Лекція 13.
Робота з файлами.

FACULTY OF APPLIED MATHEMATICS AND INFORMATICS
LVIV UNIVERSITY

План на сьогодні

1. Потоки вводу та виводу
2. Відкриття та закриття файлів
3. Перевірка файлу на помилки
4. Читання з файлу
5. Запис у файл
6. Робота з fstream
7. Робота з string streams

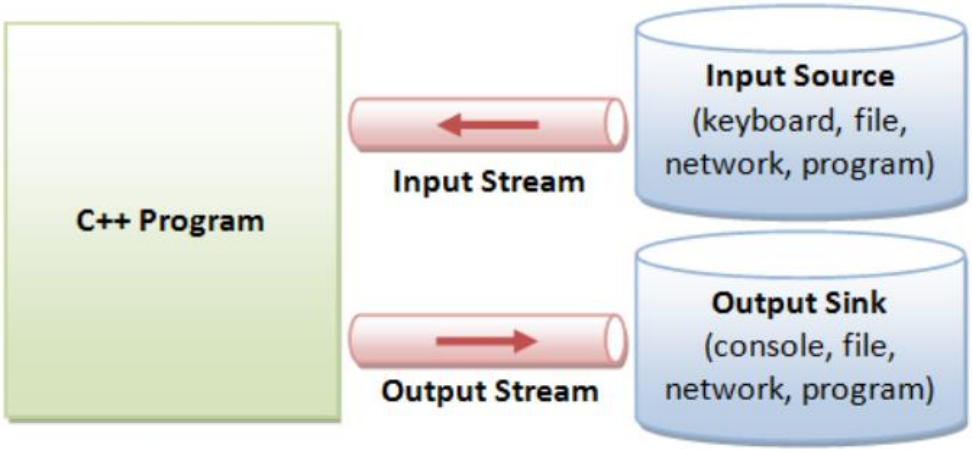
FACULTY OF APPLIED MATHEMATICS AND INFORMATICS
LVIV UNIVERSITY

Потоки вводу та виводу

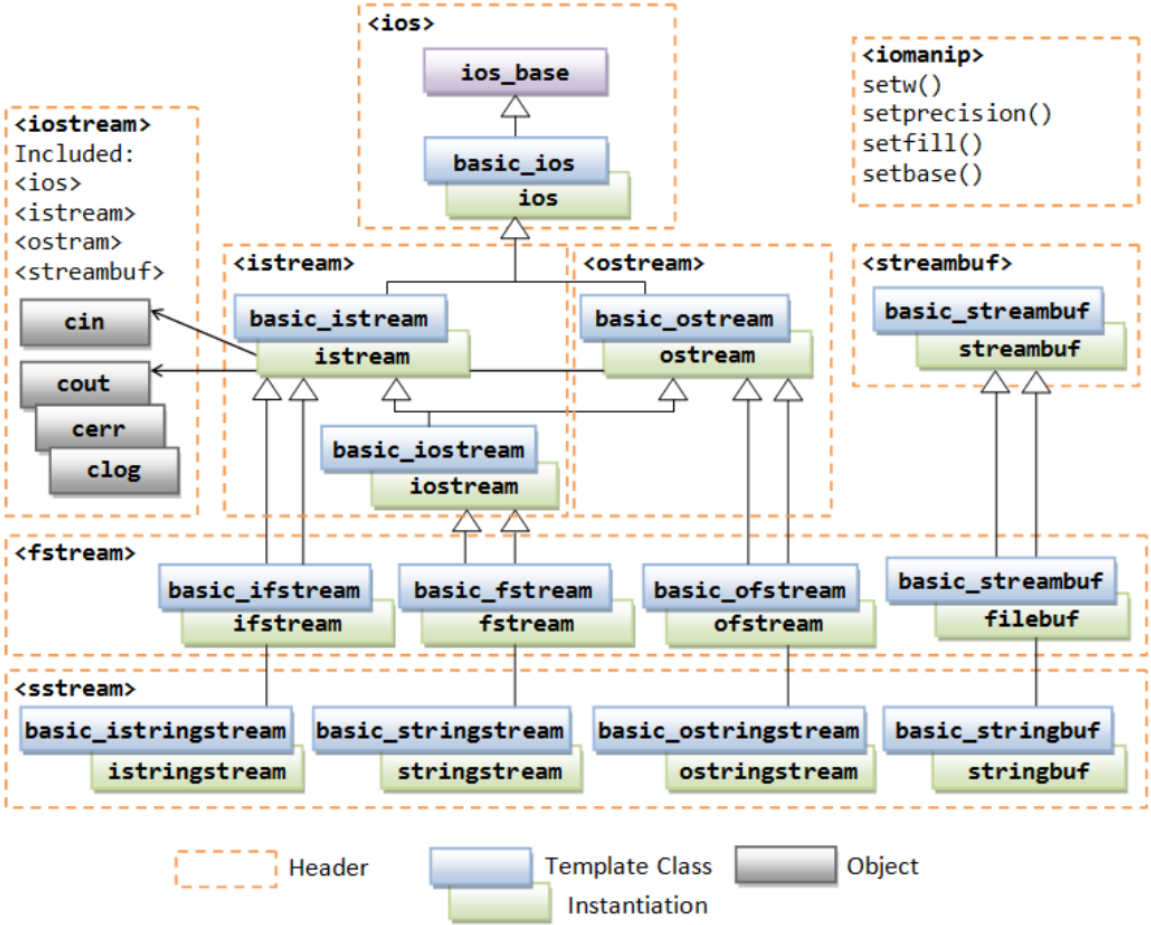
FACULTY OF APPLIED
MATHEMATICS AND
INFORMATICS
LVIV UNIVERSITY

Input & Output Streams

C/C++ IO базується на потоках, які є послідовністю байт, що надходять і виходять з програм. Під час операцій введення байти даних надходять із джерела введення (наприклад клавіатури, файлу, мережі чи іншої програми). Під час операцій виведення байти даних переходять від програми до приймача виводу (наприклад консолі, файлу, мережі чи іншої програми). Потоки діють як посередники між програмами та фактичними пристроями вводу-виводу.



C++ IO Headers


<ios>
ios_base
basic_ios
ios
<iomanip>
setw()
setprecision()
setfill()
setbase()
<iostream>
Included:
<ios>
<istream>
<ostram>
<streambuf>
cin
cout
cerr
clog
<istream>
basic_istream
istream
basic_iostream
iostream
<ostream>
basic_ostream
ostream
<streambuf>
basic_streambuf
streambuf
<fstream>
basic_ifstream
ifstream
basic_fstream
fstream
basic_ofstream
ofstream
basic_streambuf
filebuf
<sstream>
basic_istringstream
istringstream
basic_stringstream
stringstream
basic_ostringstream
ostringstream
basic_stringbuf
stringbuf
Header
Template Class
Instantiation
Object

Форматований вивід - width, fill, alignment

```
// Test setw() - need <iomanip>
cout << "|" << setw(5) << 123 << "|" << 123 << endl;  // |  123|123
        // setw() is non-sticky. "|" and 123 displayed with default width
cout << "|" << setw(5) << -123 << "|" << endl;            // | -123|123
        // minus sign is included in field width
