6. Įvedimo, išvedimo srautai

6.1. Išvedimas ekrane ir įvedimas klaviatūra.

Įvedimo srautų klasė `istream` ir išvedimo srautų klasė `ostream` aprašytos faile `iostream.h`. Jų objektai `cin` ir `cout` palaiko klasėse numatytus veiksmus.

Anksčiau buvo nagrinėjami operatoriai >> ir <<.

Išvedamų skaičių formatui valdyti buvo panaudotos faile `iomanip`.h esančios priemonės `setw( int )` ir `setprecission( int ).`

🖬 6.1 pratimas. Nuoroda `setw` galioja artimiausiam dydžiui, o `setprecission` iki naujo nurodymo. Demonstracinės programos rezultatas bus toks:

```
A=    18.236
A=18.2
A=     18.236
A=123.46
```

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

main(){
  float A = 18.236;
  cout<<"A=" << setw(9) << A << endl;
  cout<<"A=" << setprecision(3) << A << endl;
  cout<<"A="
      << setw(10) << setprecision(5)  << A <<endl;
  A = 123.45678;
  cout<<"A=" << A <<endl;
  getch();
 }
```

Klasės ostream metodas with(int) skirtas nurodyti išvedamo sraute duomenų laukams. Galioja artimiausiam dydžiui. Tai setw analogas.

🖬 6.2 pratimas. Parodomas duomenų lauko valdymo metodas.

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

main(){  int i;
   for ( i = 2; i < 6; i++){
 cout.width( 3 );     cout<<"i=";
 cout.width( i );     cout<< i <<endl; }
 getch();
}
```

Ekrane matysime:	i= 2 i= 3 i= 4 i= 5

Išvedant duomenis nurodytu formatu nepanaudotos pozicijos užpildomos tarpo simboliais. Metodas `fill( int )` leidžia pakeisti užpildymo simbolį norimu. Pakeitimas galioja iki naujo nurodymo.

🖬 6.3 pratimas. Demonstruojamas duomenų laukų užpildymas simboliais.

	Ekrane matysime:

`#include <iostream.h>`
`#include <iomanip.h>`
`#include <conio.h>`
 	
`main(){`
`int i;`
`cout.fill( '.');`
`for ( i = 2; i < 6; i++){`
`cout<<"i=";`
`cout.width( i );`
`cout<< i <<endl; }`
`getch();`
`}`
	`i=.2`
	`i=..3`
	`i=...4`
	`i=....5`

🖬 6.4 pratimas. Metodas `put` skirtas vienam simboliui išvesti ekrane.

	Ekrane matysime

`#include <iostream.h>`
`#include <conio.h>`
 	
`main(){`
	`Katinas`

```
  int i;
  char zodis[] = "Katinas";
  for ( i = 0; zodis[i]; i++)
    cout.put( zodis[ i ]);
    cout<< endl;
  getch();
 }
```

6.5 pratimas.

Metodas `put` programose dažnai vartojamas kartu su funkcija `int toupper ( int )`, kuri mžąją raidę keičia didžiaja. Kiti simboliai nekinta.

	Ekrane matysime

`#include <iostream.h>`
`#include <conio.h>`
`#include <ctype.h>`
 	
`main(){`
`int i;  char sim;`
`char zodis[] = "Katinas ir Pele";`
`  for ( i = 0; zodis[i]; i++) {`
`    sim = toupper( zodis[ i ]);`
`    cout.put( sim );  }`
`    cout<< endl << zodis << endl;`
`  getch();`
`}`
	`KATINAS IR PELE`
	`Katinas ir Pele`

6.6 pratimas.

Įvedimo klasėje metodas `get` skirtas vieno simbolio įvedimui klaviatūra.

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

main(){  char sim;
cout<<" Ar dar dirbsite?( T/N ):";
do {  sim = cin.get();
      sim = toupper( sim ); }
while (( sim != 'T' ) && ( sim != 'N' ));
cout<< "Jus pasirinkote : "<< sim << endl;
getch();
  }
```

Ekrane matysime:	`Ar dar dirbsite?( T/N ):`
	`t`
	`Jus pasirinkote : T`

6.7 pratimas.

Įvedimo klasėje metodas `getline` skirtas vienos eilutės įvedimui klaviatūra. Matome, kad paprastai įvedama eilutė iki pirmojo tarpo simbolio arba iki galo, jeigu tarpo simbolio nebuvo. Panudoję `getline` metodą galima įvesti norimą simbolių skaičių: nurodomas skaičius. Jeigu tiek simbolių nebuvo, ivedami visi simboliai iki eilutės galo. Įvedama eilutė visuomet papildoma nuliniu simboliu (eilutės galas).

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

 main(){
  char A[30];  int n;
  cout<<"Iveskite eilute:\n";
  cin>> A;                  // Pirmasis eilutės žodis.
  cout<<"Eilute: *"<< A <<"*\n";
cin.getline( A, 12 );//Vienuolika simbolių ir '\0'.
cout<<"Eilute: *"<< A <<"*\n";
                    // Iki galo arba tiek, kiek telpa.
  cin.getline( A, sizeof( A ));
  cout<<"Eilute: *"<< A << "*\n";
  n = cin.gcount(); // Eilutės ilgis su nuliniu simboliu.
  cout<< "n= "<< n << endl;
  getch();
 }
```

