ინგა გაბისონია, ლალი გესელია,

გურანდა ჩარქსელიანი

ამოცანათა კრებული დაპროგრამებაში

მეთოდური მითითებებითა და

ამოხსნებით

ნაწილი IV

გამომცემლობა „უნივერსალი“
თბილისი 2012

UDC(უაკ) 004.42(076)

გ-127

კრებულის ამ ნაწილში განხილულია სხვადასხვა სირთულის ამოცანები, რეალიზებული C++ დაპროგრამების ენაზე (Devcpp ვერსია 4.9.9.2, Borland C++ ვერსიები 3.1, 5.02 და ა.შ.). განხილული ამოცანები თემატურად დაკავშირებულია სტრიქონებთან, ფუნქციებთან და ფაილებთან. მოყვანილია ზოგიერთი ალგორითმების მოკლე აღწერილობები. კრებულში შესულია როგორც სავალდებულო მასალა, ასევე ორიგინალური ამოცანებიც.

რეცენზენტი - თსუ-ს ზუსტ და საბუნებისმეტყველო მეცნიერებათა ფაკულტეტის ასოცირებული პროფესორი, ფიზიკა-მათემატიკის დოქტორი ირინა ხუციშვილი

გამომცემლობა **„უნივერსალი“, 2012**

თბილისი, 0179, ი. ჭავჭავაძის გამზ. 19, ☎: 2 22 36 09, 5(99) 17 22 30
E-mail: universal@internet.ge

ISBN 978-9941-0-0753-8(ყველა ნაწილის)

ISBN 978-9941-0-5120-3(მეოთხე ნაწილის)

1. შესავალი

განვიხილოთ რამდენიმე საინტერესო ამოცანა, რომლებიც წინა ნაწილებში განხილულ საკითხებს ეხება.

ამოცანა 1. დავითვალოთ მატრიცაში მთავარი დიაგონალის ზემოთ განლაგებული ელემენტების საშუალო არითმეტიკული.

```
#include <iostream.h>
#include <conio.h>
main()
int mas[5][5], i, j, sum, s;
float sash;
cout<<”shemovitanot masivis elementebi”;
for(i=0; i<n; i++)
for(j=0; j<n; j++)
cin>>mas[i][j];
s=0; sum=0;
for(i=0; i<n; i++)
for(j=0; j<n; j++)
if (j<i)  {s++; sum+=mas[i][j];}
sash=sum/s;
cout<<”mocemuli matricis mtavar diagonals zemot “;
cout<<”elementebis sashualoa”<<sash;
getch();
}
```

ამოცანა 2. ვიპოვოთ მთელი დადებითი რიცხვის ყველა გამყოფი.

```
#include <iostream.h>
#include<conio.h>
int main()
{int num;
cout<<”\n shemoitant ricxvi….”; cin>>num;
int half=num/2;
int div=2;
while (div<=half)
```

```
{ if (!(num%div))  cout<<div<<"\n";
div++;}
getch();
}
```

ამოცანა 3. მოცემულია ნამდვილი რიცხვი, ამოვიღოთ ამ რიცხვიდან ფესვი ცნობილი იტერაციული ფორმულით: $y_n = \frac{1}{2}\left(y_{n-1} + x/y_{n-1}\right)$ წინასწარ მოცემული eps სიზუსტით , (სადაც y_{n-1} ფესვის წინა მნიშვნელობაა, y_n - ფესვის მომდევნო მნიშვნელობაა, fabs() - აბსოლუტური მნიშვნელობის პოვნის სტანდარტული ფუნქციაა.

```
# include <iostream.h>
#include <conio.h>
#include <math.h>
int main()
{double x,eps;
double  yp,y=1;
cout<<"shemoitanet argument da sizuste";
cin>>x>>eps;
do
{yp=y;
y=(yp+x/yp)/2;}
while (fabs(y-yp)>=eps);
cout<<"fesvi "<<x<<"-dan  ="<<y;
getch(); }
```

ამოცანა 4. დავწეროთ პროგრამა, რომელიც ითვლის ჰიპერბოლურ სინუსს x ნამდვილი ცვლადისათვის eps მოცემული სიზუსტით შემდეგი ფორმულის მეშვეობით:

$$sh\,x = 1 + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \cdots$$

```
#include <iostream.h>
#include <conio.h>
```

```
#include <math.h>
int main()
{const int miter=500;  //iteraciaTa raodenobis shezgudva
double  x, eps;
cout<<"\n shemoitanet argument da sizuste";
cin>>x>>eps;
bool f=true; //scorad  gamotvlis nishani
double y=x; ch=x;
for( int n=0; fabs(ch)>eps; n++)
{ch*=x*x/(2*n+2)/(2*n+3); //mckrivis mimdinare cevri
y+=ch;
if (n>miter)
{cout<<"\n mckrivi ganshladia";
f=false; break;}
}
if (f) cout<<"\n funqciis mnishvneloba:"<<y;
getch();
return 0;}
```

ამოცანა 5. მოცემულია მთელი რიცხვების მასივი, რომელიც კლავიატურიდან უნდა წავიკითხოთ. დავწეროთ პროგრამა, რომელიც გაარკვევს, წარმოადგენს ეს მასივი მუდმივ მიმდევრობას თუ არა.

```
#include <iostream.h>
#include <conio.h>
main()
{int a[10], i, k;
cout<<"\n shemovitanot masivis elementebi";
for(i=0; i<10; i++)
cin>>a[i];
k=0;
for(i=1; i<10; i++)
if ( a[i]!=a[0])  k++;
if (k==0)  cout<<"\n mimdevroba mudmivia";
   else cout<<"\n mimdevroba mudmivi ar aris";
```

```
getch();
return 0;}
```

იგივე ამოცანის უკეთესი რეალიზაციაა:

```
#include <iostream.h>
#include <conio.h>
main()
{int a[10], i; bool t=true;
cout<<"shemovitanot masivis elementebi";
for(i=0; i<10; i++)
cin>>a[i];
for(i=1; (i<10)&&(t==true); i++)
   if (a[i]!=a[0]) t=false;
if (t==false) cout<<"\n mimdevroba mudmivi ar aris";
else cout<<"\n mimdevroba mudmivia";
getch();
return 0;}
```

ამოცანა 6. მოცემულია მთელი ტიპის მასივი, დავითვალოთ ამ მასივში მხოლოდ მარტივი ელემენტების რაოდენობა.

```
#include <iostream.h>
#include <conio.h>
main()
{int a[50], i,k,j,n=0;
cout<<"\n shemovitanot  masivis elementebi";
for (i=0; i<50; i++)  cin>>a[i];
for (i=0; i<50; i++)
{
if  (a[i]==2)  {n=n+1;break;}
k=0;
for (j=2; j<=a[i]/2; j++)
if (a[i] % j==0) {k=1; break;}
if (k==0) n=n+1;}
cout<<"martivi elementebis raodenoba n="<<n;
```

```
getch();
return 0;}
```

იგივე პროგრამის სხვანაირი რეალიზაცია:

```
#include <iostream.h>
#include<conio.h>
main()
{ int a[10], i, x, k=0; bool t;
cout<<”shemoitanet masivis elementebi”;
for(i=0; i<10; i++)  cin>>a[i];
for(i=0; i<10; i++)
{t=true;
for(x=2; (x<=a[i]/2)&&(t=true); x++)
if (a[i] % x==0)  t=true;
if ( (t=true)&&(a[i]!=1) )  k++;}
cout<<”\n martivi elementebis raodenobaa”<<k;
getch();
return 0;}
```

ამოცანა 7. შემოვიტანოთ მთელი ტიპის მართკუთხა a[n][m] მასივი, ვიპოვოთ ამ მასივის სტრიქონების ელემენტთა ჯამები და დავალაგოთ მასივის ელემენტები ამ ჯამების ზრდადობის მიხედვით.

```
#include<iostream.h>
#include<iomanip.h>
#include<conio.h>
int main()
{int const n=3, m=4;
 int i,j,a[n][m];
int sum[n];
cout<<"shemovitanot masivis elementebi:";
for(i=0; i<n; i++)
for(j=0; j<m; j++)
cin>>a[i][j];
```

```
cout<<"gamovitanot masivis elementebi da";
 cout<<"mati shesabamisi jamebi:"<<endl;
for(i=0; i<n; i++)
{
sum[i]=0;
for(j=0;j<m;j++)
sum[i]+=a[i][j];
}
for(i=0;i<n;i++){
for(j=0;j<m;j++)
cout<<a[i][j]<<" ";
cout<<"|"<<sum[i]<<endl;
}
cout<<endl;
long buf_sum;
int nmin,buf_a;

for(i=0;i<n-1;i++)
{nmin=i;
for(j=i+1;j<n;j++)
if (sum[j]<sum[nmin]) nmin=j;
buf_sum=sum[i]; sum[i]=sum[nmin];
sum[nmin]=buf_sum;s
for(j=0;j<m;j++)
{buf_a=a[i][j];a[i][j]=a[nmin][j];a[nmin][j]=buf_a;}}
cout<<"gamovitanot dalagebuli masivi:"<<endl;
for(i=0;i<n;i++)
{
for(j=0;j<m;j++)
cout<<setw(4)<<a[i][j]<<" ";
cout<<endl;
}
getch();
return 0;
}
```

2. ამოცანები სიმბოლოთა მასივებზე

ჩვენ საილუსტრაციოდ განვიხილავთ მაგალითებს, სადაც გამოვიყენებთ სიმბოლოებისაგან შემდგარ მასივებს.

