალეთ მასივის უარყოფითი ელემენტების ნამრავლი;
- შეადარეთ ერთმანეთს წინა ორ ეტაპზე მიღებული შედეგები.

5. მოცემულია ერთგანზომილებიანი მასივი **int D[14]**.
   - მასივის ელემენტებს შემთხვევითი რიცხვების გენერირების გზით [1;120] დიაპაზონიდან მიანიჭეთ მთელი ტიპის კვაზი-შემთხვევითი მნიშვნელობები;
   - გამოთვალეთ მასივის კენტინდექსიანი ელემენტების ნამრავლი;
   - გამოთვალეთ მასივის ლუწინდექსიანი ელემენტების კუბების ჯამი;
   - შეადარეთ ერთმანეთს წინა ორ ეტაპზე მიღებული შედეგები.
6. მოცემულია ერთგანზომილებიანი მასივი **int F[18]**.
   - მასივის ელემენტებს შემთხვევითი რიცხვების გენერირების გზით [1;70] დიაპაზონიდან მიანიჭეთ მთელი ტიპის კვაზი-შემთხვევითი მნიშვნელობები;
   - გამოთვალეთ მასივის 20-ზე ნაკლები მნიშვნელობის მქონე ელემენტების კვადრატების ჯამი;
   - გამოთვალეთ მასივის 40-ზე მეტი მნიშვნელობის მქონე ელემენტების ნამრავლი;
   - შეადარეთ ერთმანეთს წინა ორ ეტაპზე მიღებული შედეგები.
7. მოცემულია ერთგანზომილებიანი მასივი **int G[11]**.
   - მასივის ელემენტებს შემთხვევითი რიცხვების გენერირების გზით 1;60] დიაპაზონიდან მიანიჭეთ მთელი ტიპის კვაზი-შემთხვევითი მნიშვნელობები;

- გამოთვალეთ მასივის კენტინდექსიანი და ამავდროულად ლუწი მნიშვნელობის მქონე ელემენტების ჯამი;
- გამოთვალეთ მასივის ლუწინდექსიანი და ამავდროულად კენტი მნიშვნელობის მქონე ელემენტების ჯამი;
- შეადარეთ ერთმანეთს წინა ორ ეტაპზე მიღებული შედეგები.

8. მოცემულია ერთგანზომილებიანი მასივი **int H[19]**.
   - მასივის ელემენტებს შემთხვევითი რიცხვების გენერირების გზით [-15;27] დიაპაზონიდან მიანიჭეთ მთელი ტიპის კვაზი-შემთხვევითი მნიშვნელობები;
   - განსაზღვრეთ მინიმალური ელემენტი მასივის დადებითი მნიშვნელობის ელემენტებს შორის;
   - განსაზღვრეთ მაქსიმალური ელემენტი მასივის უარყოფით ელემენტებს შორის;
   - დააჯამეთ წინა ორ ეტაპზე მიღებული შედეგები.
9. მოცემულია ერთგანზომილებიანი მასივი **int S[15]**.
   - მასივის ელემენტებს შემთხვევითი რიცხვების გენერირების გზით [-10;20] დიაპაზონიდან მიანიჭეთ მთელი ტიპის კვაზი-შემთხვევითი მნიშვნელობები;
   - განსაზღვრეთ მაქსიმალური ელემენტი მასივის დადებითი მნიშვნელობის ელემენტებს შორის;
   - განსაზღვრეთ მინიმალური ელემენტი მასივის უარყოფით ელემენტებს შორის;
   - შეკრიბეთ წინა ორ ეტაპზე მიღებული შედეგები.
10. მოცემულია ერთგანზომილებიანი მასივი **int K[16]**.
    - მასივის ელემენტებს შემთხვევითი რიცხვების გენერირების გზით [-10;15] დიაპაზონიდან მიანიჭეთ მთელი ტიპის კვაზი-შემთხვევითი მნიშვნელობები;

- განსაზღვრეთ მაქსიმალური ელემენტი მასივის დადებითი მნიშვნელობის ელემენტებს შორის;
- განსაზღვრეთ მაქსიმალური ელემენტი მასივის უარყოფით ელემენტებს შორის;
- გადაამრავლეთ წინა ორ ეტაპზე მიღებული შედეგები.

