Работа с входно/изходни файлове

Базово програмиране
Лекция 10
Милена Карова
кат. КНТ ТУ-Варна

Потоков вход и изход

- C++ разглежда въвеждането и извеждането на информация като четене от текстов поток (istream - input stream) и запис в изходен поток (ostream – output stream)
- Файл- набор от еднотипни данни, които се съхраняват върху външен носител
- Ако данните „идват“ от външен поток, то той може да е от обект клавиатура (cin>>) или файл свързан с друго входно устройство
- Ако данните „отиват“ към изходен поток, то той може да е от обект екран на монитор (cout<<) или файл свързан с друго изходно устройство

Променлива свързана с файла (файлов указател)

- Входно-изходните потоци са дефинирани в iostream.h
- Можете да дефинирате други входно-изходни потоци (не свързани с cin и cout) , които се пренасочват към файлове. Веднъж дефинирани, те се използват по същия начин както се използват cin>> и cout<<
- Ако дефинирате потока in_stream, то променливата е the_number се въвежда

```
int the_number;
in_stream >> the_number;
```

- Ако се дефинира потока out_stream, то извеждането е

```
out_stream << "the_number is " << the_number << endl;
```

Четене и запис

- Когато извеждаме данните във файл казваме че записваме. Записът започва от началото напред и компютърът не се връща назад за да редактира данни от предходни записи
- В C++ потокът е специален тип променлива, позната като обект.
- Ако искаме да запишем потокът във файл или да прочетем от файл, потокът трябва да се декларира и свърже с файла
- Потокът може да се свърже и прекъсне с един файл и после да се свърже с друг файл, т. е. потоковите променливи могат да променят своите стойности
- Потоковите променливи не могат да се присвояват като обикновени променливи или да участват в действия

istream и ostream

- Типът на input file stream е ifstream, а на output file stream е ofstream
```
ifstream in_stream;
ofstream out_stream;
```
- ifstream и ofstream са дефинирани в fstream.h
- Потоковите променливи трябва да се свържат с файловете. Използват се функции за работа с потоци
```
in_stream.open("infile.dat");
```
- От тук нататък името на потоковата променлива е файлова променлива и е свързана със съответния файл

```
int one_number, another_number;
in_stream >> one_number >> another_number;
```

```
ofstream out_stream;
out_stream.open("outfile.dat");
```

```
out_stream << "one_number = " << one_number
           << " another_number = " << another_number;
```

Име на файла и файлова променлива

- При запис на файл, ако файлът съществува вече, изстрива се неговото съдържание и се отваря наново за запис
- Разлика между двете „имена“ на файла – реалното име на файла и файловата променлива
- Външното име на файла се използва само веднъж при привързване с файловата променлива. Оттам нататък програмата работи с файловата променлива
- Всеки файл трябва да се затвори, щом приключи работата на програмата с него

```
in_stream.close( );
out_stream.close( );
```

Пример за файлов вход/изход

```
//Reads three numbers from the file infile.dat, sums the numbers,
//and writes the sum to the file outfile.dat.
#include <fstream>

int main( )
{
    using namespace std;
    ifstream in_stream;
    ofstream out_stream;

    in_stream.open("infile.dat");
    out_stream.open("outfile.dat");

    int first, second, third;
    in_stream >> first >> second >> third;
    out_stream << "The sum of the first 3\n"
               << "numbers in infile.dat\n"
               << "is " << (first + second + third)
               << endl;

    in_stream.close( );
    out_stream.close( );

    return 0;
}
```

infile.dat
(Not changed by program.)

```
1
2
3
4
```

outfile.dat
(After program is run.)

```
The sum of the first 3
numbers in infile.dat
is 6
```

Функция fail()

- Възможно е дефиниране на файлова променлива едновременно за вход и изход

fstream fp; но тогава отварянето трябва да се съпъства с аргументи показващи статуса на файловата променлива

fp.open("primer.dat", ios::out);

Възможно отварянето на файла да бъде неуспешно по ред причини и това трябва да се предвиди `in_stream.fail( )`

fail() функция може да се използва за да провери дали файла съществува Няма аргументи и връща булева стойност

Входно/изходна техника

- Поставя се непосредствено след отваряне на файла

```
in_stream.open("stuff.dat");
if (in_stream.fail( ))
{
    cout << "Input file opening failed.\n";
    exit(1);          <-- Ends the program
}
```

Може да имате няколко отворени файла едновременно.