ამოცანა 8. მოცემულია სიმბოლოებისაგან შემდგარი მასივი, რომელიც უნდა შემოვიტანოთ კლავიატურიდან. დავთვალოთ ამ მასივში ციფრების ოდენობა. ციფრების დათვლა განვახორციელოთ ფუნქციაში. (მასივის შემოტანის დროს უნდა გავითვალისწინოთ რომ სიმბოლოს წარმოადგენს აგრეთვე ე.წ. ”პრობელი” ანუ ღილაკი space.)

```
#include <iostream.h>
#include <conio.h>
#define n 10
int datvla(char a[ ], int m);
main()
{ char a[n]; int i,k;
cout<<”\n shemovitanot masivis elementebi”;
for (i=0; i<n; i++) cin>>a[i];
k=datvla(a,n);
cout<<”\n cifrebis raodenobaa”<<k;
getch(); }
int datvla(char a[ ], int m)
{int i, d=0;
for(i=0; i<n; i++)
if ( (a[i]<=’0’) && (a[i]>=’9’)) d++;
return d;
}
```

ამოცანა 9. მოცემულია მასივი, რომელიც სიმბოლოებისაგან შედგება. დავწეროთ პროგრამა, რომელიც გაარკვევს, კლავიატურიდან შემოტანილ მასივში რა არის მეტი, ციფრები თუ სხვა სიმბოლოები?

```
#include <iostream.h>
#include <conio.h>
#define n 15
main( )
{char a[n]; int i,b,d;
cout<<"\n shemovitanot masivis elementebi";
for (i=0; i<n; i++)
cin>>a[i];
for (i=0; i<n; i++)
if ((a[i]>='0') && (a[i]<='9') b++;
else d++;
if (b>d) cout<<"\n cifrebis raodenoba metia";
else if (b==d) cout<<"\n cifrebis raodenoba da";
 cout<<"simboloebis raodenoba tolia";
else cout<<"\n simboloebis raodenoba metia";
getch();
return 0;}
```

იგივე ამოცანის რეალიზაცია შეიძლება ფუნქციის გამოყენებით:

```
#include <iostream.h>
#include <conio.h>
int metoba (char a[ ], int n)
{int  i, p, s=0, k=0;
for (i=1; i<=n; i++)
if ((a[i]='0') && (a[i]<='9') s=s+1; else k=k+1;
if (k>s) p=0; else   if(k<s) p=1;
else p=2;
return p;}
main ()
{int i,m;
char a[10];
cout<<"\n shemovitanot masivis elementebi";
for(i=1; i<=10; i++) cin>>a[i];
m=metoba(a,10);
if (m==0) cout<<"\n simboloebi metia";
```

```
else if (m==1) cout<<"\n ricxvebia meti";
else cout<<"\n ricxvebis da simboloebis odenoba tolia";
getch();}
```

ამოცანა 10. მოცემულია char ტიპის მასივი, რომელიც კლავიატურიდან უნდა წავიკითხოთ. დავწეროთ პროგრამა, რომელიც გაარკვევს, ამ მასივში დიდი ასოითი სიმბოლოებია უფრო მეტი, თუ პატარა ასოითი სიმბოლოები .

```
#include <iostream.h>
#include <conio.h>
void abc (char a[ ], int n)
{int i, k=0, p=0;
for (i=0; i<n; i++)
{if ((a[i]>='a') && (a[i]<='z')) k++;
if ((a[i]>='A') && (a[i]<='Z')) p++;}
if (k<p) cout<<"\n patarebi metia"<<k;
else if (k==p) cout<<"\n ertnairi raodenobaa"<<k;
else cout<<"\n didebi metia"<<p;
}
main()
{int i;
char a[10];
cout<<"\n shemovitanot masivis elementebi \n";
for (i=0; i<10; i++)  cin>>a[i];
abc(a,10);
getch();
}
```

ამ ამოცანის მოდიფიკაციას წარმოადგენს შემდეგი ამოცანა:

ამოცანა 11. დავწეროთ პროგრამა, რომელიც სიმბოლურ მასივში დაითვლის ცალ-ცალკე დიდი და პატარა ასოების რაოდენობას. სხვა სიმბოლოები არ გავითვალისწინოთ.

```
#include <iostream.h>
#include<conio.h>
```

```
void abc (char a[ ], int n)
{int i,  k=0,p=0;
for ( i=0; i<n; i++)
if ((a[i]>='a') && (a[i]<='z'))  k++;
else  if ((a[i]>='A') && (a[i]<='Z')) p++;
cout<<"\n patara asoebis raodenoba"<<k;
cout<<"\n didi asoebis raodenoba"<<p;}
main()
{int i; char a[ 10];
cout<<"\n shemovitanot masivis elementebi";
for (i=0; i<10; i++) cin>>a[i];
abc (a,10);
getch();}
```

ამოცანა 12. მოცემულია სიმბოლური მასივი, რომელიც კლავიატურიდან უნდა შემოვიტანოთ. დავწეროთ პროგრამა, რომელიც გაარკვევს, არის თუ არა ეს მასივი პალინდრომი (ანუ აქვს თუ არა მასივს სიმეტრიის ღერძი).

ცხადია, ერთმანეთს უნდა შევადაროთ a[0] და a[n-1] ელემენტები, a[1] და a[n -2] ელემენტები, a[2] და a[n -3] ელემენტები.... და ზოგადად, a[i] და a[n-1-i] ელემენტები.

```
#include <iostream.h>
#include <conio.h>
#define n 15
bool palindromi (char a[ ], int m)
{int i;
for ( i=0; i<m/2; i++)
if (a[i]!=a[n-1-i] ) return false;
return true;}
main()
{ char a[n]; bool p;
 int i;
cout<<"\n shemovitanot masivis elementebi";
for (i=0; i<n; i++)  cin>>a[i];
```

```
p=palindromi (a,n);
if (p==true) cout<<"\n masivi simetriulia";
else cout<<"\n masivi ar aris simetriuli";
getch();}
```

ამოცანა 13. მოცემულია სიმბოლური მასივი, რომელიც კლავიატურიდან უნდა შემოვიტანოთ. დავწეროთ პროგრამა, რომელიც გაარკვევს, ამ მასივის მხოლოდ ლუწ-ინდექსიანი ელემენტებისაგან შედგენილი მიმდევრობა არის თუ არა მუდმივი.

```
#include <iostream.h>
#include <conio.h>
#define n 20
bool  mimdevroba (char a[ ], int m)
{int i;
for (i=2; i<m; i+=2)
if (a[i]!=a[2]) return false;
return true;
}
main( )
{char a[n];
int i; bool p;
cout<<"\n shemovitanot masivis elementebi";
for (i=1; i<=n; i++)  cin>>a[i];
p=mimdevroba (a,n);
if (p==true) cout<<"\n mimdevroba mudmivia";
else cout<<"\n mimdevroba mudmivi ar aris";
getch();
return 0;}
```

ანალოგიურ სტილში შეიძლება დაიწეროს:

ამოცანა. მოცემულია სიმბოლური მასივი. დავწეროთ პროგრამა, რომელიც გაარკვევს, გვხვდება თუ არა ამ მასივში განმეორებადი სიმბოლოები. დავბეჭდოთ ის სიმბოლო, რომელიც ყველაზე მეტჯერ გვხვდება მოცემულ სიმბოლურ მასივში.

ამოცანა. დავწეროთ 12 ამოცანის ანალოგი პროგრამა კენტ-ინდექსიანი ელემენტების მიმდევრობის მუდმივობის შესახებ.

3. მუშაობა სტრიქონებთან.

ახლა მოვიყვანოთ მაგალითები, რომლებიც C++-ზე სტრიქონებთან მუშაობას შეეხება. აქვე განვიხილავთ სპეციალურ ფუნქციებს, რომლებიც სტრიქონებთან მუშაობისას გამოიყენება. ნებისმიერი პროგრამა სიმბოლოების თანმიმდევრობას წარმოადგენს. სწორედ სიმბოლოების გაერთიანებით მიღებულ ჯგუფებს გააჩნიათ გარკვეული შინაარსი. სწორედ ეს ჯგუფები დამუშავდება კამპილატორის მიერ და საბოლოოდ წარმოადგენს ინსტრუქციათა ერთობლიობას. პროგრამა აგრეთვე შეიძლება შეიცავდეს სიმბოლურ მუდმივებს. სიმბოლური მუდმივი (კონსტანტა) წარმოადგენს ერთმაგ ბრჭყალებში ჩაწერილი სიმბოლოს შესაბამის რიცხვით მნიშვნელობას. სტრიქონი წარმოადგენს სიმბოლოთა თანმიმდევრობას, რომელიც დამუშავდება, როგორც ერთიანი მოდული. სტრიქონი შეიძლება შეიცავდეს ასოებს, ციფრებს, და სხვადასხვა სპეციალურ სიმბოლოებს, როგორიცაა +, -, *, $,და სხვა. სტრიქონული მუდმივები იწერება ორმაგ ბრჭყალებში. ”qalaqi Tbilisi”.

C++-ზე სტრიქონი წარმოადგენს სიმბოლოთა მასივს, რომელიც დაბოლოვდება ნულოვანი სიმბოლოთი (’\0’). სტრიქონზე წვდომა ხორციელდება სტრიქონის პირველ სიმბოლოზე მიმთითებლის მეშვეობით. ამიტომაც შეიძლება ჩავთვალოთ, რომ C++-ზე სტრიქონი წარმოადგენს ისეთ მიმთითებელს, რომელიც მიუთითებს სტრიქონის პირველ სიმბოლოზე. ამ თვალსაზრისით სტრიქონი მასივის ანალოგია. (მასივში მიმთითებელი მიუთითებს მასივის პირველ ელემენტზე). სტრიქონი შეიძლება წარმოვადგინოთ როგორც მასივი, რომელიც შედგება სიმბოლოებისაგან, ან char* ტიპის ცვლადი.

```
char color [ ] =”blue”;
char *colorPtr=”blue”;
```

ორივე ბრძანება მიანიჭებს "blue" მნიშვნელობას. პირველი ვარიანტი შექმნის 6-ელემენტიან color მასივს, რომელიც შეიცავს სიმბოლოებს 'b', 'l', 'u', 'e', '\0', . მეორე ვარიანტი შექმნის colorPtr მიმთითებელ ცვლადს, რომელიც მიუთითებს მეხსიერებაში სადმე ჩაწერილ "blue" სტრიქონზე. პირველი ვარიანტი შესაძლებელია სხვანაირადაც ჩავწეროთ:

char color [ ] = {'b', 'l', 'u' ,'e', '\0'};

სტრიქონის ზომა აუცილებლად უნდა გულისხმობდეს ნულოვანი სიმბოლოს ადგილსაც. მასივში სტრიქონის ჩაწერისას შესაძლებელია ვისარგებლოთ შემდეგი კონსტრუქციით:

cin>>word; ან cin>>setw(20)>>word; (შეზღუდვა - 19 სიმბოლო).