პროგრამული კოდი და მიღებული შედეგები:

(ქულა)

(ხელმოწერა)

ლაბორატორიული სამუშაო №15
მოქმედებები სიმბოლურ მასივებზე

სიმბოლური კონსტანტა (მუდმივა) არის მთელი ტიპის მნიშვნელობა, რომელიც ერთმაგ აპოსტროფებში მოთავსებული სიმბოლოს სახითაა წარმოდგენილი. მაგალითად, **'z'** ჩანაწერი ნიშნავს მოცემული სიმბოლოს მთელრიცხა მნიშვნელობას, ხოლო **'\n'** ახალ სტრიქონზე გადასვლის სიმბოლოს მთელრიცხვა მნიშვნელობას.

სტრიქონი განიხილება, როგორც სიმბოლოთა თანამიმდევრობა, რომელიც მუშავდება ერთიანი მოდულის სახით. იგი შეიძლება შეიცავდეს ასოებს, ციფრებს, სხვადასხვა სპეციალურ სიმბოლოებს (მაგ: **+**, **–**, *****, **/**, **$**) და ა.შ.

სტრიქონული კონსტანტები (ლიტერალები) ორმაგ აპოსტროფებში თავსდება შემდეგი სახით: "Lela", "12 Main Street", "555-1212" და სხვ.

სტრიქონი C++-ში წარმოადგენს სიმბოლოთა მასივს, რომელიც ბოლოვდება ნულოვანი სიმბოლოთი (**'\0'**).

სტრიქონი შეიძლება განვსაზღვროთ სიმბოლური მასივის ან ***char*** ტიპის ცვლადის სახით. მაგალითად:

***char* color[ ]="green";**

***char* *colorPtr="rgeen";**

სტრიქონებთან სამუშაოდ C++-ში გამოიყენება **<string>** ბიბლიოთეკის შემდეგი ფუნქციები:

1. **strcpy** ფუნქცია, რომელიც შეიცავს ორ არგუმენტს და ახდენს თავისი მეორე არგუმენტის (სტრიქონის) კოპირებას პირველ არგუმენტში (სიმბოლურ მასივში). აღსანიშნავია, რომ კოპირებას განიცდის სტრიქონის დამამთავრებელი ნულოვანი სიმბოლოც;
2. **strncpy** ფუნქცია, რომელიც **strcpy** ფუნქციის ეკვივალენტურია და მისგან იმით განსხვავდება, რომ სამ არგუმენტს

შეიცავს და მიუთითებს სიმბოლოთა იმ რაოდენობას, რომელთა კოპირებაც უნდა განხორციელდეს სტრიქონიდან მასივში;

3. **strcat** ფუნქცია, რომელიც ორ არგუმენტს შეიცავს და თავის მეორე არგუმენტს (სტრიქონს) ამატებს თავის პირველ არგუმენტში (სიმბოლურ მასივში);
4. **strncat** ფუნქცია, რომელიც **strcat** ფუნქციის ეკვივალენტურია და მისგან იმით განსხვავდება, რომ სამ არგუმენტს შეიცავს და მიუთითებს სიმბოლოთა იმ რაოდენობას, რომელთა დამატება უნდა განხორციელდეს სტრიქონიდან მასივში;
5. **strcmp** ფუნქცია, რომელიც ორ არგუმენტს შეიცავს და თავისი არგუმენტების შედარებას ახდენს. იგი იძლევა ნულოვან შედეგს, თუკი არგუმენტები ერთიმეორის ტოლია, ხოლო უარყოფით შედეგს იმ შემთხვევაში, თუ პირველი არგუმენტი მეორეზე ნაკლებია და დადებით შედეგს დარჩენილ შემთხვევაში (ე.ი. თუ პირველი არგუმენტი მეორეზე მეტია).
6. **strncmp** ფუნქცია, რომელიც **strcmp** ფუნქციის ეკვივალენტურია და მისგან იმით განსხვავდება, რომ სამ არგუმენტს შეიცავს და სტრიქონებს შეადარებს სიმბოლოთა მხოლოდ მითითებული რაოდენობით. რადგან ეგმ-ში ყველა სიმბოლო წარმოდგენილია რიცხვითი კოდების სახით, ამიტომ, როდესაც კომპიუტერი ერთმანეთს ადარებს სიმბოლოებს, სინამდვილეში ადგილი აქვს ამ სიმბოლოების რიცხვითი კოდების შედარებას. ამასთან, სიმბოლოთა კოდები განლაგებულია ლათინური ანბანის ასოების მიხედვით.
7. **strtok** ფუნქცია, რომელიც გამოიყენება სტრიქონის ლექსემებად დაყოფის მიზნით. **ლექსემა** წარმოადგენს იმ სიმბოლოთა თანამიმდევრობას, რომლებიც ერთმანეთისგან გამოყოფილია პუნქტუაციის ნიშნებით ან ჰარით.