Screen Output (If the file infile.dat does not exist)

```
Input file opening failed.
```

```
//Reads three numbers from the file infile.dat, sums the numbers,
//and writes the sum to the file outfile.dat.
#include <fstream>
#include <iostream>
#include <cstdlib>

int main( )
{
    using namespace std;
    ifstream in_stream;
    ofstream out_stream;

    in_stream.open("infile.dat");
    if (in_stream.fail( ))
    {
        cout << "Input file opening failed.\n";
        exit(1);
    }

    out_stream.open("outfile.dat");
    if (out_stream.fail( ))
    {
        cout << "Output file opening failed.\n";
        exit(1);
    }

    int first, second, third;
    in_stream >> first >> second >> third;
    out_stream << "The sum of the first 3\n"
               << "numbers in infile.dat\n"
               << "is " << (first + second + third)
               << endl;

    in_stream.close( );
    out_stream.close( );

    return 0;
}
```

Добавяне във файл

- Възможност да не се изтрива старото съдържание и да се добавят нови данни във файла

```
ofstream outStream;
outStream.open("important.txt", ios::app);
```

- Ако файлът не съществува, то той се създава и се записват данните в него
- ios::app е дефинирано в iostream.h, както и останалите стойности на ios::
- Името на файла може да бъде стрингова променлива

```
char file_name[16];
```

```
ifstream in_stream;
in_stream.open(file_name);
if (in_stream.fail( ))
{
    cout << "Input file opening failed.\n";
    exit(1);
}
```

Пример за добавяне на записи

```
//Appends data to the end of the file data.txt.
#include <fstream>
#include <iostream>

int main( )
{
    using namespace std;

    cout << "Opening data.txt for appending.\n";
    ofstream fout;
    fout.open("data.txt", ios::app);
    if (fout.fail( ))
    {
        cout << "Input file opening failed.\n";
        exit(1);
    }

    fout << "5 6 pick up sticks.\n"
         << "7 8 ain't C++ great!\n";

    fout.close( );
    cout << "End of appending to file.\n";

   return 0;
}
```

data.txt
(Before program is run.)

```
1 2 buckle my shoe.
3 4 shut the door.
```

data.txt
(After program is run.)

```
1 2 buckle my shoe.
3 4 shut the door.
5 6 pick up sticks.
7 8 ain't C++ great!
```

Форматиране Функции put и get

- Форматиране на данните във файловете е възможно така както и с обекта cout<<
- Функция get прочита

```
out_stream.setf(ios::fixed);
out_stream.setf(ios::showpoint);
out_stream.precision(2);
```

един символ и го записва в променлива от тип char

- Чете празните интервали и символа '\n'

```
cout << "Enter a line of input and I will echo it:\n";
char symbol;
do
{
    cin.get(symbol);
    cout << symbol;
} while (symbol != '\n');
cout << "That's all for this demonstration.";
```

```
Enter a line of input and I will echo it:
Do Be Do 1 2    34
Do Be Do 1 2    34
That's all for this demonstration.
```

```
cout.put(next_symbol);
cout.put('a');
out_stream.put('Z');
```

put изисква като аргумент променлива или константа от тип char

Функция eof()

- Функцията показва дали всичко от файла е прочетено и дали нещо остава от него.
- Няма аргументи

```
fin.eof( )
```

- Връща истина, ако е стигнат края на файла и лъжа в противен случай. Най-често се използва not eof()

```
if (! fin.eof( ))
    cout << "Not done yet.";
else
    cout << "End of the file.";
```

```
in_stream.get(next);
while (! in_stream.eof( ))
{
    cout << next;
    in_stream.get(next);
}
```

