Лекція 16

Робота з файлами

Файл – поіменована сукупність даних, розташованих на зовнішньому носії.

На початку роботи для отримання доступу до даних файл необхідно відкрити. При відкритті файлу встановлюється вказівник на поточну позицію (як правило на початок даних у файлі). Після виконання будь-якої операції над даними вказівник автоматично переміщується на одну позицію даних вперед. Після завершення роботи з даними файлу його необхідно закрити.

Інформація про файл зберігається в управляючій структурі, яка має тип **FILE**.

Розрізняють два типи файлів: текстові та двійкові (бінарні).

Текстовий файл зберігає інформацію у вигляді послідовності символів. Виведення виконується аналогічно виводу на екран. Текстові файли можуть бути створені та відредаговані у будь-якому текстовому редакторі.

Бінарні файли призначені для зберігання послідовності байтів. Структура такого файлу визначається програмно.

Файли розташовуються на зовнішніх носіях і мають наступну структуру:

На початку записана інформація про файл **BOF** (*Begin of FILE*), його ім’я, тип, довжина і т.д. В кінці розміщується ознака кінця файлу **EOF** (*End of File*).

Для роботи з файлами використовують наступні макроси:

NULL – визначає пустий вказівник;

EOF — ознака кінця файлу

FOPEN_MAX – максимальна кількість одночасно відкритих файлів.

Організація роботи з файлами засобами С

Для запису даних в файл потрібно виконати

1. Описати покажчик на файл FILE * filename;

2. Відкрити файл (функція fopen)

3. Записати даних в файл (функція **fprintf**)

Функція **fprintf** аналогічна функції printf, єдиною відмінністю є перший параметр – покажчик на файл. За допомогою цієї функції висновок здійснюється не на екран, а в файл.

4. Закрити файл (функція **fclose**).

int fclose (FILE * filename);

Повертає 0 при успішному закритті файлу і NULL в іншому випадку.

Оголошення файлу:

FILE * ідентифікатор;

Приклад

FILE * f; // f – ім’я файлу (ідентифікатор у програмі)

Для відкриття файлу використовується функція **fopen**:

fopen (ім'я фізичного файлу, режим доступу)

Ім’я фізичного файлу – шлях до файлу на диску та його ім’я на диску.

Режим доступу – рядок, який вказує режим відкриття файлу і тип файлу

Типи файлу:

- бінарний (b);
- текстовий (t).

Режими відкриття

Значення	Опис
r	Файл відкривається тільки для читання. Вказівник встановлюється на початок.
w	Файл відкривається тільки для запису. Якщо відповідний фізичний файл існує, він буде перезаписаний. Вказівник встановлюється на початок.

a	Файл відкривається для запису в кінець (для дозаписи) або створюється, якщо не існує. Вказівник встановлюється в кінець
r+	Файл відкривається для читання і запису. Вказівник встановлюється на початок.
w+	Файл відкривається для запису і читання. Якщо відповідний фізичний файл існує, він буде перезаписаний. Вказівник встановлюється на початок.
a+	Файл відкривається для запису в кінець (для дозаписи) або створюється, якщо не існує. Вказівник встановлюється в кінець

Команда відкриття (створення порожнього) файлу запишеться так:

```
FILE * fo;
fo = fopen ("test.txt", "wt");
```

Можна задати і повний шлях до файлу, наприклад:

```
fo = fopen ("c: \\ tmp \\ test.txt", "wt");
```

Не забуваємо, що одиночний символ \ всередині рядка Сі задається двома похилими Слеш \\. Це часта помилка.

Після відкриття файлу в файлову змінну ***fo*** занесеться деяке число. Якщо таким числом буде нуль, вважається, що файл відкрити не вдалося. У Сі нерідкі записи виду

```
if ((fo = fopen ("c: \\ tmp \\ test.txt", "wt")) == 0) {
  // помилка!   }
```

де одночасно відкривається файл і перевіряється, чи успішно це зроблено. Наступна програма записує в файл 100 рядків з числами. Результуючий файл буде текстовим, його можна відкрити і переглянути, наприклад, в Блокноті.

```
FILE * fo;
fo = fopen ("test.txt", "wt");
int i;
for (i = 0; i <100; i ++) {
  fprintf (fo, "% d \ n", i);   }
fclose (fo);
```

Комбінація символів "\ n" в кінці рядка формату означає переклад на новий рядок в виводяться текстових даних.

Крім цих функцій для роботи з файлами є ще дві:

- int remove (const char * filename);

Ця функція видаляє з диска файл, покажчик на який зберігається в файловій змінній **filename**. Функція повертає нульове значення, якщо файл не вдалося видалити.

- int rename (const char * oldfilename, const char * newfilename);

Функція перейменовує файл; перший параметр – старе ім'я файлу, другий – нове. Повертає 0 при невдалому завершенні програми.