სტრიქონების დამუშავებისათვის გამოიყენება სპეციალური ბიბლიოთეკა, რომელიც შეიცავს სხვადასხვა ფუნქციებს. გავეცნოთ ძირითად ფუნქციებს:

char *strcpy (char *s1, const char *s2)

s2 სტრიქონს ჩაწერს (ჩააკოპირებს) s1 სიმბოლოთა მასივში, აბრუნებს s1-ს.

char *strncpy (char *s1, const char *s2, size_t n)

არაუმეტეს n რაოდენობის სიმბოლოებს s2 სტრიქონიდან ჩაწერს (ჩააკოპირებს) s1 სიმბოლოთა მასივში, აბრუნებს s1-ს.

char *strcat (char *s1, const char *s2)

ამატებს s2 სტრიქონს s1 სტრიქონში. s2 სტრიქონის პირველი სიმბოლო ჩაიწერება s1 სტრიქონის ნულოვანი სიმბოლოს ადგილზე, აბრუნებს s1-ს.

char *strncat (char *s1, const char *s2, size_t n)

ამატებს არაუმეტეს n რაოდენობის სიმბოლოებს s2 სტრიქონიდან s1 სტრიქონში. s2 სტრიქონის პირველი სიმბოლო ჩაი წერება s1 სტრიქონის ნულოვანი სიმბოლოს ადგილზე, აბრუნებს s1-ს.

int strcmp (const char *s1, const char *s2)

ერთმანეთს ადარებს s1 და s2 სტრიქონებს. ფუნქცია აბრუნებს 0-ს, ნულზე ნაკლებ რიცხვს ან ნულზე მეტ რიცხვს, თუ შესაბამისად s1=s2, s1<s2, s1>s2.

int strcmp (const char *s1, const char *s2, size_t n)

ერთმანეთს ადარებს პირველ n სიმბოლოს s1 სტრიქონიდან და s2 სტრიქონებს. ფუნქცია აბრუნებს 0-ს, ნულზე ნაკლებ რიცხვს ან ნულზე მეტ რიცხვს, თუ შესაბამისად s1=s2, s1<s2, s1>s2.

char *strtok(char *s1, const char *s2)

ეს ფუნქცია იწვევს s1 სტრიქონის ”ლექსემებად” დაყოფას, რომლებიც s2-შია ჩაწერილი. შესაძლებელია ამ ფუნქციის რამდენიმეჯერ გამოძახება.

size_t strlen (const char *s)

განსაზღვრავს s სტრიქონის სიგრძეს. აბრუნებს იმ სიმბოლოთა რაოდენობას, რომლებიც ნულოვანი სიმბოლოს წინ არის ჩაწერილი.

განვიხილოთ საილუსტრაციო მაგალითები:

```
#include  <iostream.h>
#include  <string.h>
#include  <conio.h>
main()
{
char x[ ]=”gilocavt dabadebis dges!”;
char y[35], z[15];
cout<<”striqoni x masivshi: “<<x<<endl;
cout<<”striqoni y masivshi: “<<strcpy( x, y)<<endl;
strncpy( z, x, 9);
z[9]=’ \0’;
cout<<”striqoni z  masivshi: “<<z<<endl;
getch( );
return 0;
}
```

```
#include  <iostream.h>
#include  <string.h>
#include  <conio.h>
main()
{
char s1[35]=”bednier ”;
char s2[ ] =”axal cels gisurvebt ”;
```

```
char s3[40]="";
cout<<" s1= "<<s1<<endl<<" s2= "<<s2<<endl;
cout<<"strcat(s1, s2)= "<<strcat (s1, s2)<<endl;
cout<<"strncat(s1, s2, 8)= "<<strncat(s1, s2, 8)<<endl;
cout<<"strcat(s3, s1) ="<< strcat(s3, s1)<<endl;
getch();
return 0;
}

#include <iostream.h>
#include <iomanip.h>
#include <string.h>
#include <conio.h>

main()
{
char *s1 = "bednier axal cels";
char *s2 = "bednier axal cels";
char *s3 = "bednier dges gisurvebt";

cout<<"s1 = "<<s1<<endl<<"s2 = "<<s2<<endl
<<"s3 = "<< s3<<endl<< endl<< "strcmp (s1, s2) ="
<<setw(2)<<strcmp(s1,s2)<<endl<<"strcmp(s1, s3) ="
<<setw(2)<<strcmp(s1,s3)<<endl<<"strcmp(s3, s1) ="
<<setw(2)<<strcmp(s3, s1)<<endl<<endl;

cout<<"strncmp(s1, s3, 8) ="<<setw(2)
<<strncmp(s1, s3, 8)<<endl
<<"strcmp(s1, s3, 9) ="<<setw(2)
<<strncmp(s1, s3, 9)<<endl
<<"strcmp(s3, s1, 9) ="<<setw(2)
<<strncmp(s3, s1, 9)<<endl;
getch();
return 0;
}
```

განვიხილოთ strlen ფუნქციის გამოყენება:

```
#include<iostream.h>
#include<string.h>
#include<conio.h>
main()
{ char *string1=”abcdefghijklmnopqrstuvwxyz”;
char  *string2=”sami”;
char *string3=”Boston”;
cout<< ”striqonis sigrdze\””<<string1<<”\”-“<<strlen(string1)<<endl
<< ”striqonis sigrdze\””<<string2<<”\” - “<<strlen(string2)<<endl
<< “striqonis sigrdze\”’<<string3<<”\” - ” <<strlen(string3) <<endl;
getch();
return 0;
}
```

შედეგი:

```
სტრიქონის სიგრძე   ”abcdefghijklmnopqrstuvwxyz” – 26
სტრიქონის სიგრძე   ”sami” – 4
სტრიქონის სიგრძე   ”Boston” – 6
```

საინტერესოა აგრეთვე strtok ფუნქცია, რომლის გამოყენების მაგალითს ჩვენ აქ მოვიყვანთ:

```
#include<iostream.h>
#include<string.h>
#include<conio.h>
main()
{ char string=”es tcinadadeba sheicavs khut leqsemas”;
char  *tokenptr;
 cout<< ”striqoni, romelic iyofa leqsemebad:” <<endl
 <<string<<endl<<endl<<”leqsemebi:”<<endl;
  tokenptr= strtok(string, “ ”);
```

```
while (tokenptr!=NULL ) {
cout <<tokenptr << endl;
tokenptr = strtok(NULL, " ");
}

  getch();
  return 0;
}
```

striqoni, romelic iyofa leqsemebad:
es tcinadadeba sheicavs khut leqsemas

leqsemebi:
es
winadadeba
Seicavs
xuT
leqsemas

კლავიატურიდან სტრიქონის შეტანისათვის და გამოტანისათვის ჩვენ შეგვიძლია გამოვიყენოთ ჩვენთვის კარგად ნაცნობი cin>> და cout<< კონსტრუქციები.

```
#include<iostream.h>
#include<conio.h>
main()
{
char str[80];
cout<<"sheviyvanoT striqoni:";
cin>>str;
cout<<" tqveni striqoni:";
cout<<str;
getch();
return 0;
}
```

შედეგი:

შევიყვანოთ სტრიქონი: ეს შემოწმებაა
თვენი სტრიქონი; ეს

ამ ფრაგმენტის შესრულების შედეგად მოხდება წინადადებიდან მხოლოდ ერთი სიტყვის წაკითხვა; ამ პრობლემის გადასაწყვეტად , ანუ სტრიქონის სრულად წასაკითხად უნდა გამოვიყენოთ ფუნქცია gets();

```
#include<iostream.h>
#include<cstdio.h>
#include<conio.h>
main()
{
char str[80];
cout<<”sheviyvanoT striqoni:”;
gets(str);
cout<<” tqveni striqoni:”;
cout<<str;
getch();
return 0;
}
```

შედეგი:

SeviyvanoT striqoni: es Semowmebaa
Tveni striqoni; es Semowmebaa

ახლა განვიხილოთ კიდევ ერთი ფუნქცია strcpy(), რომელიც აკოპირებს “gamarjoba” სტრიქონს str სტრიქონში.

```
#include<iostream.h>
#include<cstring.h>
#include<conio.h>

main()
```

```
{
char str[80];
strcpy(str,  “gamarjoba” )
cin>>str;
cout<<” tqveni striqoni:”;
cout<<str;
getch();
return 0;
}
```

შემდეგ ხშირად გამოყენებად ფუნქციას წარმოადგენს strcat(), რომლის საშუალებით ხდება s2 სტრიქონის მიერთება (კონკატენაცია) s1 სტრიქონზე.

```
#include<iostream.h>
#include<cstring.h>
#include<conio.h>

main()
{
char s1[20], s2[10];
strcpy(s1,  “gamarjoba” );
strcpy(s2,  “yvelas!” );
strcat(s1,s2)
cout<<s1;
getch();
return 0;
}
```

ახლა განვიხილოთ strcmp() ფუნქცია, რომელიც ახდენს s1 და s2 სტრიქონების შედარებას.

```
#include<iostream.h>
#include<cstring.h>
#include<cstdio.h>
#include<conio.h>
```

```
bool password();
main()
{
 if (password())  cout << “shesvla neba dartulia.\n”;
else   cout<<”shezRudvaa. \n”;
getch();
return 0;
}
bool password()

{   char s[80];
cout<<”sheiyvanet paroli:”;
gets (s);
   if (strcmp(s, “paroli”)) {     //striqonebi ar emTxveva
cout<<”paroli tyuilia. \n”;
return false;
}
return true;      //striqonebi ertmanets emTxveva.
}
```

ხშირად გამოყენებადი ფუნქცია strlen(str) აბრუნებს სტრიქონის სიგრძეს

```
#include<iostream.h>
#include<cstring.h>
#include<cstdio.h>
#include<conio.h>
main()
{
 char str[80];
cout<<”shemovitanoT striqoni:”;
gets(str);
cout<<”striqonis sigrZea:” <<strlen(str);
getch();
return 0;
}
```

ამ უკანასკნელი ფუნქციის საილუსტრაციოდ მოვიყვანოთ სტრიქონის რევერსის პროგრამა (რევერსი შებრუნებული სახით დაბეჭდვას ნიშნავს).

```
#include<iostream.h>
#include<cstring.h>
#include<cstdio.h>
#include<conio.h>
main()
{
char str[80];
int i;
cout<<”shemovitanoT striqoni:”;
gets(str);
for(i=strlen(str)-1; i>=0; i--) cout<<str[i];
getch();
return 0;
}
```

მოვიყვანოთ კომბინირებული მაგალითი, რომელიც ზემოთ განხილული ფუნქციების დემონსტრირებას წარმოადგენს:

```
#include<iostream.h>
#include<cstring.h>
#include<cstdio.h>
#include<conio.h>

main()
{
char s1[80], s2[80];
cout<<”Semovitanot ori striqoni”;
gets(s1); gets(s2);
cout<<”mati sigrZeebi:”<<strlen(s1);
cout<<’ ’<<strlen(s2)<<’\n’;
if (!strcmp(s1,s2))
```

```
cout<<”striqonebi tolia\n”;
else cout<<”striqonebi ar aris toil\n”;
strcat(s1,s2);
cout<<s1<<’\n’;
strcpy(s1,s2);
cout<<s1<<”da”<<s2<<’ ’;
cout<<”axla tolebia\n”;
getch();
return 0;
}
```

