C++ programavimo kalba

I/O biblioteka

(2 paskaita)

I/O operatoriai

- Išvedimo <<
- Įvedimo >>

```
#include <iostream>
using namespace std;
void main()
{
   float A = 18.236;
   cout << "1. A="  << A << endl;
   cout << "2. A=" <<  A*2.0 << endl;
}
```

Srauto formatavimo manipuliatoriai

- **setw (int** n**)** n **–** lauko plotis simboliais,
- **setprecision (int** n**)** n **–** skaitmenų skaičius,
- **setfill(char c)** c – tai simblis, kuriuo užpildomi tarpai.

Jei **setw()** nurodyto lauko dydžio skaičiui nepakanka, manipuliatorius ignoruojamas. **setw()** galioja artimiausiai išvedamai reikšmei, o **setprecission()** ir **setfill()** – iki naujo nurodymo.

```
#include <iostream.h>
#include <iomanip.h>
void main() {
float A = 18.236;
cout << "1. A=" << setw(9) << A << endl;
cout << "2. A=" << setprecision(3) << A << endl;
cout << "3. A=“ << setw(10) << setprecision(5) << A <<endl;
A = 123.45678;
cout << "4. A=" << A << endl;
```

Srauto formatavimo funkcijos

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

```
#include <iostream>
#include <iomanip>
void main() {
 for(int i=2; i<6; i++) {
     cout.width(3);
     cout << "i=";
     cout.width(i);
     cout << i << endl;
  }
}
```

Srauto formatavimo funkcijos

- Metodas fill(int) leidžia pakeisti užpildymo simbolį norimu (nurodomas kaip argumentas).

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

ostream metodai (funkcijos)

- Metodas put() skirtas vienam simboliui išvesti ekrane.

  Sintaksė: **ostream::put(char *ch*);**

```
#include <iostream>
using namespace std;
void main() {
    char zodis[ ] = "Katinas";
    for(int i=0; zodis[ i ]; i++)
            cout.put(zodis[i]);
            cout << endl;
   }
```

istream metodai (funkcijos)

get() – tai istream klasės metodai, skirti neformatuotam nuskaitymui iš srauto.

Galimi sintaksės variantai:

int get() **-** skirtas vieno simbolio nuskaitymui iš srauto;

istream:: get(char* *pch*, **int** *nCount*, **char** *delim* **= '\n');**
Nuskaito eilutę iš *nCount* simbolių arba kol bus sutiktas *delim*

istream:: get(char* *pch*, **int** *nCount* **);**
Nuskaito eilutę iš *nCount* simbolių

Pavyzdžiai

```
#include <iostream>
#include <iomanip>
using namespace std;
        void main() {
        char sim;
        cout << "Ar dar dirbsite?( T/N ): ";
        do {    cin>>ws;                    // white space skipping manipulator
                sim = cin.get();
                sim = toupper(sim);
                if (sim !='T' && sim != 'N')
                { cout << "Ivestas neteisingas simbolis t.y.: " << sim<<endl ;
                cout << " Kartokite dar… Ar dar dirbsite?( T/N ): >"; }
            }
                while( (sim != 'T') && (sim != 'N') );
        }
```

Pavyzdžiai

```
#include <iostream>
#include <fstream>
using namespace std;
void main () {
   char c, str[256];
   ifstream is;
   cout << “Iveskite failo pavadinima: ";
   cin.get (str,256);
   is.open (str);            // failo atidarymas
   while ( is.good() )       // ciklas vykdomas, kol įmanoma nuskaityti
  {
    c = is.get();            // nuskaitome simbolį iš failo
    cout << c;
  }
  is.close();                // uždarome failą
}
```

istream metodai (funkcijos)

Įvedimo klasėje metodas getline() skirtas vienos eilutės įvedimui klaviatūra.

Sintaksė:

getline(char* *pch*, **int** *nCount*, **char** *delim* **= '\n');**

Paprastai įvedama eilutė iki pirmojo tarpo simbolio arba iki galo, jeigu tarpo simbolio nebuvo.

Panaudoję getline() metodą galima įvesti norimą simbolių skaičių (nurododant skaičių).

Jeigu tiek simbolių nebuvo, įvedami visi simboliai iki eilutės galo.

Įvedama eilutė visuomet papildoma nuliniu ASCII simboliu (\0).