Для читання даних з файлу потрібно виконати:

1.Описати покажчик на файл **FILE** * filename;

2. Відкрити файл (функція **fopen**)

3.Считать дані з файлу (функція **fscanf**)

Функція **fscanf** аналогічна функції **scanf**, єдиною відмінністю є перший параметр – покажчик на файл. За допомогою цієї функції виведення здійснюється не на екран, а в файл. У разі необхідності при читанні треба контролювати можливість читання чергового компонента за допомогою функції **feof**.

4. Закрити файл (функція **fclose**)

Програма зчитування з файлу всіх рядків з числами запишеться так:

```
FILE * fi;
fi = fopen ("test.txt", "rt"); // rt  відкриття текстового файлу на читання
int n;
while (! feof (fi)) {
  fscanf (fi, "% d", & n);   }
fclose (fi);
```

Приклад.

У файлі abc.txt зберігаються матриця A(N, M) і B(M, K). Нехай структура файлу наступна: в першому рядку зберігаються числа n і, рядка матриці A, за ним рядок матриці B значень m і k. Знайти матрацю C = A.B і записати її в файл **rez.txt**.

#include <stdio.h> #include <alloc.h> int main ()

```
{int i, j, n, m, l, k;
 float * b, * c * a, s, temp;
FILE * f;
f = fopen ("abc.txt", "r");
fscanf (f, "% d% d", & n, & m);
a = (float *) calloc (n * m, sizeof (float));
for (i = 0; i <n; i ++)
                for (j = 0; j <m; j ++)
{fscanf (f, "% g", & temp);
* (A + i * m + j) = temp; }
 fscanf (f, "% d% d", & m, & l);
b = (float *) calloc (m * l, sizeof (float));
c = (float *) calloc (n * l, sizeof (float));
for (i = 0; i <m; i ++)
          for (j = 0; j <l; j ++)
{fscanf (f, "% g", & temp);
 * (B + i * l + j) = temp; }
 fclose (f);
f = fopen ("rez.txt", "w");
for (i = 0; i <n; i ++)
for (j = 0; j <l; * (c + i * l + j) = s, j ++)
for(s=0,k=0;k<m;k++)
                s+=*(a+i*m+k)**(b+k*l+j);
                fprintf(f,"Matrica C\n");
for(i=0;i<n;fprintf(f,"\n"),i++)
 for(j=0;j<l;j++)
fprintf(f,"%g\t",*(c+i*l+j));
fclose(f);
free(a);
 free(b);
free(c);}
```

Робота з текстовими файлами за допомогою файлових потоків

Для роботи з файлами в мові С++ передбачена бібліотека **"fstream.h"**. Тому на початку програми необхідно цю бібліотеку підключити:

#include <fstream.h>

fstream надає функціонал для зчитування даних з файлу і для запису в файл.

Найбільш часто використовуються наступні операції:

- Оператори перенаправлення вводу \ виводу - << і >>
- Методи запису і читання рядків **getline**() і **get**() з **put**()
- Передача потокового запису і читання методами **write**() і **read**()
- Методи відкриття \ створення і закриття файлів **open**() і **close**()
- Методи перевірки чи відкритий файл **is_open**() і досягнутий кінець файла **eof**()
- Налаштування форматованого виведення для >> за допомогою **width**() і **precision**()
- Операції позиціонування **tellg**(), **tellp**() і **seekg**(), **seekp**()

Це не всі можливості, які надає бібліотека **fstream**. Розглядати всі зараз ми не будемо, познайомимося з вище переліченими. Почнемо з класу читання.

Створення потоків

У програмі перед першим зверненням до потоку введення або виведення необхідно "створити" потік. Оператори для створення потоків схожі на описи змінних, і вони зазвичай розміщуються на початку програми або функції поряд з описами змінних. Наприклад, оператори

ifstream in_stream;

ofstream out_stream;

створюють потік з ім'ям "in_stream", який є об'єктом класу (як типу даних) "Ifstream" (input-file-stream, файловий потік введення), і потік з ім'ям "out_stream", є об'єктом класу "ofstream" (output-file-stream, файловий потік виводу).

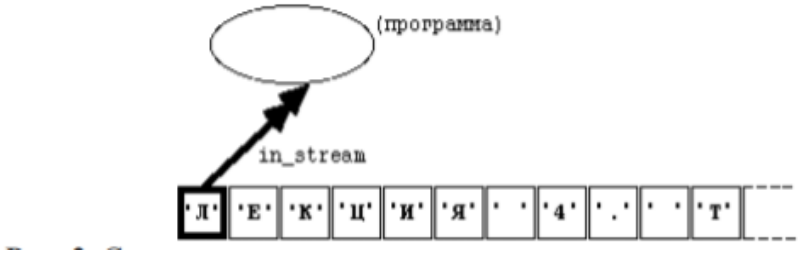
Аналогію між потоками і звичайними змінними (типу "int", "char" і т.д.) не слід розуміти занадто буквально. Наприклад, до потоків не можна застосовувати оператор присвоювання (наприклад, не можна записати "in_stream1 = in_stream2").