კიდევ ერთი საკმაოდ ხშირად მოხმარებადი საილუსტრაციო პროგრამა, რომელიც სტრიქონის დიდი რეგისტრიდან პატარა რეგისტრში გადაყვანას ემსახურება (toupper( ) ფუნქციის მეშვეობით):

```
#include<iostream.h>
#include<cstring.h>
#include<cstdio.h>
#include<conio.h>
main()
{
char str[80];
int i;
strcpy(str,”test”);
for(i==0; str[i]; i++)  str[i]=toupper(str[i]);
cout<<str;
getch();
return 0;
}
```

ამოცანა 14. მოცემულია სტრინგი (სტრიქონი), რომელიც შემოგვაქვს კლავიატურიდან. დავწეროთ ფუნქცია, რომელიც გაარკვევს ამ სტრინგში ციფრული სიმბოლოები უფრო მეტი რაოდენობისაა, თუ ასოითი სიმბოლოები.

```
#include <iostream.h>
#include <conio.h>
#include <string.h>
int funkcia (char a[ ] )
{
int i, c=0, d=0;
for (i=0;  a[i] !='\0';  i++)
{ if ((a[i]>='0') && (a[i]<='9')) c++;
if ((a[i]>='A')&&(a[i]<='Z')||(a[i]>='a')&&(a[i]<='z')) d++;}
if (c>d) return 1;
if (c<d) return -1;
return 0;
 }
main()
{
char a[n];  int k;
cout <<"shemoitanet stringi";
cin>>a;
k=funkcia(a);
if (k==1)  cout<<"cifruli simboloebi metia";
if (k==-1)  cout<<"asoiti simboloebi metia";
if (k==0) cout<<"cifruli da asoiti simboloebi tolia";
getch ( );}
```

სტრიქონის შემოტანა-გამოტანისათვის C++-ში არსებობს სპეციალური ფუნქციები gets() და puts(). ორივე აღწერილია <stdio.h> ფაილში. ორივეს გააჩნია ერთადერთი პარამეტრი - სტრიქონი.

gets() კითხულობს კლავიატურიდან შეტანილ ყველა სიმბოლოს (მათ შორის არა მარტო ასოითი სიმბოლოები, არამედ სპეციალური სიმბოლოებიც) და ანიჭებს ამოკითხულ სიმბოლოთა მიმდევრობას თავის პარამეტრს (მრგვალ ფრჩხილებში მოცემულ სტრიქონს). puts() ფუნქციას გამოაქვს თავისი პარამეტრის - სტრიქონის - მნიშვნელობა ეკრანზე.

```
#include <stdio.h>
```

```
#include <conio.h>
        main ( )
        {
        char a[30];
        prinf("shemovitanot striqoni");
        gets(a);
        printf("\n striqonshi chaicera :");
        puts(a);
        getch();
        return 0;}
```

gets() -ის გამოყენების დროს სიმბოლოების მიმდევრობის შეტანა უნდა დასრულდეს ENTER ღილაკით. კამპილატორი დაამატებს სტრიქონს ნულოვან ბაიტს.
puts() ფუნქცია სტრიქონის გამოტანის შემდეგ ავტომატურად გადაიყვანს კურსორს ახალ სტრიქონზე, ანუ ფუნქცია ამატებს სტრიქონის ბოლოში ახალ სტრიქონზე გადასვლის სიმბოლოს.

ამოცანა 15. დავწეროთ პროგრამა, რომელიც ახდენს მოცემული სტრიქონის ინვერტირებას (ანუ შებრუნებას):

```
        #include <stdio.h>
        #include <conio.h>
        #include <string.h>
        main()
        {char a[20],p; int i,m;
        gets(a);
        m=strlen(a);
        for(i=0; i<m/2; i++)
        {
        p=a[i]; a[i]=a[m-i-1]; a[m-i-1]=p;
        }
        printf ("%s", a);
        getch();
        return 0;
        }
```

ამოცანა 16. წავიკითხოთ სტრიქონი. გავცვალოთ ამ სტრიქონში სიმბოლოები ვერტიკალური ღერძის მიმართ, ანუ პირველ სიმბოლოს გავუცვალოთ ადგილი ბოლო სიმბოლოსთან, მეორეს - ბოლოს წინასთან და ა.შ.

```
#include  <stdio.h>
#include <conio.h>
#include <string.h>
main( )
{char a[50],p; int i, m, j;
gets(a);
m=strlen(a);
for(i=0, j=m-1; i<j; i++, j--)
{ p=a[i]; a[i]=a[j]; a[j]=p; }
printf( “%s”, a);
getch();
return 0; }
```

ამოცანა 17. მოცემულია სტრიქონი, ამ სტრიქონიდან დავბეჭდოთ ის სიტყვები, რომლებიც იწყება ”ა”, ”დ”, ”კ”, ”რ” ასოებით.

```
#include <iostream.h>
#include <conio.h>
#include <string.h>
main ()
{
int len;
char str[255], strnew[255]="\0";
char *pointer;
cout<<"SemovitanoT striqoni: ";
cin.getline(str,254);
pointer=strtok(str, " ");

if(pointer[0]=='a'||pointer[0]=='d'||pointer[0]=='k'||pointer[0]=='p')
{strcat(strnew,pointer);strcat(strnew," ");}
```

```
while(1)
{
pointer=strtok(NULL, " ");
if(pointer==NULL)
{break;}

if(pointer[0]=='a'||pointer[0]=='d'||pointer[0]=='k'||pointer[0]=='p')
{strcat(strnew,pointer);strcat(strnew," ");}
}
cout<<" akhali striqoni: "<<strnew<<endl;
getch();
return 0;
}
```

cin.getline(str,254) ფუნქციით ხდება მთლიანი სტრიქონის შემოტანა.
strtok(str, " ფუნქციით ხდება str სტრიქონის დაყოფა ინტერვალებით, პირობითი ოპერატორით:
if(pointer[0]=='a'||pointer[0]=='d'||pointer[0]=='k'||pointer[0]=='p')
ვამოწმებთ , იწყება თუ არა ამ 4 სიმბოლოდან ერთ-ერთით მოცემული სიტყვა, რომელზეც უთითებს pointer; თუ იწყება ამ ასოებიდა ერთ-ერთით, მაშინ იწერება strnew სტრიქონში strcat(strnew,pointer) ფუნქციით და ემატება strcat(strnew," ") ფუნქციით ინტერვალი. while(1) არის უსასრულო ციკლი, რომელშიც მეორდება ეს პროცესი, მანამ სანამ არ მიაღწევს სტრიქონის ბოლომდე. როგორც კი მიაღწევს სტრიქონის ბოლოს ანუ თუ pointer==NULL-ს break-ით წყდება ციკლი და გამოვიტანთ ეკრანზე ახალ strnew სტრიქონს.

ამოცანა 18. მოცემული გვაქვს სიმბოლოებისგან შემდგარი სტრიქონი, შეუცვალოთ ადგილები ლუწ და კენტ ინდექსიან ელემენტებს.

```
#include <iostream.h>
#include <conio.h>
#include <string.h>
```

```
  main ()
{
int len;
char str[255];
char tmp;
cout<<"shemovitanot striqoni: ";
cin.getline(str,254);
len=strlen(str);
for(int i=0;i<len-1;i++)
{
if(i%2==0)
{
tmp=str[i];
str[i]=str[i+1];
str[i+1]=tmp;
}
}
cout<<str<<endl;
getch();
return 0;
}
```

cin.getline(str,254)-ით შემოგვაქვს მთელი სტრიქონი, len=strlen(str) ფუნქციით ვპოულობთ str სტრიქონის სიგრძეს. ციკლის ოპერატორში ვამოწმებთ 0-დან დაწყებული len-1-მდე ინდექსების ლუწ -კენტობას.

ამოცანა 19. დავწეროთ strchr() ფუნქციის ანალოგი. (strchr() ფუნქცია აბრუნებს მიმთითებელს საძებნ სიმბოლოზე).

```
#include <iostream.h>
#include <conio.h>
 using namespace std;
char* findch(char ma[], char a);
 main()
{
char line[128];
```

```
char fchar;
char *pointer;
cout<<"Sevitanot striqoni: ";
cin.getline(line,127);
cout<<"shevitanot sadzebni simbolo: ";
cin>>fchar;
pointer=findch(line,fchar);
cout<<"gamogvagvs simbolo "<<pointer[0]<<endl;
getch();
return 0;
}
char* findch(char ma[],char a)
{
int i=0;
while(ma[i]!='\0')
{
if(ma[i]==a)
return &ma[i];
else
i++;
}
return NULL;
}
```

ამოცანა 20. დავწეროთ strncat() ფუნქციის ანალოგი. (strncat(s1,s2,n) ფუნქცია პირველ სტრიქონს მიუერთებს მეორე სტრიქონის მითითებულ n რაოდენობის სიმბოლოებს).

```
#include <iostream.h>
#include <conio.h>
char* mystrncat(char *s1, char *s2, int num);
int main()
{       char *p1=new char [20];
char *p2=new char [20];
strcpy(p1, "abce");
strcpy(p2, "fghi");
```

```
char *p3;
p3=mystrncat(p1,p2, 3);
cout<<p3<<endl;
getch();
}
char* mystrncat(char *s1, char *s2, int num)
{       int len;
len=strlen(s1);
int i;
for(i=0; i<num+len; i++)
{
s1[len+i]=s2[i];
}
s1[i]=0;
return s1;
}
```

4. სიმბოლური ინფორმაციის შეტანა-გამოტანის საშუალებები.

იმ შემთხვევაში, როცა საჭიროა სიმბოლური ინფორმაციის დამუშავება, ეს ინფორმაცია პროგრამას შეიძლება მიეწოდებოდეს როგორც კლავიატურიდან, ასევე რაიმე ფაილიდან. ჯერ განვიხილოთ სიმბოლოების კლავიატურიდან შეტანის შესაძლებლობები.

სიმბოლოების შეტანა-გამოტანის სპეციალური სტანდარტული ფუნქციები C-ზე შემდეგია: getchar(), putchar(). ამ ფუნქციების პროტოტიპები მოთავსებულია <stdio.h>-ში.