Pavyzdys

```
#include <iostream>
using namespace std;
void 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";
  cin.getline(A, sizeof(A));                 // Iki galo arba tiek, kiek telpa
  cout << "Eilute: *" << A << "*\n";
  n = cin.gcount();                          // Eilutės ilgis su nuliniu simboliu
  cout << "n= " << n << endl;
}
```

istream

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, kiek nurodyta, jeigu eilutėje jų yra daugiau.

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

Pavyzdys

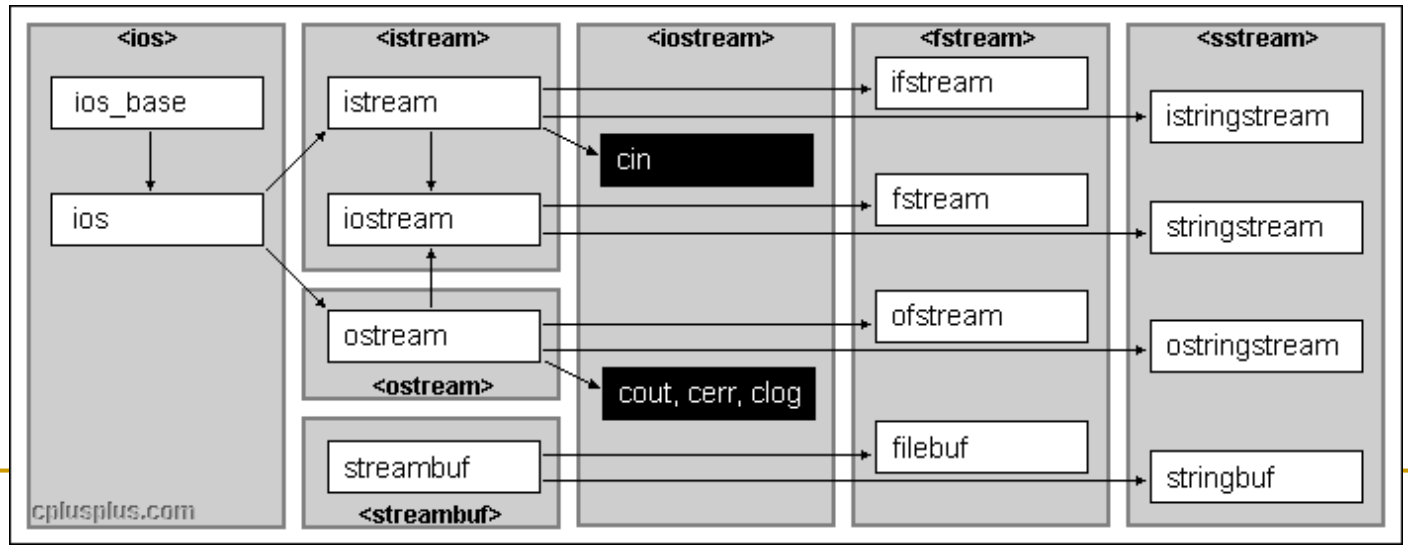
```
#include <iostream>
using namespace std;

int main () {
  char name[256], title[256];

  cout << “Ivesk savo varda: ";
  cin.getline (name,256);

  cout << “Ivesk megstamo filmo pavadinima: ";
  cin.getline (title,256);

  cout << name << "‘megsta filma: " << title;

  return 0; }
```

ifstream, ofstream

Įvedimo srautu iš failo klasė **ifstream** ir išvedimo srautu į failą klasė **ofstream** aprašytos faile **fstream.h**.

Tai išvestinės klasės iš **istream** ir **ostream**.

Galima sukurti objektus, kurie nukreipia įvedimo/išvedimo srautus iš/į failus. Galioja operatoriai **>>** ir **<<**. Konstruktoriaus parametru yra failo vardas. Failo uždarymui naudojamas metodas **close().**



Pavyzdys

```
#include <iostream>
#include <fstream>
using namespace std;
main()
{
    ofstream rezultatai ("Rez58.txt");
    ifstream duomenys ("Duom58.txt");
    int A;
        duomenys >> A;
        cout << " A= " << A << endl;
        rezultatai << A;
    duomenys.close();
    rezultatai.close();
}
```

