13 PASKAITA

Turinys: Duomenų mainai su failais.
Failas. Srautas.
Formatavimas plačiau.
Srautai – funkcijų argumentai.

Bendros žinios

Čia – tik trumpa įžanga į minėtas temas. Daug kas bus nagrinėjama supaprastintai.

Failas – vieną vardą turinti, logiškai susijusi duomenų seka kompiuterio išorinėje arba vidinėje atmintyje.
Kad įvestis/išvestis (I/O) iš klaviatūros/į ekraną arba iš failo/į failą būtų programoje atliekama panašiai, įvesta "srauto" programinė abstrakcija: simbolių arba bitų seka, nukreipiama į kokį nors kompiuterio įrenginį. Kad būtų galima srautą susieti su reikiamu kompiuterio įrenginiu ar failu jo atmintyje, juos reikia "atidaryti" metodu *open*.
Faktiškai srautas yra tam tikros klasės objektas. Tą objektą galima susieti su vienu failu, dirbti su juo, jį atjungti nuo failo, susieti su kitu failu ir t.t. Srautai *cout* ir *cin* taip pat yra iš anksto apibrėžti (antraštiniame faile *iostream*) klasių objektai.

Klasė įvesties srautui: *ifstream*,
klasė išvesties srautui: *ofstream*;
abi jos apibrėžtos antraštiniame faile *fstream*.

Srauto gyvavimo ciklas:

1. Skelbimas ir susiejimas su failu

```
ifstream is;
ofstream os;
is.open( "duomenys.dat" );
os.open( "rezultatai.dat") ;
```

2. Darbas su failais, taikant operacijas << ir >> ir operuojant vidiniais programos failų vardais *is* ir *os*. *"duomenys.dat"* – išorinis programos atžvilgiu, OS vardas.

3. Baigus darbą, failų atsiejimas

```
is.close( );
os.close();
```

Failų vardus galima nurodyti kaip čia, trumpąja forma – jei failai yra darbiniame kataloge (tame kataloge, kur yra darbinės aplinkos failas *.dsw); arba pilnąja forma – jei failai yra kitame kataloge.
Atjungimas – nebūtinas (korektiškai baigianti savo darbą programa juos automatiškai atjungia), bet rekomenduotinas (jei programa "lūžta" – neuždarytas failas gali būti sugadintas).

Pavyzdys: iš failo nuskaityti du skaičius ir jų sumą išvesti į kitą failą.

```
#include <fstream>
using namespace std;
//
int main( ){
        ifstream is;  // skelbimas
        ofstream os;
        is.open( "duomenys.dat" );  // susiejimas
        os.open( "rezultatai.dat");
                int s1, s2;
                is>>s1>>s2;  // darbas; dialogo nereikia
                os<<"Ivesti skaiciai: "<<s1<<"  "<<s2
                  <<" Ju suma: "<<(s1+s2)<<endl;
                        is.close( );  //  uzdarymas
                        os.close( );
        return 0;
}
```

Tokia programa – nesaugi: duomenų failo nurodytu vardu kataloge gali nebūti. Todėl metodu *fail* (grąžina *bool* reikšmę *true*, jei nepavyksta) reikėtų tikrinti, ar pavyksta atverti failą:

```
. . .
if( is.fail( ) ){
        cout<<"Nera tokio pradinio failo. Stop \n";
        exit( 1 );
}
. . .
```

Failo papildymas. Failų vardų įvestis dialoge

Dažnai patogu, kai failų vardus įveda pats programos vartotojas. Failų vardus galima saugoti *char* formato vienmačiuose masyvuose, kurių ilgis vienetu ilgesnis nei yra prasminės informacijos. Vienas vardo simbolis – vienas masyvo elementas.
Kad duomenis būtų galima jungti prie esančio failo pabaigos, metodui *open* reikia nurodyti ir antrąjį argumentą *ios::app*.
Pavyzdys: ta pat programa kaip anksčiau, tik rezultatas prirašomas pradinio failo gale.

```
#include <fstream>
#include <iostream>
#include <cstdlib>
using namespace std;
//
int main( ){
        char fv[ 16 ]; // failo vardas – iki 15 simboliu ilgio
        ifstream is;
        ofstream os;
        cout<<"Iveskite pradinio failo varda iki 15 simboliu ilgio:\n";
```

```
        cin>>fv;
        is.open( fv );
        if( is.fail( ) ){
                cout<<"Nera tokio failo: "<<fv<<" Stop"<<endl;
                exit( 1 );
        }
        int s1, s2;
        is>>s1>>s2;
        is.close( );
        os.open( fv, ios::app ); // prideti rezultatus prie to pat failo
        os<<"Skaiciu suma: "<<(s1+s2)<<endl;
        os.close( );
        return 0;
}
```

Formatavimas

1 būdas: metodais su argumentais-vėliavėlėmis. Iškvietus juos, paveikiamas tolesnis srautas. Metodai:
precision (p), *p* – skaitmenų po kablelio skaičius;
setf(ios::flag), flag:

fixed – išvedimas su fiksuota kablelio vieta
scientific – išvedimas su plaukiančia kablelio vieta
showpoint – visada išvesti dešimtainį tašką
showpos – visada prieš teigiamus skaičius išvesti +
right – lygiuoti pagal dešinį išvesties lauko kraštą
left – tas pat, kairį

Pagal numatymą yra pakelta tik vėliavėlė *right*.

2 būdas: funkcijomis-manipuliatoriais. Taip pat veikia tolesnį srautą. Manipuliatoriai:
setprecision(p)
setiosflags(ios::flag)
setw(w)
endl

Srautai-funkcijų argumentai. Failo pabaigos požymis

Srauto objektas į funkciją atiduodamas tik adresu. Šiaip jau geras stilius reikalauja visą I/O atlikti pagrindinėje programoje, arba specializuotose I/O funkcijose – tada ir reikėtų perduoti joms srautų objektus.
Failo pabaigą aptikti galima taip:

1. Metodu *bool eof( )*, grąžinančiu reikšmę *true*, jei pabaiga aptikta.
2. Tiesiog tikrinant operacijos >> reikšmę: operacijai priskiriama reikšmė *true*, jei ji įvykdoma, ir *false* – neįvykdžius.

Pavyzdys: skaityti visą bet kaip surašytą pradinį *double* skaičių failą *"duomenys.dat"* ir sukurti tvarkingai suformatuotą failą *"rezultatai.dat"*, kurio vienoje eilutėje būtų po vieną skaičių.

```
#include <fstream>
#include <iostream>
#include <iomanip>
#include <cstdlib>
using namespace std;
//
void for_file( ifstream& is, ofstream& os, int precision, int width );
//
int main( ){
        ifstream is;
        ofstream os;
//
        is.open( "duomenys.dat" );
        if( is.fail( ) ){
                cout<<"Nera pradinio failo\n";
                exit( 1 );
        }
        os.open( "rezultatai.dat", ios::app );
        if( os.fail( ) ){
                cout<<"Klaida atveriant rezultatu faila\n";
                exit( 1 );
        }
//
        for_file( is, os, 5, 10 );
//
        is.close( );
        os.close( );
//
        return 0;
}
//
void for_file( ifstream& is, ofstream& os, int p, int w ){
        os.setf( ios::fixed );
        os.setf( ios::showpoint );
        os.setf( ios::showpos );
        os.precision( p );
                double s;
                while( !is.eof( ) ){
                        is>>s;
                        os<<setw( w )<<s<<endl;
                }
//  arba:
//                 while( is >> s ){
//                        os<<setw( w )<<s<<endl;
//                }

}
```