პირველი მათგანი ch=getchar() კითხულობს თითოეულ სიმბოლოს კლავიატურიდან და აბრუნებს მის კოდს. ხოლო მეორე ფუნქცია putchar(ch) განათავსებს ch სიმბოლოს ეკრანზე. სიმბოლოების კლავიატურიდან შეტანა უნდა დავასრულოთ CTRL+Z(^Z) კლავიშების კომბინაციის აკრეფით. ამ ნიშნის ამოსაცნობად არსებობს სპეცილური ფუნქცია EOF(end of file), რომლის პროტოტიპიც აგრეთვე <stdio.h>-შია განთავსებული.

იმისათვის, რომ კლავიატურიდან შეტანილი სიმბოლოების მიმდევრობა გამოვიტანოთ ეკრანზე, უნდა ჩავწეროთ:

```
# include <stdio.h>
main()
{
char ch;
printf("Shemoitanet simbolota mimdevroba: ");
ch=getchar();
printf("Chven shevitanet mimdevroba: ");
while (ch!=EOF)
{
putchar(ch);
ch=getchar();
}
return 0;
}
```

ციკლი შეიძლება უფრო მოკლედაც ჩავწეროთ: while((ch=getchar()) !=EOF) putchar(ch);

ამოცანა 21. ქვემოთ მოყვანილი პროგრამა ითვლის კლავიატურიდან შემოტანილი სიმბოლოების რაოდენობას:

```
#include <stdio.h>
main()
{
long n=0;
n=0;
while (getchar()!=EOF)
++n;
printf("n=%ld", n);
return 0;
}
```

ამოცანა 22. შემდეგი პროგრამა დაადგენს, რამდენ სტრიქონად არის დაყოფილი კლავიატურიდან შეტანილი მიმდევრობა.

```
#include <stdio.h>
main()
{
char k; int n;                          /* n-striqonebis mtvlelia */
n=0;
while ( (k=getchar())!=EOF )
if (k=='\n')  ++n;
printf("n=%d", n);
return 0;
}
```

ამოცანა 23. მოვიყვანოთ პროგრამა, რომელიც კლავიატურიდან შემოსულ მიმდევრობაში დაითვლის სიმბოლოების, სიტყვების და სტრიქონების რაოდენობას.

```
#include <stdio.h>
main()
{
char c; int nc, nl, nw;
nc=nl=nw=0;
while (( c=getchar())!=EOF)
{++nc;
if (c=='\n') ++nl;
if (c==' '|| c=='\n' ||c=='\t') nw++;
}
printf("nl=%d nw=%d nc=%d", nl,nw,nc);
return 0;
}
```

ამოცანა 24. კიდევ ერთი საილუსტრაციო მაგალითი: კლავიატურიდან შემოტანილი სიმბოლოთა მიმდევრობა

დავბეჭდოთ ეკრანზე შემდეგი ცვლილებების განხორციელებით - პატარა ასოები გადავაქციოთ დიდებად და დიდები - პატარებად.

```
#include<stdio.h>
#include<ctype.h>
main()
{
char c;
while  ((c=getchar())!=EOF)
putchar ( isupper(c) ? tolower(c) : toupper(c) );
 /*aq ? da :  specialuri operciebia */
return 0;
}
```

ამოცანა 25. შემდეგ საილუსტრაციო მაგალითში კლავიატურიდან შემოტანილ სიმბოლოთა მიმდევრობაში დავადაგინოთ სიმბოლო-ციფრების რაოდენობა, ინგლისური ასოების რაოდენობა, ვიპოვოთ სიმბოლო-ციფრებით შედგენილი რიცხვის მნიშვნელობა და გავზარდოთ ის სამით (რადგანაც მას სიმბოლოების ერთობლიობიდან გადავაქცევთ ჩვეულებრივ რიცხვად, რომლის მიმართ შეიძლება არითმეტიკული ოპერაციების ჩატარება).

```
#include <stdio.h>
#include <conio.h>
main()
{
char c; int  nc=0, na=0; long k=0;
while ( (c=getchar() )!=EOF)
{
if ( c>='0' && c<='9')
{
nc++;
k=k*10+c-'0';
}
if ( c>='a' && c<='z' || c>='A' && c<='Z' ) na++;
```

```
}
k=k+3;
printf("nc=%d na=%d k=%ld \n", nc, na, k);
getch();
return 0;
}
```

ამოცანა 26. ახლა განვიხილოთ შემდეგი საილუსტრაციო მაგალითი: კლავიატურიდან შემოვიტანოთ სიმბოლოთა მიმდევრობა, შემდეგ ამ მიმდევრობაში ამოვაგდოთ სიმბოლო-ციფრები, დიდი ასოები შევცვალოთ პატარებით, პატარები - დიდებით, ყოველი სიმბოლო '+' შევცვალოთ ' * '-ით.

```
#include <stdio.h>
#include<conio.h>
main()
{
char p;
while (( p=getchar()) !=EOF)
{
if ( p<'0' || p>'9')          /*tu p ar aris cipri*/
{
if (p>='a' && p<='z')  p=p-'a'+'A';
  /* tu p patara simboloa*/
else if(p>='A'&& p<='Z') p=p-'A'+'a';
/* tu p didi simboloa*/
else  if (p=='+')  p='*';
putchar(p);
}
}
getch();
return 0;
}
```

ამოცანა 27. მოცემულია სიმბოლური მასივი, დავწეროთ პროგრამა, რომელიც შეამოწმებს, სიმბოლური მასივი არის თუ არა ანბანის მიხედვით დალაგებული.

```
#include<iostream.h>
#include<conio.h>
int dalageba(char a[], int n)
{int i;
for(i=0;i<n;i++)
if(!((a[i]>='A'&& a[i]<='z')||(a[i]>='a'&& a[i]<='z')))return -1;
for(i=0;i<n-1;i++)
if (a[i]>a[i+1]) return 1;
return 0;}
main()
{char a[5];
int i,k;
 for(i=0;i<5;i++)
cin>>a[i];
k=dalageba (a,5);
if (k==-1)cout<<"ar aris asoebis masivi";
if (k==0)cout<<"anbanis rigis mixedvit dalagebuli masivia";
if (k==1) cout<<"ar aris dalagebuli";
getch();
}
```

ამოცანა 28. დავწეროთ პროგრამა, რომელიც შეავსებს სიმბოლურ მასივს შეავსებს შემდეგნაირად: მთავარ დიაგონალზე ჩავსვათ სიმბოლო - a, მთავარი დიაგონალის ზემოთ ჩავსვათ სიმბოლო - b, მთავარი დიაგონალის ქვემოთ ჩავსვათ სიმბოლო - c.

```
#include <iostream.h>
#include <conio.h>
#define n 5
main()
{ char a[n][n];
int i,j;
```

```
for(i=0; i<n; i++)
for(j=0; j<n; j++)
{if(i==j) a[i][j]='a';
if(i<j) a[i][j]='b';
if(i>j) a[i][j]='c';
}
cout<<"shevsebuli masivi \n";
 for(i=0;i<n;i++)
 {for(j=0;j<n;j++)
 cout<<a[i][j]<<" ";
 cout<<endl;
 }
 getch();
 return 0;
}
```

ამოცანა 29. დავწეროთ პროგრამა, რომელიც სიმბოლურ მასივში გაარკვევს, მეორდება თუ არა რომელიმე სიმბოლოები მასივში.

```
#include<iostream.h>
#include<conio.h>
bool ganmeoreba(char a[],int n)
{ int i,j;
for(i=0;i<n;i++)
for(j=i+1;j<n;j++)
if(a[i]==a[j]) return true;
return false;
}
main()
{char a[5];
int i; bool k;
for(i=0;i<5;i++)
cin>>a[i];
k=ganmeoreba (a,5);
if(k==true)cout<<"ganmeorda";
```

```
else cout<<"ar ganmeorda";
getch();
}
```

ამოცანა 30. დავწეროთ პროგრამა, რომელიც გაარკვევს, თუ რამდენი სიტყვისგან შედგება სიმბოლურ მასივში ჩაწერილი წინადადება.

```
#include <iostream.h>
#include <conio.h>
int sitkvebi(char a[],int n)
{int i,k=0;
for(i=0;i<n;i++)
 if (a[i]==' ') k++;
k=k+1;
cout<<k;
return k;}
main()
{char a[10]; int i,p;
cout<<"shemovitanot winadadeba";
for(i=0;i<10;i++)
cin>>a[i];
p=sitkvebi(a,10);
cout<<"sitkvebis raodenoba"<<p;
getch();
}
```

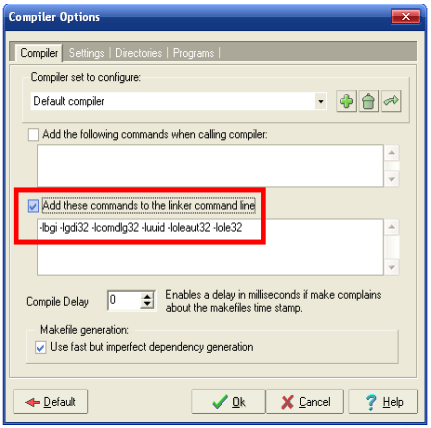
5. გრაფიკულ რეჟიმში მუშაობის ძირითადი საშუალებები

გრაფიკულ რეჯიმში სამუშაოდ აუცილებელია Dev C++ ჩართული გვქონდეს შესაბამისი მოდული, ამისათვის შემდეგნაირად უნდა მოვიქცეთ:

1. ფაილი grapics.h გადმოვიწეროთ და მოვათავსოთ C:\Dev-Cpp\include საქაღალდეში;
2. ფაილი Libbgi.a გადმოვწეროთ და შევინახოთ C:\Dev-Cpp\Lib საქაღალდეში;

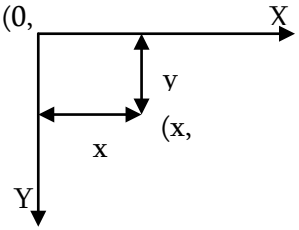
3. გავუშვათ პროგრამა Dev-C++ და Tools მენიუდან გამოვიძახოთ Compiler Options.

4. ჩანართში Compiler ჩავრთოთ ალამი Add these commands to the linker command line შესაბამის ველში ჩავწეროთ **`-lbgi -lgdi32 -lcomdlg32 -luuid -loleaut32 -lole32`**



ამ ინსტრუქციების შესრულების შემდეგ თქვენ შეგიძლიათ გამოიყენოთ ბრძანებები (ოპერატორები) გრაფიკული პროგრამების შესაქმნელად.

სანამ ეკრანზე რაიმეს დავხატავთ, უნდა ვიცოდეთ როგორაა წარმოდგენილი კოორდინატები.

