Лекція 3

ВВІД-ВИВІД У МОВІ С++

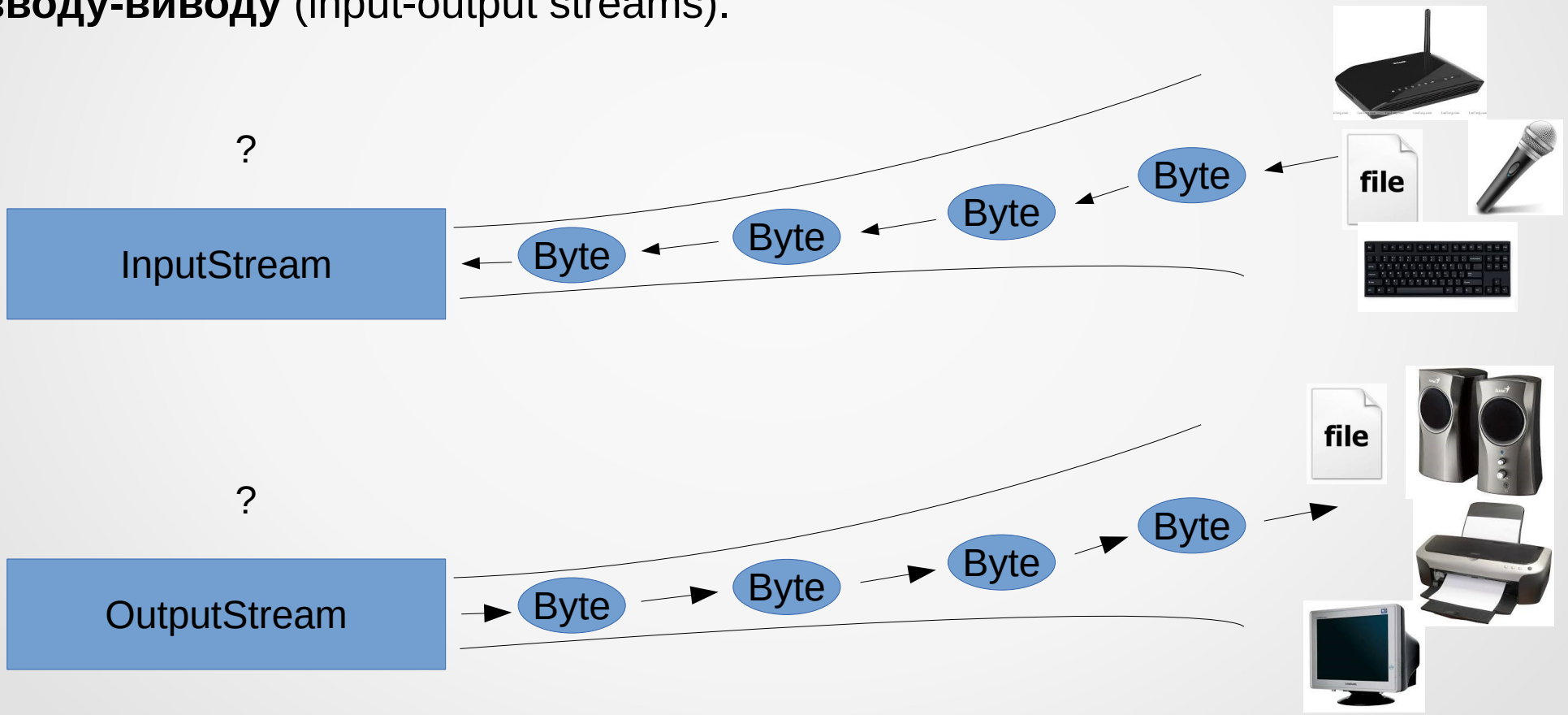
Лекція 3. Ввід-вивід у мові С++

План

1. Стандартні потокові класи вводу-виводу.
2. Маніпулятори. Форматований ввід-вивід.
3. Файловий ввід-вивід.