Підключення та відключення потоків від файлів

Після створення потоку його можна підключити до файлу (відкрити файл) за допомогою функції-члена "**open**(...)". Для підключення потоку **ifstream** з ім'ям "***in_stream***" до файлу з ім'ям "Lecture.txt" треба застосувати наступний виклик:

in_stream.open ("Lecture.txt");

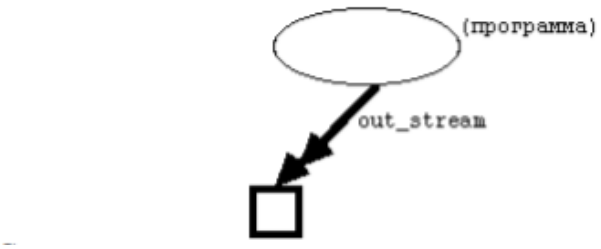
Цей оператор підключить потік "***in_stream***" до початку файлу "Lecture.txt" (Графічно стан програми після виконання оператора показано на рис).



Щоб до файлу "Lecture.txt" підключити потік виведення **ofstream** з ім'ям "***Out_stream***", треба виконати аналогічний оператор:

out_stream.open ("Lecture.txt");

Цей оператор підключить потік "***out_stream***" до файлу "Lecture.txt", але при цьому колишній уміст файлу буде видалено. Файл буде підготовлений до прийому нових даних.



Для відключення потоку "***in_stream***" від файлу, до якого він підключений (для закриття файлу), треба викликати функцію-член "**close** ()":

in_stream.close ();

Функція-член відключення від файлу у потоку виведення:

out_stream.close ();

виконує аналогічні дії, але, додатково, в кінець файлу додається службовий символ "***end-of-file*** (маркер кінця файлу)", навіть якщо в потік виводу не записувалися ніякі даних, то після відключення потоку "***out_stream***" в файлі "Lecture.txt" буде один службовий символ. В такому випадку файл "Lecture.txt" залишиться на диску, але він буде порожнім.

Запис у файл

```
// file.cpp: визначає точку входу для консольного додатку.
#include "stdafx.h"
#include <fstream>
using namespace std;
int main (int argc, char * argv [])
{  ofstream fout ("cppstudio.txt"); // створюємо об'єкт класу ofstream для запису і пов'язуємо його з файлом cppstudio.txt
    fout << "Робота з файлами в C ++"; // запис рядка в файл
    fout.close (); // закриваємо файл
    system ("pause");
    return 0;}
```

Читання з файлу

```
// file_read.cpp: визначає точку входу для консольного додатку.
#include "stdafx.h"
#include <fstream>
#include <iostream>
using namespace std;
int main (int argc, char * argv [])
{   setlocale (LC_ALL, "rus"); // коректне відображення кирилиці
    char buff [50]; // буфер проміжного зберігання зчитує з файлу тексту
    ifstream fin ("cppstudio.txt"); // відкрили файл для читання
    fin >> buff; // вважали перше слово з файлу
    cout << buff << endl; // надрукували це слово
    fin.getline (buff, 50); // вважали рядок з файлу
    fin.close (); // закриваємо файл
    cout << buff << endl; // надрукували цей рядок
    system ("pause");
    return 0;}
```

Перевірка помилок виконання файлових операцій

Файлові операції, наприклад, відкриття і закриття файлів, відомі як один з найбільш ймовірних джерел помилок. В надійних комерційних

програмах завжди виконується перевірка, успішно чи ні завершилася файлова операція. У разі помилки викликається спеціальна функція-обробник помилки.

Найпростіший спосіб перевірки помилок файлових операцій полягає у виклику функції-члена "**fail** ()". виклик

in_stream.fail ();

повертає істинне значення (***True***), якщо остання операція потоку "***in_stream***" привела до помилки (може бути, була спроба відкриття неіснуючого файлу).

Після помилки потік "***in_stream***" може бути пошкоджений, тому краще не продовжувати роботу з ним.