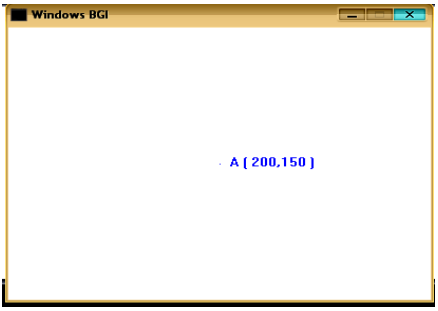


- საწყისი კოორდინატი, წერტილი(0,0), არის ეკრანის მარცხენა ზედა კუთხეში.
- X წრფე მიმართულია მარჯვნისაკენ, Y წრფე კი - ქვევით(მათემატიკური კოოტდინატთა სისტემისაგან განსხვავებით)
- წერტილის ნებისმიერი კოორდინატა X -არის მანძილი ეკრანის მარჯვენა კიდემდე, ხოლო Y-მანძილი მარცხენა კიდემდე.

16 სტანდარტული ფერის გამოსაყენებლად მიღებულია რიცხვითი და სიმბოლური აღნიშვნები

0	**BLACK**	**8**	**DARKGRAY**
1	**BLUE**	**9**	**LIGHTBLUE**
2	**GREEN**	**10**	**LIGHTGREEN**
3	**CYAN**	**11**	**LIGHTCYAN**
4	**RED**	**12**	**LIGHTRED**
5	**MAGENTA**	**13**	**LIGHTMAGENTA**
6	**BROWN**	**14**	**YELLOW**
7	**LIGHTGRAY**	**15**	**WHITE**

ამოცანა 31. გავხსნათ გრაფიკულ რეჟიმში ფანჯარა ზომით 400X300 პიქსელი და აღვნიშნოთ წერტილი შესაბამისი კოორდინატებით A(200,150).




```
#include <graphics.h>
#include <conio.h>
main()
{
initwindow ( 400, 300 );
/*ხსნის გრაფიკულ
რეჟიმში ფანჯარას
ზომით 400X300 პიქსელი
*/
setcolor (14);// რთავს ყვითელ ფერს
outtextxy(210,140,"A ( 200,150 )" ); //აღნიშნულ მისამართზე წერს
//შესაბამის ტექსტს
putpixel ( 200, 150, 14 ); //აღნიშნულ მისამართზე ჩნდება
//ყვითელი წერტილი(პიქსელი)
getch();
closegraph(); //იხურება გრაფიკულ რეჟიმში მომუშავე ფანჯარა
}
```

ამოცანა 32. გავხსნათ გრაფიკულ რეჟიმში ფანჯარა ზომით 400X300 პიქსელი და დავხაზოთ ერთი მონაკვეთი, რომლის საწყისი კოორდინატი იქნება moveto(40, 30) ხოლო მეორე ბოლოს -

lineto(200, 200) და მისი პარალელური მონაკვეთი line(140,30,300, 200)

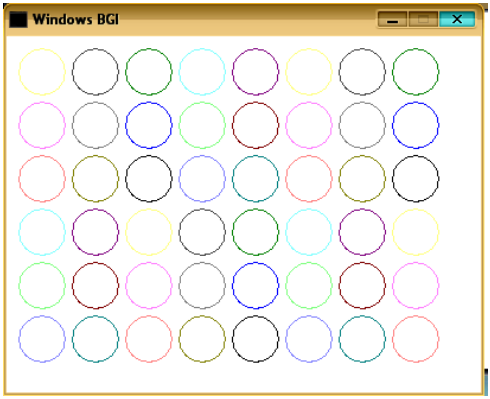
```
#include <graphics.h>
#include <conio.h>
int main(void)
{
initwindow ( 400, 300 );
setcolor (12); //ირთვება ღია წითელი ფერი
moveto(40, 30); //აფიქსირებს საწყის კოორდინატს
outtextxy(45,10,"( 40,30 )" ); //აღნიშნულ მისამართზე წერს
//მითითებულ ტექსტს
lineto(200, 200); //წერტილიდან (40,30) ხაზავს მონაკვეთს
//(200,200)
outtextxy(205,190,"( 200,200 )" );
setcolor (2); // ირთვება მწვანე ფერი
line(140,30,300, 200); //ხაზავს მონაკვეთს, რომლის საწყისი
//წერტილია(140,30) ხოლო ბოლო -(300,200)
getch();
closegraph();
return 0;
}
```

ამოცანა 33. ფანჯარაში ზომით 600X400 ცენტრში დავხაზოთ, წრეწირი,ელიფსი და რკალი.

```
#include <graphics.h>
#include <conio.h>
int main(void)
{
/* ფანჯრის ზომა */
initwindow ( 600, 400 );
//ცვლადების გამოცხადება და ინიციალიზაცია
int midx, midy;
int stangle = 0, endangle =180;
int xradius = 10, yradius = 50;
```

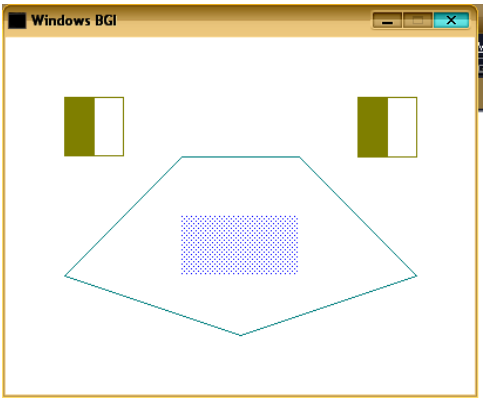
```
int radius = 100;
//შუაწერტილის პოვნა
midx = getmaxx() / 2;  //x  ღერძის შუაწერტილის პოვნა
midy = getmaxy() / 2; //y  ღერძის შუაწერტილის პოვნა
//ვხატავთ მიმდევრობით სამ გეომეტრიულ ფიგურას
setcolor (4);
ellipse(midx, midy, stangle, 2*endangle, xradius, yradius);
 /* ელიფსი */
setcolor (5);
arc(midx, midy, stangle, endangle, radius);  /* რკალი */

setcolor (6);
circle (midx, midy,radius/2);  /* წრეწირი */
getch();
closegraph();
return 0;
}
```

ამოცანა 34. ფანჯარა ზომით 400X300 პიქსელი შეივსოს ფერადი წრეწირებით.




```
#include
<graphics.h>
#include <conio.h>
int main(void)
{int i=1;
initwindow ( 400,
300 );
for(int x=30;
x<380;x=x+45)
for(int y=30;
y<280;y=y+45)
{if (i>15)i=1;  setcolor (i++);circle ( x, y, 20 );}
getch();
closegraph();
return 0;
}
```

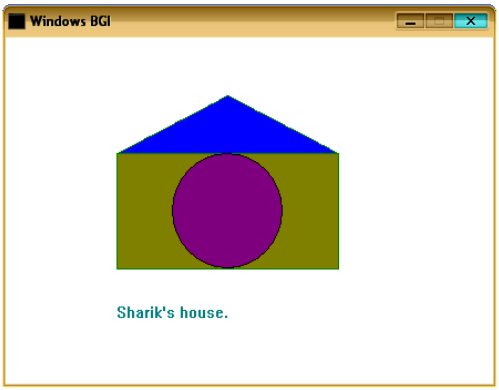
ამოცანა 35. გავხსნათ გრაფიკულ რეჟიმში ფანჯარა ზომით 400X300 პიქსელი და დავხაზოთ




```
#include <graphics.h>
#include <conio.h>
int main()
{
initwindow ( 400, 300
);
setcolor(12);
moveto (150, 100);
lineto (250, 100);
lineto (350, 200);
lineto (200, 250);
lineto (50, 200);
lineto (150, 100);
setfillstyle ( 11, 14 );
bar (150, 150, 250, 200);
setcolor ( 9 );
rectangle (75, 50, 100, 99);
setfillstyle ( 1, 9 );
bar (50, 50, 75, 100);
rectangle (325, 50, 350, 100);
setfillstyle ( 1, 9 );
bar (300, 50, 325, 101);
getch();
closegraph();
return 0;
}
```

ამოცანა 36. ფანჯარაში ზომით 440 X300 დავხატოთ სახლი და გავუკეთოთ წარწერა ” Sharik's house ”.

```
#include <graphics.h>
#include <conio.h>
main()
{
initwindow (440, 300);
// ვხატავთ  ცისფერ
//მართკუთხედს
//იასამნისფერი
//ჩარჩოთი
setfillstyle (1, 9);
bar (100,100,300,200);
setcolor (13);
/*ვხატავთ ყვითელ სამკუთხედს იასამნისფერი ჩარჩოთი*/
rectangle (100,100,300,200);
moveto (100,100);
lineto (200, 50);
lineto (300,100);
/*ვხატავთ მწვანე წრეს თეთრი ჩარჩოთი*/
setfillstyle (1, 14);
floodfill (200, 75, 13);
setcolor (15);
circle (200, 150,50);
setfillstyle (1, 10);
floodfill (200,150, 15);
setcolor (12);
//გეომეტრიული ფიგურების ქვეშ ვარდისფრად ვწერთ შემდეგ
//ტექსტს Sharik's house
outtextxy (100, 230, "Sharik's house.");
getch();
closegraph();
}
```

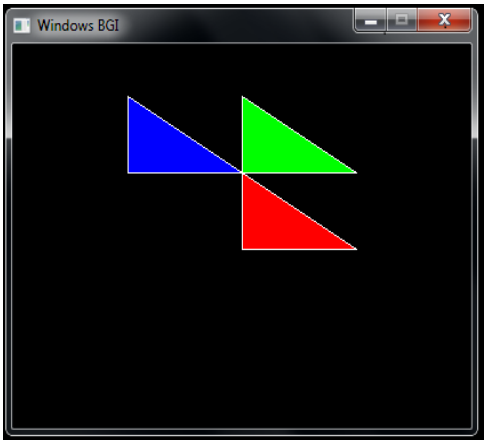


COLOR(0,0,0) შავი
COLOR(255, 0, 0) წითელი
COLOR(0, 255, 0) მწვანე
COLOR(0, 0, 255) ლურჯი

COLOR(255, 255, 255) თეთრი
COLOR(100, 100, 100) ნაცრისფერი
COLOR(255, 0, 255) იისფერი
COLOR(255, 255, 0) ყვითელი