1. Стандартні потокові класи вводу-виводу

У мові програмування C++, як і у C, немає вбудованих засобів вводу-виводу. У C++ для цього використовують спеціальні класи, які називаються **потоками вводу-виводу** (input-output streams).



1. Стандартні потокові класи вводу-виводу

Потоки вводу-виводу реалізовані в бібліотеці **iostream**, опис класів якої можна підключити до програми за допомогою директиви

#include <iostream>

У iostream визначено три стандартні потоки:

- **cin** – стандартний потік вводу (аналог **stdin** у мові C);
- **cout** – стандартний потік виводу (аналог **stdout**);
- **cerr** – стандартний потік виводу повідомлень про помилки (**stderr** в C);

Ці потоки визначені у **std** – стандартному просторі імен C++. Для їх використання потрібно або підключити цей простір директивою

using namespace std;

або в явному вигляді вказувати використовуваний простір імен, наприклад, **std::cout**.

1. Стандартні потокові класи вводу-виводу

У класі, що реалізує стандартний потік виводу (**ostream**), перевантажена побітова операція зсуву вліво мови С (“<<”). Тут вона використовується для виведення інформації (даних) у потік. Наприклад, команда

```
cout << "x=" << 10.5;
```

виведе на екран (за замовчуванням) текст “x=10.5”. Тобто спочатку в потік буде відправлено ланцюжок байтів, що утворюють рядок “x=”, а потім – “10.5”.

Аналогічним чином у класі, що реалізує стандартний потік вводу (**istream**), перевантажена операція “>>”, яка тепер має новий зміст: отримання із вхідного потоку даних. Наприклад, в результаті виконання наступного коду:

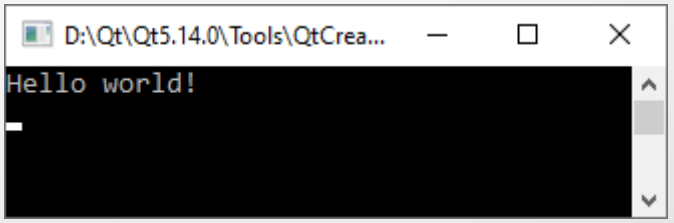
```
int a, b, c;

cin >> a >> b >> c;
```

буде зроблено спробу послідовного зчитування із вхідного потоку (клавіатури) трьох цілих чисел і присвоєння їх значень змінним a, b і c відповідно.

1. Стандартні потокові класи вводу-виводу

Приклад використання стандартного потоку виводу:

```
#include <iostream>

using namespace std;    // Підключення простору імен std

int main()
{
    // Виведення у стандартний потік виводу (екран)
    cout << "Hello world!" << endl;
    return 0;
}
```



1. Стандартні потокові класи вводу-виводу

Приклад використання потоків із явним вказуванням простору імен:

```
#include <iostream>

int main()
{
	int value;

	std::cout << "Input integer value" << std::endl;
	std::cin >> value;
	std::cout << "Entered value: " << value << std::endl;

	return 0;
}
```

Примітка. Виведення у потік std::endl призводить до переходу на новий рядок.

1. Стандартні потокові класи вводу-виводу

Таким чином, виведення інформації в потік здійснюється у загальному вигляді таким чином:

cout << вираз_1 << вираз_2 << ... << вираз_n;

де вираз_і – це деякий вираз, змінна чи константа.

Аналогічно, введення інформації з потоку здійснюється виразом:

cin >> змінна_1 >> змінна_2 >> ... >> змінна_n;

Примітка 1. При множинному введенні з потоку дані, що вводяться, повинні бути розділені пробілом, символом табуляції ('\t') або символом кінця рядка ('\n').

Примітка 2. Для введення рядка, що містить символи пропуску, необхідно використовувати спеціальний маніпулятор потоку getline().