8. **strlen** ფუნქცია, რომელიც არგუმენტის სახით იღებს სტრიქონს და განსაზღვრავს მასში შემავალი სიმბოლოების რაოდენობას (სტრიქონის სიგრძეს) ნულოვანი სიმბოლოს ჩაუთვლელად.

C++-ში სტრიქონებთან სამუშაოდ გამოიყენება **<ctype.h>** ბიბლიოთეკის ფუნქცია **toupper**, რომელიც ლათინური ანბანის დიდი ასოებით ჩაწერილ სტრიქონს ამავე ანბანის პატარა სიმბოლოებით ცვლის.

მაგალითი 15.1. შევადგინოთ პროგრამა, რომელიც მოახდენს ნებისმიერი ტექსტის შემცველი სტრიქონის პირველი 17 სიმბოლოს კოპირებას სხვა სტრიქონში.

ამოხსნა

```
#include <iostream>
#include <string>
main( )
{
char s1[20]="Happy New Year!";
char s2[30]=" ";
cout<<s1<<endl;
cout<<strncpy(s2, s1, 17)<<endl;
return 0;
}
```

მაგალითი 15.2. შევადგინოთ პროგრამა, რომელიც ნებისმიერი ტექსტის შემცველ სტრიქონს ამატებს სხვა სტრიქონში.

ამოხსნა

```
#include <iostream>
#include <string>
main( )
{
char s1[50]="My Name ";
char s2[ ]="Is Lela Gachechiladze.";
cout<<s1<<endl;
cout<<strcat(s1, s2)<<endl;
```

```
return 0;
}
```

მაგალითი 15.3. შევადგინოთ პროგრამა, რომელიც სამ სიმბოლურ მასივს ერთმანეთს შეადარებს.

ამოხსნა:

```
#include <iostream>
#include <string>
main( )
{
char s1[ ]="My name is Lela.";
char s2[ ]="My name is Lela.";
char s3[ ]="My name is Lena.";
cout<<"strcmp(s1, s2)="<<strcmp(s1, s2)<<endl;
cout<<"strcmp(s1, s3)="<<strcmp(s1, s3)<<endl;
cout<<"strcmp(s3, s2)="<<strcmp(s3, s2)<<endl;
return 0;
}
```

მაგალითი 15.4. შევადგინოთ პროგრამა, რომელიც განსაზღვრავს ნებისმიერი სამი სტრიქონის სიგრძეს.

ამოხსნა

```
#include <iostream>
#include <string>
main( )
{
char s1[ ]="London";
char s2[ ]="Paris";
char s3[ ]="Tbilisi";
cout<<"London="<<strlen(s1)<<endl;
cout<<"Paris="<<strlen(s2)<<endl;
cout<<"Tbilisi="<<strlen(s3)<<endl;
return 0;
}
```

მაგალითი 15.5. შეადგინოთ პროგრამა, რომელიც განსაზღვრავს თუ რამდენი "**o**" სიმბოლოა შემდეგ სტრიქონში: "**Welcome to Moscow!**"

ამოხსნა

```
#include <iostream>
#include <string>
main( )
{
char s[ ]="Welcome to Moscow!";
char k;
 int m=0, i=0;
 cout<<"Enter a symbol:";
 cin>>k;
 while(s[i]!='\0')
 {
 if(s[i]==k)
 m++;
 i++;
 }
 cout<<"Result="<<m<<endl;
 return 0;
 }
```