ამოცანა 37. დავხაზოთ სამკუთხედები:

```
#include <conio.h>
#include <graphics.h>
/* ფუნქცია samkutxedi  */
void samkutxedi ( int x, int y, int c )
{
moveto ( x, y ); // ვუთითებთ კურსორის კოორდინატებს
/*ვხაზავთ კონტურს*/
lineto ( x, y-60 );
lineto ( x+100, y );
lineto ( x, y );
//მიღებულ სამკუთხედს
//საზღვრამდე  ვღებავთ
setfillstyle ( 1, c );
floodfill ( x+20, y-20, 15);
}
//ძირითადი ფუნქციის
//დასაწყისი
main()
{
initwindow (400, 300);
/*მივმართავთ ფუნქციას  samkutxedi  3-ჯერ */
samkutxedi (100, 100, COLOR(0,0,255));
samkutxedi (200, 100, COLOR(0,255,0));
samkutxedi (200, 160, COLOR(255,0,0));
getch();
closegraph();
}
```



ახლა განვიხილოთ მარტივი ანიმაციის მაგალითები:

ამოცანა 38. დავხატოთ კვადრატი და ვამოძრაოთ ვერტიკალური მიმართულებით.

```
#include <graphics.h>
#include <conio.h>
/* draw ფუნქცია ხატავს კვადრატს*/
void draw( int x, int y, int color )
{
setfillstyle(1, color);
bar (x,y,x+10,y+10);
}
main()
{
int x,y;
initwindow(400,400);
// ფანჯარას  ზომით 400 X400 პიქსელი  ვღებავთ ლურჯ
//ფერში
setfillstyle(1, COLOR(0,0,255));
bar(0,0,399,399);
x=50; y=400;
while(x+5<200)
{
//მივმართავთ draw ფუნქციას ყვითელი ფერის კვადრატის
//დასახატად
draw(x,y,COLOR(255,255,0));
delay(50);//ვაყოვნებთ
//მივმართავთ draw ფუნქციას ლურჯი ფერის კვადრატის
//დასახატად
draw(x,y,COLOR(0,0,255));
y--; //მოძრაობის ეფექტი იქნება ფანჯრის ქვედა საზღვრიდან
//ზედა საზღვრამდე
}
getch();
closegraph();
}
```

ამოცანა 39. ეკრანზე შემთხვევითი ფერადი წერტილების საშუალებით შევქმნათ ციმციმის ეფექტი

```
#include <graphics.h>
#include <conio.h>
#include <stdlib.h>
int random (int N) { return rand() % N; }
// შემთხვევითი რიცხვის პოვნის ფუნქცია
main()
{
int x, y, R, G, B;
initwindow (400, 300);
while ( ! kbhit () ) { // სანამ კლავიშს დავაწვებით
/*შემთხვევითი კოორდინატები*/
x = random(400);
y = random(300);
/*შემთხვევითი ფერი (R,G,B)*/
R = random(256);
G = random(256);
B = random(256);
if ( getpixel(x,y) != 0 ) /* თუ წერტილი შავი ფერის არაა*/
putpixel ( x, y, 0 ); /* ვღებავთ შავ ფერში*/
else  putpixel ( x, y, COLOR(R,G,B) ); /*შემთხვევითი ფერი*/
}
getch();
closegraph();
}
```

6. ფაილებთან მუშაობა.

ცნობილია, რომ პროგრამული ტექნოლოგიები საკმაოდ სწრაფად ვითარდება. ამასთან დაკავშირებით არსებობს დიდი მოცულობის სხვადასხვა ტიპის მონაცემების დამუშავება-გადაგზავნის საჭიროება. ჩვეულებრივ, მონაცემები ინახება ფაილების სახით. თანაც თითქმის ყველა ტიპის მონაცემი შესაძლებელია ფაილური სტრუქტურის მატარებელი იყოს.

ფაილური სისტემები ჯერ კიდევ MS DOS-ის აღმაფრენის წლებში ფართოდ იყო გავრცელებული. ფაილის ცნება არ შეცვლილა FAT32 და NTFS სისტემების შემოღების შემდეგაც (სწორედ ამ სისტემებს ემყარება windows 98, windows 2000 და ა.შ.) არ შეცვლილა ფაილებზე მიმართვის მეთოდებიც.

ჩვენ შევისწავლით C++-ზე ფაილებთან მუშაობის ძირითად შესაძლებლობებს. ერთის მხრივ, ფიზიკური ფაილი წარმოადგენს ბაიტების თანმიმდევრობას, რომელსაც გააჩნია უნიკალური სახელი, ტიპი, ზომები. ის მოთავსებულია ინფორმაციის რაიმე მატარებელზე (მაგ., მაგნიტურ დისკზე). მეორე მხრივ, ფაილი წარმოადგენს მონაცემთა სტრუქტურას, რომელიც შედგება ერთნაირი ტიპის ელემენტების თანმიმდევრობისგან, თანაც ფაილის ზომები პრაქტიკულად შეზღუდული არ არის. ფაილი და მასივი გარკვეული თვისებებით ჰგავს ერთმანეთს, თუმცა არსებობს რიგი განსხვავებები. მასივი ყოველთვის ოპერატიულ მეხსიერებაშია ჩაწერილი. მისი ზომები წინაწარ არის გამოცხადებული და პროგრამის მუშაობის პროცესში ეს ზომები არ იცვლება (იგულისხმება ჩვეულებრივი აქამდე განხილული მასივის ტიპი). მასივი, ისევე როგორც ფაილი, შედგება ერთნაირი ტიპის ელემენტებისაგან. ფაილის ზომები წინასწარ განსაზღვრული არ არის. ფაილი ჩვეულებრივ იწერება ინფორმაციის მუდმივ მატარებელზე. როგორც უკვე ვთქვით, ფაილიც შედგება ერთნაირი ტიპის ელემენტებისაგან, რომლებიც თანმიმდევრულადაა განლაგებული. პროგრამის მუშაობის შედეგად ფაილის ზომები შეიძლება შეიცვალოს. პირველადი წარმოდგენით, ფაილი შეიძლება შევადაროთ მასივს, რომელიც ზომებში არ არის შეზღუდული. ყველა ფაილს გააჩნია თავისი სახელი და ტიპი, ამიტომ ერთ პროგრამაში შეიძლება ერთდროულად რამდენიმე ფაილთან მუშაობა. ფაილი შეიძლება შეიცავდეს ნებისმიერი ტიპის მონაცემებს, მაგრამ ფაილის ელემენტებად არ შეიძლება გამოვიყენოთ ფაილები. იმისათვის, რომ პროგრამაში გვქონდეს ფაილებთამ მუშაობის შესაძლებლობა, უნდა ჩავრთოთ #include <fstream.h>. იმისათვის, რომ პროგრამაში ფაილი გამოვიყენოთ ფაილი, აუცილებელია , ისევე როგორც ნებისმიერი სხვა ცვლადის, გამოცხადება. მაგალითად, C++-ზე ფაილის გამოცხადება ხდება შემდეგნაირად :

ofstream filename(“c:/ input.txt”, ios::app);
fstream a(“d:/b.txt”);
ბრჭყალებში მიუთითებენ ფიზიკური ფაილის სახელს, გაფართოებას, ადგილმდებარეობას. Filename წარმოადგენს იგივე ფაილის სახელს C++ პროგრამაში, ანუ ეს არის სახელი, რომლითაც მივმართავთ იგივე ფიზიკურ ფაილს, ოღონდ პროგრამის ტექსტში. თუ ფაილი არ არსებობს აღნიშნულ ადგილას, მაშინ ახალი ფაილი შეიქმნება. მძიმის შემდეგ მითითებულია ფაილზე მიართვის რეჟიმი. მოვიყვანოთ ფაილზე მიმართვის სხვადასხვა რეჟიმები (ios აღნიშნავს შემოტანა-გამოტანის ნაკადს):
ფაილებზე მიმართვისათვის გამოიყენება შემდეგი რეჟიმები:

ios :: in - წაკითხვა ფაილიდან. ეს რეჟიმი ifstream ნაკადისათვის ჩართულია ავტომატურად (”დუმილით”). კურსორი მიუთითებს ფაილის დასაწყისს ”დუმილით”. (ფაილის წაკითხვა ხდება პირველი ელემენტიდან).

ios :: out - ჩაიწერება ახალი ინფორმაცია ზემოდან უკვე არსებულზე, ofstream ნაკადისათვის ჩართულია ”დუმილით”, ავტომატურად. კურსორი მიუთითებს ფაილის დასაწყისს ”დუმილით”.

ios :: ate - ჩაწერისა და წაკითხვის რეჟიმი. ჩაწერა შესაძლებელია უკვე არსებულ ინფორმაციაზე. კურსორი მიუთითებს ფაილის დასასრულს ”დუმილით”.

ios :: app - ჩაწერის რეჟიმი, ხდება ინფორმაციის დამატება ფაილის დასასრულში უკვე არსებული ჩანაწერების შემდეგ.

ifstream filename (“input.txt”) - გახსნის input.txt ფიზიკურ ფაილს წაკითხვის რეჟიმში . პროგრამაში ამ ფაილს მივმართავთ filename სახელით. არგუმენტი ios :: in შეიძლება გამოვტოვოთ, რადგან ეს ნაკადი იქნება ”დუმილით” ჩართული. იმისათვის, რომ ფაილი გავხსნათ როგორც წაკითვის, ისე ჩაწერის რეჟიმში შეიძლება გამოვიყენოთ ბრძანება:
fstream filename(“input.txt”, ios ::in | ios ::out);

ამოცანა 40. განვიხილოთ თანმიმდევრული წვდომის ფაილში ინფორმაციის ჩაწერა და შემდეგ მისი ამობეჭდვა:

```
#include <iostream.h>
```

```
#include <conio.h>
#include <fstream.h>
main()
{
char num[10], name[10];
ofstream output("teona.txt", ios ::app);
cin>>num>>name;
cout<<"monacemebi shetanilia, velodebi axals"<<endl;
output<<num<<"  "<<name<<endl;
output.close();
getch();  }
```

იმისათვის, რომ უკვე არსებული ფაილიდან წავიკითხოთ ინფორმაცია და დავბეჭდოთ ის, ეკრენზე უნდა გამოვიყენოთ შემდეგი სახის პროგრამა. (ფაილი უნდა იყოს უკვე შექმნილი პროგრამულად ან Notepad-ის მეშვეობით).

ამოცანა 41. დავწეროთ პროგრამა, რომელიც უკვე არსებული ფაილიდან კითხულობს ინფორმაციას და დაბეჭდავს მას.