Failų atidarymo rėžimai

Rėžimas	Aprašymas
ios::in	Failas atidaromas skaitymui
ios::out	Failas atidaromas rašymui
ios::ate	Failas atidaromas rašymui, išvedimas pradedamas nuo failo galo (at end)
ios::app	Failas atidaromas papildymui
ios::trunc	Naikinamas failo turinys
ios::binary	Failas atidaromas dvejatainiame režime

Srautų sintaksė

- **ifstream (const char *name, ios::openmode = ios::in);**

  (nuskaitymo iš failo srautas)

- **ofstream (const char *name, ios::openmode = ios::out | ios::trunc);**

  (*rašymo į failą srautas*)

- **fstream (const char *name, ios::openmode = ios::in | ios::out);**

  (*rašymo ir skaitymo iš/į failą srautas*)

Pavyzdžiai

```
#include <fstream>
using namespace std;
void main() {
ofstream fs;
fs.open(“File.dat”, ios::app);          //ofstream fs("File.dat", ios::app);

if ( ! fs )
    cout<< “Failo nepavyko atidaryti”<<endl;

if ( fs.is_open() )                     // is_open() grąžina true/false
    cout<< “Failo nepavyko atidaryti”<<endl;
}
```

Metodas eof()

Metodas **eof()** skirtas failo pabaigai nustatyti: grąžina reikšmę 0, kai failo pabaiga dar nepasiekta, ir 1, kai jau turime failo pabaigą.

```
#include <fstream>
main() {
        ofstream R ("Rez59.txt");
        ifstream D ("Duom59.txt");
        char A[80];
        while( ! D.eof() ) {
                D.getline(A, sizeof(A));    //   kas bus, jei D >> A;  ?
                cout << A << endl;
                R << A << endl;
                }
        D.close(); R.close();
        }
```

istream::get ifstream sraute

```
#include <iostream.h>
#include <fstream.h>
    main() {
    ofstream R("Rez511.txt");
    ifstream D("Duom511.txt");
    char sim;
      while(!D.eof()) {
           sim = D.get();      // get() naudojamas, kaip istream
           cout << sim;
            R << sim; }
    D.close();
    R.close();
    }
```

ios.fail()

Kiekvienu skaitymo iš failo žingsniu rekomenduojama patikrinti, ar veiksmas buvo sėkmingai atliktas.

Tam skirtas metodas **fail(),** kurio rezultatas yra 0, kai klaidų nebuvo, ir 1, kuomet įvyko klaida.

Klaidų pranešimams ekrane skirtas išvedimo srautu objektas **cerr**.

Metodas **fail()** naudojamas išvedime, norint įsitikinti, ar veiksmas buvo sėkmingas.

```
#include <iostream.h>
#include <fstream.h>
main(){
    ofstream R ("Rez512.txt");
    ifstream D ("Duom512.txt");
    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();
}
```

ostream::write istream::read

Šiose klasėse yra write() ir read() metodai, skirti duomenų mainams su failais baitų lygyje t.y. nuskaitymas ir įrašymas atliekamas nurodant baitų kiekį.

- **write(const char*** *pointer*, **int** *nCount* **);**
- **read (char*** *pointer*, **int** *nCount* **);**

```
#include <iostream.h>
#include <fstream.h>
    struct Studentas{      char pav[20]; int metai;   float ugis; };

main() {
Studentas A;
Studentas S = {"Petras ", 21, 187.5};

ofstream R("Rez513.txt“, ios::app);
R.write((char *) &S, sizeof(Studentas));
R.close();

ifstream D("Rez513.txt“, ios::in);
D.read((char *) &A, sizeof(Studentas));
cout << A.pav << " " << A.metai << " " << A.ugis << endl;
D.close();
}
```

```
struct Studentas   { char pav[20];  int metai; float ugis; };
main() {
char sim;  Studentas A;
ofstream R("Rez514.txt“, ios::app);
ifstream D("Duom514.txt");
      while(!D.eof()) {
                D.getline(A.pav, sizeof(A.pav));
                D >> A.metai >> A.ugis;
                sim = ' ';                      // Eilutės pabaiga
                while(!D.eof() )
                { 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("Rez514.txt");
      while(!C.eof()) {
      C.read((char *) &A, sizeof(Studentas));
         if (!C.fail())
                cout << A.pav << A.metai << " " << A.ugis;
                cout << endl; }
      C.close();
      }
```