cout << "|" << setw(5) << 1234567 << "|" << endl;      // |1234567|
        // no truncation of data

// Test setfill() and alignment (left|right|internal)
char ch = cout.fill( );
cout << "Initial fill = " << ch << '\n';
cout << setfill('_');   // Set the fill character (sticky)
cout << setw(6) << 123 << setw(4) << 12 << endl;      // ___123__12
cout << left;          // left align (sticky)
cout << setw(6) << 123 << setw(4) << 12 << endl;      // 123___12__
cout << setfill(ch);
```

Форматований вивід - alignment

```
cout << showpos;  // show positive sign
cout               << '|' << setw(6) << 123 << '|' << endl;  // |  +123| (default alignment)
cout << left       << '|' << setw(6) << 123 << '|' << endl;  // |+123  |
cout << right      << '|' << setw(6) << 123 << '|' << endl;  // |  +123|
cout << internal << '|' << setw(6) << 123 << '|' << endl;  // |+  123|
```

Форматований вивід - floating-point format

```
// default floating-point format
cout << "|" << 123.456789 << "|" << endl;          // |123.457| (fixed-point format)
        // default precision is 6, i.e., 6 digits before and after the decimal point
cout << "|" << 1234567.89 << "|" << endl;          // |1.23457e+006| (scientific-notation for e>=6)
        // default precision is 6, i.e., 6 digits before and after the decimal point
cout << "|" << 0.00001234567 << "|" << endl;    // |1.23457e-005| (scientific-notation for e<=-5)
// showpoint - show trailing zeros in default mode
cout << showpoint << 123. << "," << 123.4 << endl; // 123.000,123.400
cout << noshowpoint << 123. << endl;              // 123

// fixed-point formatting
cout << fixed;
cout << "|" << 1234567.89 << "|" << endl;        // |1234567.890000|
        // default precision is 6, i.e., 6 digits after the decimal point

// scientific formatting
cout << scientific;
cout << "|" << 1234567.89 << "|" << endl;        // |1.234568e+006|
        // default precision is 6, i.e., 6 digits after the decimal point
```