Ekrane matysime:	`Iveskite eilute:`
	`Rainas Katinas ir Pilka Pele`
	`Eilute: *Rainas*`
	`Eilute: * Katinas ir*`
	`Eilute: * Pilka Pele*`
	`n=12`

6.8 pratimas. Įvedimo klasėje metodas `getline` gali turėti trečią parametrą, nurodantį simbolį, kuriuo baigiamas įvedimas. Pavyzdyje parašyta raidė 'Z'. Jeigu įvedamoje eilutėje jos nebus,tuomet bus įvedami visi eilutės simboliai arba tik tiek, kie nurodyta, jeigu eilutėje jų yra daugiau.

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

 main(){
  char A[30];
  cout<<"Iveskite eilute:\n";
  cin.getline( A, sizeof( A ), 'Z' );
  cout<<"Eilute: *"<< A << "*\n";
  getch();
 }
```

6.2. Įvedimas iš failo ir išvedimas į failą.

Įvedimo srautu iš failo klasė `ifstream` ir išvedimo srautu klasė `ofstream` į failą aprašytos faile `fstream.h`. Galima sukurti objektus, kurie nukreipia įvedimo\išvedimo srautus iš\į failus. Jiems atitinkamai galioja operatoriai >> ir <<. Konstruktoriaus parametru yra failo vardas. Čia yra metodai, kuriuos turi jau nagrinėtos srautų klasės. Failo uždarymui yra metodas `close()`.

6.9 pratimas. Duomenų faile "Duom.dat" yra skaičius 15. Rezultatų faile "Rez.rez" rasime skaičių 15.

	Ekrane	Rezultatų faile

`#include <iostream.h>`
`#include <fstream.h>`
`#include <conio.h>`
`main(){`
`ofstream R ( "Rez.rez");`
`ifstream D ( "Duom.dat");`
`int A;`
`D>> A;`
`cout<<" A= "<< A <<endl;`
`R<< A;`
`getch();`
	A= 15	15

```
}
```

6.10 pratimas. Duomenų failo "Duom.txt" perrašymas eilutėmis į naują failą "Rez.txt".Taip programiškai gaunama failo kopija. Metodas **`eof`** skirtas failo pabaigai nustatyti: grąžina reikšmę 0, kai failo pabaiga dar nepasiekta, ir 1, kai jau turime failo pabaigą.

```
#include <iostream.h>
#include <fstream.h>
#include <conio.h>
                                        // Failo kopija.
main(){
   ofstream R ( "Rez.txt");
   ifstream D ( "Duom.txt");
 char  A[80];
   while( ! D.eof() ){
       D.getline( A, sizeof( A ));
       cout<< A <<endl;
       R<< A << endl;          }
 getch();
}
```

Duomenų ir rezultatų failuose:

`Batuotas Katinas`
`pagavo gauruota peliuka.`
`Tupi sarka kamine.`

11 pratimas. Duomenų failo "Duom.txt" perrašymas žodžiais į naują failą. Ankstesnėje programoje pakeitus skaitymo iš failo metodą operatoriumi >>, gausime rezultatų faile atskirus duoto teksto žodžius.

	Rezultatų faileir ekrane:

`#include <iostream.h>`
`#include <fstream.h>`
`#include <conio.h>`
 	
`main(){`
`ofstream R ( "Rez.txt");`
`ifstream D ( "Duom.txt");`
`char A[80];`
`while( ! D.eof() ){`
	`Batuotas`
	`Katinas`
	`pagavo`
	`gauruota`
	`peliuka.`
	`Tupi`
	`sarka`
	`kamine.`