If you prefer, you can use `cout.put(next)` *here.*

- Във втория пример цикълът извежда цялото съдържание на файла на екрана Функцията става true едвам когато е прочетен и последния символ и програмата направи опит да прочете следващ символ

Пример за редактиране на съдържанието на файл

```
//Program to create a file called cplusad.dat that is identical to the file
//cad.dat, except that all occurrences of 'C' are replaced by "C++".
//Assumes that the uppercase letter 'C' does not occur in cad.dat except
//as the name of the C programming language.

#include <fstream>
#include <iostream>
#include <cstdlib>
using namespace std;

void add_plus_plus(ifstream& in_stream, ofstream& out_stream);
//Precondition: in_stream has been connected to an input file with open.
//out_stream has been connected to an output file with open.
//Postcondition: The contents of the file connected to in_stream have been
//copied into the file connected to out_stream, but with each 'C' replaced
//by "C++". (The files are not closed by this function.)

int main( )
{
    ifstream fin;
    ofstream fout;

    cout << "Begin editing files.\n";

    fin.open("cad.dat");
    if (fin.fail( ))
    {
        cout << "Input file opening failed.\n";
        exit(1);
    }

    fout.open("cplusad.dat");
    if (fout.fail( ))
    {
        cout << "Output file opening failed.\n";
        exit(1);
    }
    add_plus_plus(fin, fout);
    fin.close( );
    fout.close( );

    cout << "End of editing files.\n";
    return 0;
}

void add_plus_plus(ifstream& in_stream, ofstream& out_stream)
{
    char next;

    in_stream.get(next);
    while (! in_stream.eof( ))
    {
        if (next == 'C')
            out_stream << "C++";
        else
            out_stream << next;

        in_stream.get(next);
    }
}
```

cad.dat
(Not changed by program.)

```
C is one of the world's most modern programming languages.
There is no language as versatile as C, and C is fun to use.
```

cplusad.dat
(After program is run.)

```
C++ is one of the world's most modern programming languages.
There is no language as versatile as C++, and C++ is fun to
use.
```

Screen Output

```
Begin editing files.
End of editing files.
```

Стойности на статуса ios

- Функцията open(), освен име на файла, съдържа стойности на ios статуса (режима) за работа с файла, представени в следната таблица:

Режим	Значение
app	Отваря файла за добавяне на данни към края му
ate	Премества указателя в края на файла при отваряне
binary	Отваря файла в двоичен режим на достъп
in	Отваря файла за четене
out	Отваря файла за запис
trunc	Изтрива съдържанието на съществуващ файл при отваряне

Пример за четене и запис в двоичен файл

- Съставете програма, която записва в двоичен файл низ и две числа от тип double. Програмата трябва да прочита данните от файла и да ги извежда на екрана

Примерен вариант на решение:

```
#include <fstream.h>
#include <iostream.h>
#include <stdlib.h>
#include <string.h>

void main()
 {      char niz[80];
   double chislo1, chislo2;
   fstream fp1;
   fp1.open("chisla.dat",ios::binary| ios::out);
   if(fp1.fail())
    {cout << endl << "Файлът не може да се отвори"<< endl;
     exit(1);}
   cout << endl << "Въведете низ:";
   cin.getline(niz,80);
   int razmer = strlen(niz);
   fp1.write(niz,razmer);
   cout << endl << "Въведете число1:";
   cin >> chislo1;
   fp1.write((char*)&chislo1,sizeof(double));
   cout << endl << "Въведете число2:";
   cin >> chislo2;
   fp1.write((char*)&chislo2,sizeof(double));
   fp1.close();
   strcpy(niz,"");
   chislo1=chislo2=0.0;
   fp1.open("chisla.dat",ios::binary|ios::in);
   fp1.read(niz,razmer);
   fp1.read((char*)&chislo1, sizeof(double));
   fp1.read((char*)&chislo2, sizeof(double));
   fp1.close();
   cout << endl << niz;
   cout << endl << chislo1 << "  " << chislo2 << endl;
}
```