У наведеному нижче фрагменті програми в разі помилки при відкритті файлу на екран видається повідомлення і програма завершує роботу за допомогою бібліотечної функції "**exit** ()" (вона описана в файлі "**stdlib.h**"):

```
#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
int main ()
{  ifstream in_stream;
  in_stream.open ("Lecture_4.txt");
  if (in_stream.fail ())
  {  cout << "Вибачте, відкрити файл не вдалося! \ n";
  exit (1);  }
```

або

```
#include "stdafx.h"
#include <fstream>
#include <iostream>
using namespace std;
int main (int argc, char * argv [])
{  setlocale (LC_ALL, "rus"); // коректне відображення кирилиці
   char buff [50]; // буфер проміжного зберігання зчитує з файлу тексту
   ifstream fin ( "cppstudio.doc"); // (ВВЕЛИ НЕ чинні ім'я ФАЙЛА)
   if (! fin.is_open ()) // якщо файл не відкритий
      cout << "Файл не може бути відкритий! \ n"; // повідомити про це
   else
   {   fin >> buff; // вважали перше слово з файлу
   cout << buff << endl; // надрукували це слово
   fin.getline (buff, 50); // вважали рядок з файлу
   fin.close (); // закриваємо файл
```

```
cout << buff << endl; // надрукували цей рядок  }
system ( "pause");
return 0;}
```

Режими відкриття файлів

Режими відкриття файлів встановлюють характер використання файлів. Для установки режиму в класі **ios_base** передбачені константи, які визначають режим відкриття файлів (див. Таблиця 1).

Константа	Опис
ios_base::in	відкрити файл для читання
ios_base::out	відкрити файл для запису
ios_base::ate	при відкритті перемістити покажчик в кінець файлу
ios_base::app	відкрити файл для запису в кінець файлу
ios_base::trunc	видалити вміст файлу, якщо він існує
ios_base::binary	відкриття файлу в двійковому режимі

Режими відкриття файлів можна встановлювати безпосередньо при створенні об'єкта або при виконанні функції open ().

ofstream *fout* ("cppstudio.txt", ios_base :: app); /* відкриваємо файл для додавання інформації до кінця файлу */

***fout*.open** ("cppstudio.txt", ios_base :: app); /* відкриваємо файл для додавання інформації до кінця файлу */

Режими відкриття файлів можна комбінувати за допомогою поразрядной логічної операції або |, наприклад: ios_base :: out | ios_base :: **trunc** – відкриття файлу для запису, попередньо очистивши його.

Об'єкти класу ofstream, при зв'язці з файлами за замовчуванням містять режими відкриття файлів ios_base :: out | ios_base :: trunc. Тобто файл буде створений, якщо не існує. Якщо ж файл існує, то його вміст буде видалено, а сам файл буде готовий до запису. Об'єкти класу ifstream зв'язуючись з файлом, мають за замовчуванням режим відкриття файлу ios_base :: in – файл відкритий тільки для читання. Режим відкриття файлу ще називають -

прапор, для зручності читання в подальшому будемо використовувати саме цей термін. У таблиці 1 перераховані далеко не всі прапори, але для початку цих повинно вистачити.

Зверніть увагу на те, що прапори ***ate*** і ***app*** по опису дуже схожі, вони обидва переміщують покажчик в кінець файлу, але прапор app дозволяє проводити запис, тільки в кінець файлу, а прапор ***ate*** просто переставляє прапор в кінець файлу і не обмежує місця запису.

Розробимо програму, яка, використовуючи операцію **sizeof**(), буде обчислювати характеристики основних типів даних в С ++ і записувати їх в файл. Характеристики:

- число байт, що відводиться під тип даних
- максимальне значення, яке може зберігати певний тип даних.

Символьне введення / виведення

4.1 Функція введення "get (...)"

Після того, як файл для введення даних відкритий, з нього можна зчитувати окремі символи. Для цього служить функція "**get**(...)". У неї є параметр типу "***Char &***". Якщо програма знаходиться в стані відкритого файлу, то після виклику:

in_stream.get (ch);

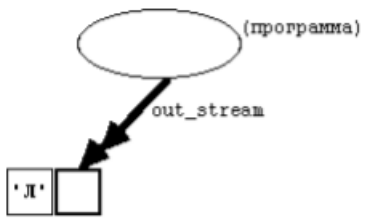
відбудеться наступне: (а) змінної "ch" буде присвоєно значення 'Л' (зчитається перший символ), і (б) потік "***in_stream***" буде підготовлений для читання наступного символу.

Функція виведення "put (...)"

За допомогою потоку виведення класу **ofstream** у відкритий файл можна записувати окремі символи. Для цього у класу **ofstream** є функція-член "**put** (...)".

Записуваний символ передається їй як параметр типу "**char**". Якщо програма перебуває в стані, представленому на рис. 3, то оператор

out_stream.put ('Л');



змінить стан на той, що показано на рис.

Зчитування цілого рядка до переведення каретки проводиться так само як і в iostream методом getline (). Причому рекомендується використовувати його перевизначену версію у вигляді функції, якщо зчитується рядок типу string:

```
// зчитування рядки з тексту
string s;
getline (file, s);
cout << s << endl;
```