Форматований вивід - setprecision

```
// Test precision
streamsize ss = cout.precision( );
cout << "Initial precision = " << ss << '\n';    // Initial precision = 6

cout << fixed << setprecision(2);   // sticky
cout << "|" << 123.456789 << "|" << endl;  // |123.46|
cout << "|" << 123. << "|" << endl;              // |123.00|

cout << setprecision(0);
cout << "|" << 123.456789 << "|" << endl;  // |123|

cout << setprecision(ss);
```

Форматований вивід - dec|oct|hex, setbase

```
cout << hex << 1234 << endl;              // 4d2
cout << oct << 1234 << endl;              // 2322
cout << setbase(10) << 1234 << endl;   // 1234 (setbase requires <iomanip> header)
cout << showbase << 123 << "," << hex << 123 << "," << oct << 123 << endl;  // 123,0x7b,0173
cout << noshowbase << dec;

// uppercase - display in uppercase (e.g., hex digits)
cout << uppercase << hex << 123 << endl;  // 7B
```

Робота з файлами

FACULTY OF APPLIED MATHEMATICS AND INFORMATICS
LVIV UNIVERSITY

Робота з файлами

Обробка файлів у C++ — це механізм для створення та виконання операцій читання/запису у файл.

Ми можемо отримати доступ до різних методів обробки файлів у C++, імпортуючи клас <fstream>.

<fstream> включає два класи для обробки файлів:

- ifstream - для читання з файлу.
- ofstream - для створення/відкриття та запису у файл.

Відкриття та закриття файлів

FACULTY OF APPLIED MATHEMATICS AND INFORMATICS
LVIV UNIVERSITY

Відкриття файлу

Щоб працювати з файлами, спочатку потрібно їх відкрити. У C++ ми можемо відкрити файл за допомогою класів ofstream і ifstream.

Наприклад, ось як можна відкрити файл за допомогою ofstream:

```
std::ofstream my_file("example.txt");
```

Тут:

- my_file - ім'я об'єкта класу ofstream.
- example.txt - ім'я та розширення файлу, який ми хочемо відкрити.

Закриття файлу

Після завершення роботи з файлом потрібно його закрити, використовуючи функцію close().

```
my_file.close();
```

Відкриття та закриття файлу

Цей код відкриває та закриває файл example.txt.

Якщо файл з таким ім'ям відсутній, то ofstream my_file("example.txt"); замість цього створить новий файл із назвою example.txt.

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
    // відкриття текстового файлу для запису
    ofstream my_file("example.txt");
    // закриття файлу
    my_file.close();
    return 0;
}
```

Перевірка файлу на помилки

Перевірка файлу на помилки

При обробці файлів важливо переконатися, що файл було відкрито без помилок, перш ніж виконувати будь-які операції з ним. Існує три загальноприйняті способи перевірки файлів на помилки:

1. Перевірка об'єкта файлу

```
ofstream my_file("example.txt");
// перевірка, чи файл було відкрито належним чином
if (!my_file) {
    // вивід повідомлення про помилку
    cout << "Error opening the file." << endl;
    // завершення функції main()
    return 1;
}
```

Цей метод перевіряє, чи файл перебуває в стані помилки, шляхом оцінки самого об'єкта файлу.

- Якщо файл відкрито успішно, умова приймає значення true.
- Якщо виникає помилка, умова приймає значення false, і ви можете відповідно обробити помилку.

Перевірка файлу на помилки

2. Використання функції is_open()

```
ofstream my_file("example.txt");
if (!my_file.is_open()) {
    cout << "Error opening the file." << endl;
    return 1;
}
```

Функція is_open() повертає:

- true - якщо файл було успішно відкрито.
- false - якщо файл не вдалося відкрити або він перебуває в стані помилки.

Перевірка файлу на помилки

3. Використання функції fail()