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
main()
{
char num[10], name[10];
ifstream input ("teona.txt");
while (!eof())
{input>>num>>name;
cout<<num<<"   "<<name<<endl;}
input.close();
getch();
}
```

ამოცანა 42. დავწეროთ პროგრამა, რომელიც უკვე შექმნილ ფაილში განსაზღვრავს, თუ რამდენჯერ გვხვდება კლავიატურიდან შეტანილი სიმბოლო ფაილში მოთავსებულ ტექსტში.

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
main()
{
char boofer,k; int s=0;
ifstream input (“teona.txt”);
cin>>k;
while (!eof())
{ input.get(boofer);
if (boofer==k) s++;}
input.close();
cout<<s;}
getch();
}
```

ამოცანა 43. დავწეროთ პროგრამა, რომელიც წინასწარ გამზადებული ფაილიდან ამოიღებს მატრიცის ელემენტებს და დაითვლის ამ ელემენტების ჯამს თითოეული სტრიქონისათვის.

პროგრამაში არ არის გამოყენებული მასივი, მაგრამ ფაილი აუცილებლად უნდა შეიცავდეს სტრიქონებისა და სვეტების რაოდენობას. თითოეული სტრიქონის ჯამი მოვათავსოთ მეორე ფაილში. ყოველი ჯამი ახალ სტრიქონშია ჩაწერილი.

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
main()
{ int n, m, i, j, boofer, sts=0;
ifstream input (“matrica.txt”);
ofstream output (“jami.txt”);
input>>n>>m;
for (i=0; i<n; i++)
{for (j=0; j<m; j++)
{input>>boofer;
sts=sts+boofer;}
```

```
output<<sts<<endl;
sts=0;}
input.close ( );
output.close ( );
getch();
return 0;
}
```

პროგრამაში გამოყენებულია ორი ფაილი. აქედან პირველი უნდა შეიქმნას სხვა პროგრამის მეშვეობით ან Notpad-ში. ამ ფაილში უნდა იყოს ჩაწერილი პირველი ორი რიცხვი: მატრიცის სტრიქონებისა და სვეტების რაოდენობა, შემდეგ - შესაბამისი რაოდენობის (mxn-ზე) ელემენტები. პირველი ფაილი იხსნება წასაკითხად, boofer ცვლადის მეშვეობით ვკითხულობთ მიმდინარე ელემენტს და ვამატებთ მას sts ჯამში. ერთი სტრიქონის ელემენტების აჯამვის შემდეგ მეორე ფაილში ჩავწერთ ჯამს და endl-ით გადავდივართ ახალ სტრიქონზე. ყოველი სტრიქონის ჯამის დათვლის შემდეგ sts უნდა გაავანულოთ, წინააღმდეგ შემთხვევაში პროგრამა დაითვლის ყველა ელემენტის ჯამს. შემდეგ ორივე ფაილი უნდა დავხუროთ.

ამოცანა 44. ვთქვათ, ფაილში ჩაწერილია მონაცემები: წყვილების რაოდენობა და რიცხვითი წყვილები. დავწეროთ პროგრამა, რომელიც განსაზღვრავს ამ რიცხვების წყვილებისათვის, არიან თუ არა ისინი ურთიერთმარტივნი და ყოველი წყვილისათვის შესაბამის ინფორმაციას ჩაწერს მეორე ფაილში.

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
int usg (int a, int b)
{ if (a>b) a=a-b; else b=b-a;
if ( b==0) return a;
usg(a,b); }
main()
{
```

```
int k, l, n, i, v;
ifstream input (“kristi.txt”);
ofstream output (“martivi.txt”);
input>>n;
for (i=0; i<n; i++)
{
input>>k>>l;
v=usg(k,l);
output<<k<<”  ”<<l;
if (v==1) output<<”urtiertmartivia”;
else output<<”urtiertmartivi ar aris”<<endl;
}
cout<<”dasrulda, nebismier klavishs daachiret”;
input.close ();
output.close();
getch();
return 0;}
```

ამოცანა 45. დავწეროთ პროგრამა, რომელიც მოცემულ ფაილში იპოვის დადებით რიცხვებს შორის იპოვის უდიდესს და ჩაწერს მეორე ფაილში. (აქ მასივს არ გამოვიყენებთ).

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
main()
{int n, max;
ifstream input (“shako.txt”);
ofstream output (“shako1.txt”);
max=0;
while(! input.eof())
{input>>n;
if (n>max) max=n;}
output<<”mocemul ricxvebs shoris udidesia”<max;
input.close ( );
output.close ( );
```

```
getch();
return 0;
}
```

ამოცანა 46. ვთქვათ მოცემულია ფაილი, რომელშიც ჩაწერილია რიცხვები, დავწეროთ პროგრამა, რომელიც მეორე ფაილში გადაიტანს მხოლოდ ლუწებს.

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
main()
{
int n;
ifstream input (“lika.txt”);
ofstream output(“lutsi.txt”);
while ( !input.eof() )
{input>>n;
if  (n%2==0)  output<<n;}
input.close( );
output.close( );
getch();
return 0; }
```

ამოცანა 47. ვთქვათ, მოცემულია ფაილი, რომელშიც ჩაწერილია რიცხვთა მიმდევრობა. დავწეროთ პროგრამა, რომელიც დაითვლის ამ რიცხვთა მიმდევრობისათვის მხოლოდ არანულოვან ელემენტთა ნამრავლს.

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
main()
{
int n, p=1;
ifstream input (“ricxvebi.txt”);
```

```
ofstream output(“namravli.txt”);
while ( !input.eof())
{ input>>n;
if (n!=0) p=p*n;}
output<<”aranulovan elementta namravlia”<<p<<endl;
input.close();
output.close();
getch();
return 0;
}
```

ამოცანა 48. დავწეროთ პროგრამა, რომელიც ფაილიდან წაიკითხავს რიცხვებს, თითოეული რიცხვისათვის იპოვის მის კვადრატს და ჩაწერს მეორე ფაილში.

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
main()
{ int n, k;
ifstream input(“ricxvebi.txt”);
ofstream output (“xarisxi.txt”);
while( !input.eof())
{input>>n;
k=n*n;
output<<n<<”ricxvis kvadratia”<<k<<endl;}
input.close();
output.close();
getch();
return 0;
}
```

ამოცანა 49. დავწეროთ პროგრამა, რომელიც ფაილიდან კითხულობს რიცხვებს და ამ რიცხვების საშუალო არითმეტიკულს ჩაწერს მეორე ფაილში.

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
main()
{float s; int p=0, k=0, n;
ifstream input("ricxvebi.txt");
ofstream output("sashualo.txt");
while( !input.eof())
{input>>n;  k++;  p=p+n;}
s=p/k;
output<<"mecemuli ricxvebis sashualo aritmetikulia"<<s<<endl;
input.close( );
output.close( );
getch();}
```

ამოცანა 50. დავწეროთ პროგრამა, რომელიც ფაილიდან კითხულობს რიცხვებს, დაადგენს თითოეული რიცხვისათვის ლუწია, კენტია თუ ნულის ტოლია, დაითვლის მათ რაოდენობებს და მხოლოდ კენტ რიცხვებს ჩაწერს მეორე ფაილში.

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
main()
{int m, l, k, n; l=k=n=0;
ifstream input ("lika.txt");
ofstream output("nino.txt");
while( !input.eof())
{input>>m;
if (m==0) m++;
if (m%2==0) l++;
else {k++; output<<m<<" ";}
}
input.close( );
output.close( );
getch();}
```

ამოცანა 51. მოცემულია მთელი რიცხვების ფაილი. იპოვეთ კენტი რიცხვების საშუალო არითმეტიკული.

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
float kentebi (ifstream &x)
{float s=0; int k=0, p;
while (y.eof()==false)
{y>>p;
if (p>0 && p%2==1) {k++; s=s+p;}
}
if (k!=0)  return   s/k;
else return -1;}
main()
{ ifstream x; float k;
x.open (“c:/t.txt”);
if (x.is_open( )==false)
{cout<<”fizikuri faili ar arsebobs”; exit(0);}
k=kentebi(x);
if (k==-1)  cout<<”ar iyo dadebiti kenti ricxvi”;
else cout<<k;
getch();
}
```

ამოცანა 52. მოცემულია მთელი რიცხვების ფაილი, დაწერეთ პროგრამა, რომელიც შექმნის მეორე ფაილს, სადაც პირველიდან იქნება მხოლოდ უარყოფითი ელემენტები გადმოწერილი.

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
void uaryofiti(ifstream &y)
{ ofstream z; int a;
open (“k.txt”);
```

```
while (y.eof( )==false)
{y>>a;
if (a<0)  z<<a;}
z.close ( );
y.close ( ); }
main ( )
{ ifstream x;
x.open (“t.txt”);
if (x.is_open( )==false)
{ cout<<”faili ar arsebobs”; exit(0);}
uaryofiti(x);
getch();
}
```

ამოცანა 53. მოცემულია მთელი ტიპის ფაილი. დავწეროთ პროგრამა, რომელიც შეიცავს ფუნქციას. ფუნქციამ უნდა გაარკვიოს, არის თუ არა ფაილის შიგთავსი ფიბონაჩის მიმდევრობა?

```
#include <iostream.h>
#include <conio.h>
#include <fstream.h>
bool  fibonachi (ifstream &x)
{ int a, b, c;
x>>a;  x>>b;
if ((a!=1 || b!=1) && (a!=0 || b!=1))
return false;
else
{ while (x.eof( ) == false)
{x>>c;
if  (c!=a+b) return false;
a=b; b=c;}
return true;}
}
main ( )
{ ifstream x; bool k;
```

```
x.open (“t.txt”);
if (x.is_open()==false) { cout<<”fizikuri faili ar arsebobs”;  exit(0); }
k=fibonachi(x);
if (k==true) cout<<”fibonachis mimdevrobaa”;
   else cout<<”ar aris fibonachis mimdevroba”;
getch();
}
```

ლიტერატურა

1. C++ How to Program, 6/e, Harvey M. Deitel and Paul J.Deitel, both from Deitel & Associates, Inc. © 2008, 1500 pp., paper(0-13-615250-3)
2. Информатика, базовый курс, под редакцией С.В.Симоновича, изво Питер, Санкт-Петербург. 2002г.
3. Э.А. Ишкова, С++, учебник, 1999.
4. Ю. Федоренко, алгоритмы и программы, учебный курс, Питер, 2001 г.
5. Т.А. Павловская, Ю.А. Щупак, С/С++ программирование на языке высокого уровня, структурное программирование, Питер, Санкт-Петербург.2003.
6. М.Динман, С++ освой на примерах, БХВ, Санкт-Петербург. 2002.
7. С.А.Немнюгин, программирование, учебник, Питер, Санкт-Петербург. 2000.

შინაარსი