2. Маніпулятори. Форматований ввід-вивід

Для завдання параметрів форматування потоку у C++ використовуються спеціальні функції, звані **маніпуляторами** (manipulators), які можуть включатися у вирази вводу-виводу.

Таблиця 1. Найпоширеніші потокові маніпулятори

Маніпулятор	Опис
endl	Вивід у вихідний потік символу кінця рядка та “скидання” буфера
dec	Виведення чисел у десятковому форматі
oct	Виведення чисел у вісімковому форматі
hex	Виведення чисел у шістнадцятковому форматі
width(), setw()	Завдання ширини поля виводу
fill(), setfill()	Заповнення порожніх знакомісць заданим значенням символу
precision()	Завдання кількості значущих цифр у числі (або після коми) залежно від використання fixed
fixed	Індикатор того, що встановлена точність відноситься до кількості знаків після коми
showpos	Виведення знаку “+” для позитивних чисел

2. Маніпулятори. Форматований ввід-вивід

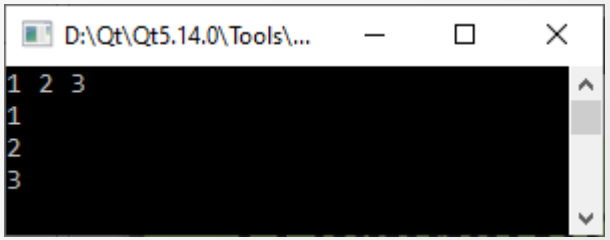
Таблиця 1 (продовження)

Маніпулятор	Опис
scientific	**Виведення числа у науковій нотації**
get()	**Зчитування символу із вхідного потоку**
getline()	Зчитування рядка із вхідного потоку

Приклад використання маніпулятора endl

```
#include <iostream>

using namespace std;

int main()
{
    int a, b, c;

    cin >> a >> b >> c;
    cout << a << endl << b << endl << c << endl;
    return 0;
}
```

Результат роботи програми

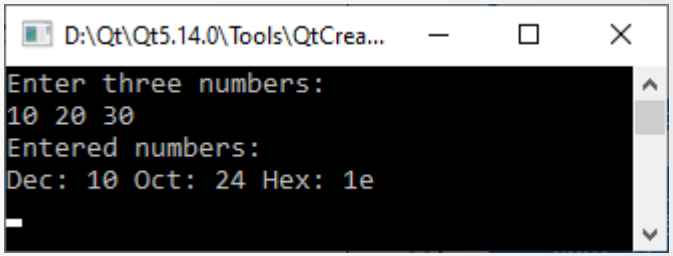


2. Маніпулятори. Форматований ввід-вивід

Приклади використання маніпуляторів dec, oct та hex

```
#include <iostream>

using namespace std;

int main()
{
    int a, b, c;

    cout << "Enter three numbers: " << endl;
    cin >> a >> b >> c;
    cout << "Entered numbers:" << endl;
    cout << "Dec: " << dec << a << " Oct: " << oct << b << " Hex: " << hex << c << endl;
    return 0;
}
```

Результат роботи програми

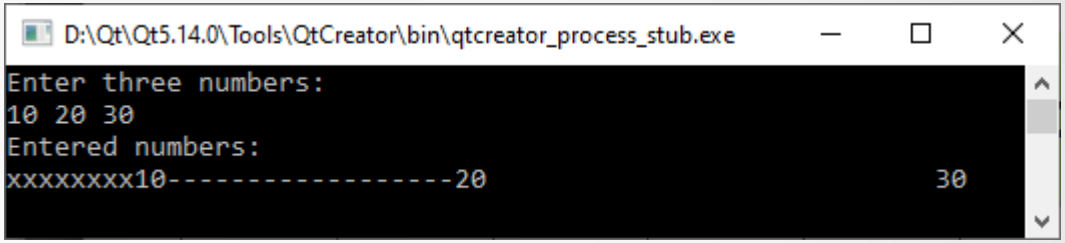


2. Маніпулятори. Форматований ввід-вивід

Приклади використання маніпуляторів width(), setw(), fill() та setfill()