```
ofstream my_file("example.txt");
if (my_file.fail()) {
    cout << "Error opening the file." << endl;
    return 1;
}
```

Функція fail() повертає:

- true - якщо файл не вдалося відкрити або він перебуває в стані помилки.
- false - якщо файл було успішно відкрито.

Читання з файлу

FACULTY OF APPLIED MATHEMATICS AND INFORMATICS
LVIV UNIVERSITY

Читання з файлу

Читання з текстових файлів здійснюється шляхом відкриття файлу за допомогою класу ifstream

```
ifstream my_file("example.txt");
```

Потрібно організувати цикл, який буде проходити по кожному рядку файлу, доки всі рядки не будуть прочитані, тобто поки ми не досягнемо кінця файлу.

Для цього ми використовуємо функцію eof(), яка повертає:

- true - якщо вказівник на файл знаходиться в кінці файлу.
- false - якщо вказівник на файл не знаходиться в кінці файлу.

Читання з файлу

Тут цикл while буде виконуватись до кінця файлу. На кожній ітерації циклу:

- getline(my_file, line); читає поточний рядок файлу та зберігає його у змінну line.
- Потім виводиться значення змінної line.

```
// змінна для зберігання вмісту файлу
string line;
// цикл до кінця файлу
while (!my_file.eof()) {
    // зберегти поточний рядок файлу у змінну "line"
    getline(my_file, line);
    // вивести змінну line
    cout << line << endl;
}
```

Запис у файл

Запис у файл

Для запису у файл ми використовуємо клас ofstream

```
ofstream my_file("example.txt");
```

```
int main() {
    // відкриття текстового файлу для запису
    ofstream my_file("example.txt");
    // перевірка файлу на наявність помилок
    if(!my_file) {
        cout << "Error: Unable to open the file." << endl;
        return 1;
    }
    // запис декількох рядків у файл
    my_file << "Line 1" << endl;
    my_file << "Line 2" << endl;
    my_file << "Line 3" << endl;
    // закриття файлу
    my_file.close();
    return 0;
}
```

Для запису у файл, використовується оператор << з об’єктом ofstream my_file.

При обробці файлів ми просто замінюємо cout на об'єкт файлу, щоб записувати у файл, а не в консоль.

Запис у наявний файл перезапише його поточний вміст.

Додавання до текстового файлу

Щоб додати новий вміст до вже існуючого файлу, необхідно відкрити файл у режимі додавання.

У C++ це можна зробити, використовуючи прапорець ios::app під час відкриття файлу

```
ofstream my_file("example.txt", ios::app);
```

```
int main() {
    ofstream my_file("example.txt", ios::app);
    if(!my_file) {
        cout << "Failed to open the file for appending." << endl;
        return 1;
    }
    // додавання декількох рядків у файл
    my_file << "Line 4" << endl;
    my_file << "Line 5" << endl;
    // закриття файлу
    my_file.close();
    return 0;
}
```

Робота з fstream

FACULTY OF APPLIED
MATHEMATICS AND
INFORMATICS
LVIV UNIVERSITY

Робота з файлами за допомогою fstream

Замість того, щоб використовувати ifstream для читання з файлу та ofstream для запису у файл, ми можемо просто використовувати клас fstream для всіх операцій з файлами. Конструктор для fstream дозволяє вказати ім'я файлу та режим для операцій з файлом.

```
fstream  f;
f.open("file.dat", ios::out | ios::in );
```

Режим	Опис
ios::in	Відкриває файл для читання (за замовчуванням для ifstream).
ios::out	Відкриває файл для запису (за замовчуванням для ofstream).
ios::app	Відкриває файл і додає новий вміст у його кінець.
ios::trunc	Відкриває файл та видаляє старий вміст
ios::binary	Відкриває файл для операцій вводу/виводу у двійковому форматі
ios::ate	Відкриває файл та розміщує вказівник файлу "в кінці" для введення/виведення.

Робота з string streams

FACULTY OF APPLIED
MATHEMATICS AND
INFORMATICS
LVIV UNIVERSITY

ostringstream

```
// construct output string stream (buffer) - need <sstream> header
ostringstream sout;

// Write into string buffer
sout << "apple" << endl;
sout << "orange" << endl;
sout << "banana" << endl;

// Get contents
cout << sout.str() << endl;
```

istringstream

```
// construct input string stream (buffer) - need <sstream> header
istringstream sin("123 12.34 hello");

// Read from buffer
int i;
double d;
string s;
sin >> i >> d >> s;
cout << i << "," << d << "," << s << endl;
```

Дякую!

FACULTY OF APPLIED
MATHEMATICS AND
INFORMATICS
LVIV UNIVERSITY