Преобразуването на тип към **(char*)** е необходимо, когато се извежда **в буфер**, който не е дефиниран като **символен масив**. Указателят от даден тип не може автоматично да се преобразува в указател от друг тип

Използване на функции seekg() и tellg()

- Съставете **C++** програма за създаване на файл от 10 символа. Да се реализира възможност за замяна на конкретен номер символ с друг и да се изведе съдържанието на файла на екрана.

```
#include <fstream.h>
#include <iostream.h>
#include <stdlib.h>
#include <string.h>

void main()
{char simvoli[] = "Varna e mojat roden grad";
 char symbol;
 long pos;
// Записва стринга във файла
 fstream fp;
 fp.open("sentence.dat",ios::out|ios::binary);
 if(fp.fail())
  {cout << endl << "Файлът не се отваря";
   exit(1);}
 fp.write(simvoli,strlen(simvoli));
 fp.close();
// Смяна на символа
 cout << endl << "Въведете, коя позиция ще променяте ";
 cin >> pos;
 fp.open("sentence.dat",ios::in|ios::binary|ios::out);
 if(fp.fail())
  {cout << endl << "Файлът не се отваря";
   exit(1);}
 fp.seekg(pos,ios::beg); // Позиционира файла върху позиция pos
 fp.get(symbol);
 cout << endl << "Символът е "<< symbol; // Визуализира
                                        // символа на позиция pos
 cout << endl<< "Въведете новия символ";
 cin >> symbol;
 fp.seekg(-1L,ios::cur);          // Връщане една позиция назад
 fp.put(symbol);                  // Заместване с новия символ
 fp.close();
 fp.open("sentence.dat",ios::in|ios::binary);
 if(fp.fail())
  {cout << endl << "Файлът не се отваря";
   exit(1);}
 fp.read(simvoli,strlen(simvoli));//Четене на променения стринг
 fp.close();
 cout << endl << "Променен стринг: " << simvoli;
}
```

Резултатът се получава в следния вид:

```
Въведете коя позиция ще сменяте 9
Символът е o
Въведете новия символ $
Променен стринг : Varna e m$jat roden grad
```

Символът ‘o’ се явява **десети** символ в низа, но позициите на указателя на файла започват да се броят от **0** и в този случай тя е **9**. Променливата **pos** е от **тип long**, тъй като функцията **seekg()** изисква като **параметър отместването** в байтове да е стойност от т**ип long**. Тази функция поставя указателя на файла в началото на указаната позиция.

Използване на функции seekg() и tellg()

- При прочитане на **символа 'o'**, указателят на файла се премества в началото на **символа 'j'**. Затова чрез:

fp.seekg(-1L,ios::cur)

указателят на файла се връща една позиция назад (отместването е **-1L** – назад спрямо текущата позиция на указателя на файла – **cur**), т. е. отново застава върху **'o'**.

- Прекият (произволен) достъп до записите на файла се осъществява с помощта на функцията **seekg()**, като позициите спрямо които се извършва отместването са: **ios::beg** – спрямо **началото** на файла; **ios::cur** – спрямо **текущата** позиция на указателя на файла; **ios::end** – спрямо **края** на файла. Ако записите на файла са по-големи от **1** байт, то тогава може да се използва оператора **sizeof**.
- f**p.seekg(pos*sizeof(тип на данните), ios::beg);**

Указателят на файла ще се **отмести** с толкова **байта**, колкото е стойността на **произведението на pos и типа на данните**.

- За осъществяване на **произволен достъп** до записите на двоичния файл, може да се използва и функцията: **tellg()**, която връща резултат тип **long** - **текущата позиция на указателя на файла.**

read и write

```
fp.seekg(0l,ios::end); // Премества указателя на файла в края
pos=fp.tellg(); // Определя дължината на файла в брой байтове
```

- Четене на стойностите на цели масиви с функциите read и write

```
fp.read((char*)&p,sizeof(person));

fp.write((char*)per,n*sizeof(person)); /* Запис на масива във
                                          файл */
```