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    int a, b, c;

    cout << "Enter three numbers: " << endl;
    cin >> a >> b >> c;
    cout << "Entered numbers:" << endl;
    cout.width(10);
    cout.fill('x');
    cout << a;
    cout << setw(20) << setfill('-') << b << setw(30) << setfill(' ') << c << endl;
    return 0;
}
```

Результат роботи програми



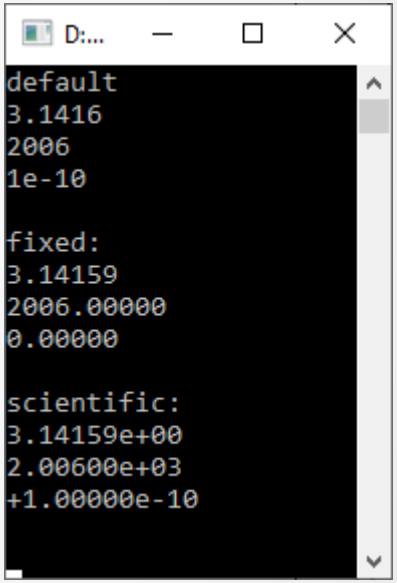
Примітка. Для використання маніпуляторів setw() та setfill() необхідно підключити заголовний файл iomanip

2. Маніпулятори. Форматований ввід-вивід

Приклади використання маніпуляторів precision(), fixed, showpos та scientific

```
#include <iostream>

using namespace std;

int main ()
{
  double a = 3.1415926534, b = 2006.0, c = 1.0e-10;

  cout.precision(5);
  cout << "default" << endl;
  cout << a << endl << b << endl << c << endl << endl;

  cout << "fixed:" << endl << fixed;
  cout << a << endl << b << endl << c << endl << endl;

  cout << "scientific:" << endl << scientific;
  cout << a << endl << b << endl << showpos << c << endl << endl;
  return 0;
}
```

Результат роботи програми

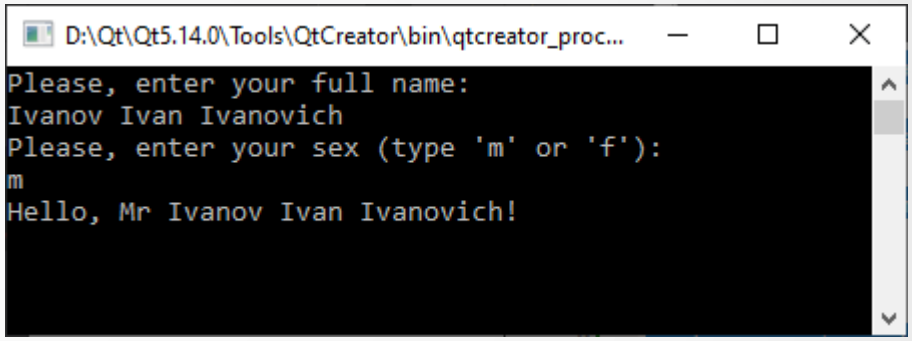


2. Маніпулятори. Форматований ввід-вивід

Приклади використання маніпуляторів getline() та get()

```
#include <iostream>

using namespace std;

const int max_len = 1000;

int main ()
{
  char name[max_len + 1], ch;