```
   D>> A;
   cout<< A <<endl;
   R<<  A << endl;      }
  getch();
 }
```

6.12 pratimas.

Duomenų failo "Duom.txt" perrašymas simboliais į naują failą ir ekraną. Ankstesnėje 10-o pratimo programoje pakeitus skaitymo iš failo metodą į `get`, gausime rezultatų faile ir ekrane duomenų failo kopiją.

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

 main(){
  ofstream R ( "Rez12.txt");
  ifstream D ( "Duom.txt");
  char  sim;
  while( ! D.eof() ){
    sim = D.get();
    cout<< sim;
    R<< sim;        }
  D.close(); R.close();
  getch();
 }
```

Dirbant su duomenų failais reikia patikrinti ar failas buvo surastas ir sėkmingai atidarytas. Kiekvienu skaitymo iš failo žingsniu būtina patikrinti, ar veiksmas buvo sėkmingai atliktas. Tam skirtas metodas **`fail(),`** kurio rezultatas yra 0, kai klaidų nebuvo, ir 1, kuomet įvyko trikis. Klaidų pranešimams ekrane skirtas išvedimo srautu objektas **`cerr`**. Metodas `fail()` naudojamas išvedime, norint įsitikinti, ar veiksmas buvo sėkmingas (buvo vietos naujams įrašui).

6.13 pratimas.

Duomenų failo "Duom.dat" skaičiai perrašomi į naują failą.

```
#include <iostream.h>
#include <fstream.h>
#include <conio.h>
main(){
  ofstream R ( "Rez13.txt");
  ifstream D ( "Duom.dat");
  int  a;
  if ( D.fail() ) cerr<<"Neatidarytas duomenu failas";
  else {
   while(( ! D.eof() ) && ( ! D.fail() )){
     D>> a;
     if( ! D.fail()){ R<< a<< endl; cout<< a <<endl; }
    } }
  D.close(); R.close();
  getch();
 }
```

"Duom.dat"	"Rez13.rez"
15 12 45 -56 3 15 89	15 12 45 -56 3 15 89

6.14 pratimas.

Šiose klasėse yra write ir read metodai, skirti duomenų mainams su failais baitų lygyje. Programa užrašo faile duotą struktūra, o po perskaito ir parodo ekrane. Metoduose užrašas (char *) nusako kompiliatoriui, kad bus naudojama rodyklė į skirtingus tipus.

```
#include <iostream.h>   //  14 pratimas
#include <fstream.h>
#include <conio.h>
struct Studentas{ char pav[20];
                  int  metai;
                  float ugis; };
main(){
  Studentas A;
  Studentas S = { "Petriukas ", 21, 187.5 };
  ofstream R ( "Rez14.txt");
  R.write( (char *) &S, sizeof( Studentas ));
  R.close();
    ifstream D ( "Rez14.txt");
    D.read( (char *) &A, sizeof( Studentas));
    cout<< A.pav << A.metai << "  " << A.ugis << endl;
    D.close();
```

```
getch();
 }
```

6.15 pratimas. Analogiškai galima sukurti baitinį failą struktūriniams duomenims iš tekstinio failo saugoti. Programa demonstruoja duomenų failo "Duom14.txt" perrašymą į failą "Rez15.txt" ir po to skaitymą iš jo ir duomenų rašymą ekrane. Duomenų failas:

```
Katinas Batuotas     45  12.5
Rapolas Didysis      15  159.98
Barbora Bulvyte      25  199.45
```

```
#include <iostream.h>  // 15 pratimas
#include <fstream.h>
#include <conio.h>

 struct Studentas{ char pav[20];
                   int  metai;
                   float ugis; };
main(){
  char sim;
  Studentas A;
ofstream R ( "Rez15.txt");
ifstream D ( "Duom14.txt");
  while( ! D.eof() ){         // Skaitomi duomenys.
    D.getline( A.pav, sizeof( A.pav));
    D>> A. metai >> A.ugis;
                              // Eilutės pabaiga.
    sim = ' ';
    while (( ! D.eof() ) && ( sim != '\n')){
      sim = D.get(); cout<< sim; }
                              // Duomenys ekrane.
    cout<< A.pav<<"  "
        << A.metai <<"  "<< A.ugis<< endl;
                              // Duomenys faile.
    R.write( (char *) &A, sizeof( Studentas ));
  }
    D.close();
    R.close();
       // Skaitomi failo duomenys ir parodomi ekrane.
   ifstream C ( "Rez15.txt");
   while( ! C.eof()) {
     C.read( (char *) &A, sizeof( Studentas));
  if ( !C.fail() )
  cout<< A.pav << A.metai << "  " << A.ugis << endl; }
  C.close();
 getch();
}
```