<u>დავალება</u>

1. შეადგინეთ პროგრამა, რომელიც ნებისმიერი ტექსტის შემცველი სტრიქონის კოპირებას მოახდენს სხვა სტრიქონში.
2. შეადგინეთ პროგრამა, რომელიც ნებისმიერი ტექსტის შემცველი სტრიქონის პირველ შვიდ სიმბოლოს სხვა სტრიქონში ამატებს.
3. შეადგინეთ პროგრამა, რომელიც ნებისმიერი ტექსტის შემცველ ოთხ მასივს ერთმანეთთან შეადარებს და განსაზღვრავს თითოეულ მასივში შემავალი სტრიქონის სიგრძეს.

4. შეადგინეთ პროგრამა, რომელიც განსაზღვრავს თქვენთვის საინტერესო სიმბოლოებისა და ციფრების რაოდენობას მოცემულ სტრიქონში.
5. შეადგინეთ პროგრამა, რომელიც ნებისმიერ სტრიქონს დაყოფს ლექსემებად.
6. შეადგინეთ პროგრამა, რომელიც ნებისმიერი ტექსტის შემცველი სამი მასივის პირველ ხუთ სიმბოლოს ერთმანეთთან შეადარებს.
7. შეადგინეთ პროგრამა, რომელიც ნებისმიერი ტექსტის შემცველი სტრიქონის პირველი ორი სიტყვის კოპირებას მოახდენს სხვა სტრიქონში.
8. შეადგინეთ პროგრამა, რომელიც განსაზღვრავს თქვენ მიერ აკრეფილ ტექსტში სასურველი სიმბოლოს რაოდენობას.
9. შეადგინეთ პროგრამა, რომელიც ნებისმიერი ტექსტის შემცველი სტრიქონის პირველ ათ სიმბოლოს სხვა სტრიქონში ამატებს.
10. შეადგინეთ პროგრამა, რომელიც განსაზღვრავს ორი სიმბოლური მასივის სიგრძეს და მოახდენს ამ მასივების კონკატენაციას (გაერთიანებას).

პროგრამული კოდი და მიღებული შედეგები:

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

ლიტერატურა

1. **ლ. გაჩეჩილაძე. დაპროგრამების საფუძვლები (ალგორითმული ენა C++. ნაწილი I.** დამხმარე სახელმძღვანელო). საქართველოს ტექნიკური უნივერსიტეტი 2004. 57 გვ. ISBN 99940-35-50-9.
2. **ლ. გაჩეჩილაძე. დაპროგრამების საფუძვლები (ალგორითმული ენა C++. ნაწილი II.** დამხმარე სახელმძღვანელო). საქართველოს ტექნიკური უნივერსიტეტი 2005. 66 გვ. ISBN 99940-40-02-2.
3. **ლ. გაჩეჩილაძე, ლ. ნონიკაშვილი. დაპროგრამების საფუძვლები (ალგორითმული ენა C++. ნაწილი III.** დამხმარე სახელმძღვანელო). საქართველოს ტექნიკური უნივერსიტეტი 2006. 120 გვ. ISBN 99940-57-40-5.
4. Х.М. Дейтел, П.Дж. Дейтел Как программировать на C++. Издательство БИНОМ. Москва 1999.
5. Шелест В. Д. Программирование. Учебное пособие. _СПБ. БХВ-Петербург.-2001.
6. П. Франка C++. Учебный курс. Издательство ПИТЕР. Санкт-Петербург. 1999.
7. Алексеев В.Е. и др. Вычислительная техника и программирование.- М. Высшая школа. -1991.

ს ა რ ჩ ე ვ ი

გვ.

იბეჭდება ავტორთა მიერ წარმოდგენილი სახით

გადაეცა წარმოებას ----------. ხელმოწერილია დასაბეჭდად ----------. ქაღალდის ზომა X-- 1/8. პირობითი ნაბეჭდი თაბახი -. ტირაჟი 100 ეგზ.

ი.მ. "გოჩა დალაქიშვილი", ვარკეთილი 3, კ. 333, ბ. 38