  cout << "Please, enter your full name: " << endl;
  cin.getline(name, max_len);
  cout << "Please, enter your sex (type 'm' or 'f'): " << endl;
  cin.get(ch);
  if (ch == 'm')
    cout << "Hello, Mr " << name << '!' << endl;
  else
    cout << "Hello, Miss" << name << '!' << endl;
  return 0;
}
```

Результат роботи програми



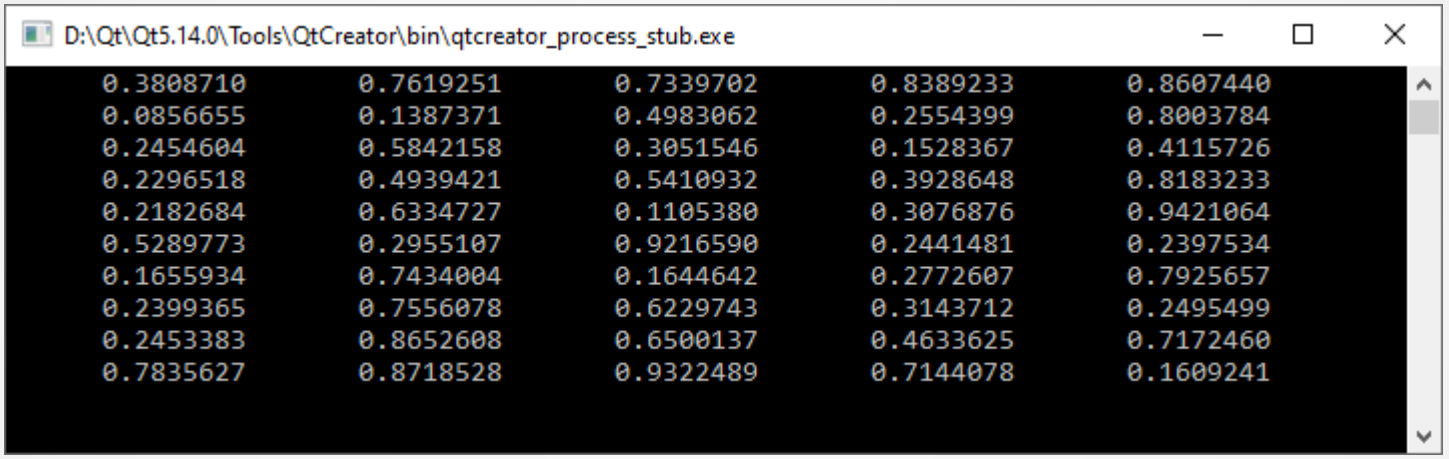
2. Маніпулятори. Форматований ввід-вивід

Приклад форматованого виводу за допомогою потокових маніпуляторів

```
#include <iostream>
#include <iomanip>
#include <ctime>

using namespace std;

int main ()
{
    double array[10][5];

    // Ініціалізація генератора випадкових чисел
    srand(unsigned(time(0)));
    // Заповнення масиву випадковими значеннями
    for (int i = 0; i < 10; i++)
        for (int j = 0; j < 5; j++)
            array[i][j] = double(rand()) /  RAND_MAX;
    // Форматований висновок масиву у вигляді таблиці
    for (int i = 0; i < 10; i++)
    {
        for (int j = 0; j < 5; j++)
            cout << fixed << setw(15) << setprecision(7) << array[i][j] << ' ';
        cout << endl;
    }
    return 0;
}
```

2. Маніпулятори. Форматований ввід-вивід

Результат роботи наведеної вище програми



3. Файловий ввід-вивід

У C++ є три основні потокові класи файлового вводу-виводу:
- **ifstream** – реалізація читання файлу (похідний від класу istream);
- **ofstream** - реалізація запису в файл (похідний від класу ostream);
- **fstream** – реалізація читання-запису файлу (похідний від класу iostream).

Для використання їх у програмі необхідно скористатися наступною директивою препроцесора:

```
#include <fstream>
```

Задекларувати потоки у програмі можна, наприклад, таким чином:

```
ifstream in;      // Файлове ввід
ofstream out;     // Файловий вивід
fstream io;       // Файловий ввід-вивід
```

Зв'язати потік із конкретним файлом можна двома способами.
1. За допомогою методу **open()**:

```
void open(const string &filename,  ios_base::openmode mode = ios_base::out);
```

Тут **filename** визначає ім'я файлу, а необов'язковий параметр **mode** режим відкриття файлу.

3. Файловий ввід-вивід

2. У конструкторі безпосередньо під час створення потокового об'єкта, наприклад, так:

ofstream out("data.txt");

У наведеному прикладі під час створення об'єкта out буде автоматично створено (перезаписано) файл під назвою "data.txt" (у поточній папці).

Перевірити успішність відкриття потоку можна за допомогою методу **is_open()**:

bool is_open() const;

У разі будь-якої помилки цей метод поверне значення **false**.

Після завершення роботи з файловим потоком його потрібно закрити. Робиться це за допомогою методу **close()**:

void close();

Примітка. При видаленні об'єкта потокового класу в деструкторі відкритий файл буде автоматично закритий. Однак правила хорошого тону в програмуванні вимагають явного виклику методу close().

3. Файловий ввід-вивід

Розглянемо наступний приклад виводу у файл текстової інформації.

```
#include <iostream>
#include <fstream>

using namespace std;

int main ()
{
    string file_name = "data.txt";
    ofstream out(file_name);

    if (!out.is_open())
    {
        cerr << "Error opening file: " << file_name << endl;
        return 1;
    }
    out << "Hello world!" << endl;
    out.close();
    return 0;
}
```

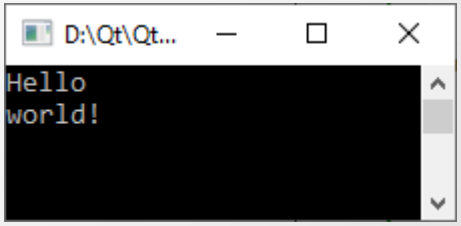
У даному прикладі для виведення у файл використовується вже знайома нам перевантажена операція виведення у потік “<<”.

3. Файловий ввід-вивід

Для читання з файлу текстової інформації скористаємося класом **ifstream**.

```
#include <iostream>
#include <fstream>

using namespace std;

int main ()
{
    string file_name = "data.txt",
           buffer;
    ifstream in(file_name);

    if (!in.is_open())
    {
        cerr << "Error opening file: " << file_name << endl;
        return 1;
    }
    while (!in.eof())
    {
        in >> buffer;
        if (in.fail())
            break;
        cout << buffer << endl;
    }
    in.close();
    return 0;
}
```

Результат роботи програми



Примітка. Тут метод eof() повертає значення true, коли досягнуто кінець файлу, а fail() – у разі виникнення помилки (наприклад, читання).

3. Файловий ввід-вивід

Для читання-запису двійкової інформації у файлах використовуються методи **read()** та **write()**.

istream &read (char *buffer, streamsize len);
ostream &write (const char *buffer, streamsize len);

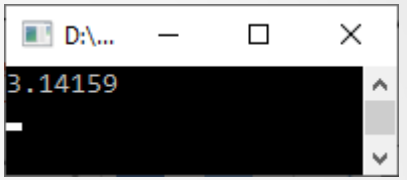
Тут **buffer** – покажчик на масив розмірністю щонайменше **len** байт. При застосуванні read() у buffer здійснюється зчитування len байт з вхідного потоку, а у випадку write() з області пам'яті, на яку вказує buffer, здійснюється виведення len байт у вихідний потік.

3. Файловий ввід-вивід

Приклад читання-запису бінарних даних

```
#include <iostream>
#include <fstream>

using namespace std;

int main ()
{
    string file_name = "data.dat";
    ofstream out;
    ifstream in;
    double pi = 3.14159;

    out.open(file_name, ios::binary);
    out.write((char*)&pi, sizeof(double));
    out.close();
    in.open(file_name, ios::binary);
    pi = 0;
    in.read((char*)&pi, sizeof(double));
    in.close();
    cout << pi << endl;
    return 0;
}
```

Результат роботи програми



Вміст файлу data.dat