Лекция 3

# ВВОД-ВЫВОД В ЯЗЫКЕ C++

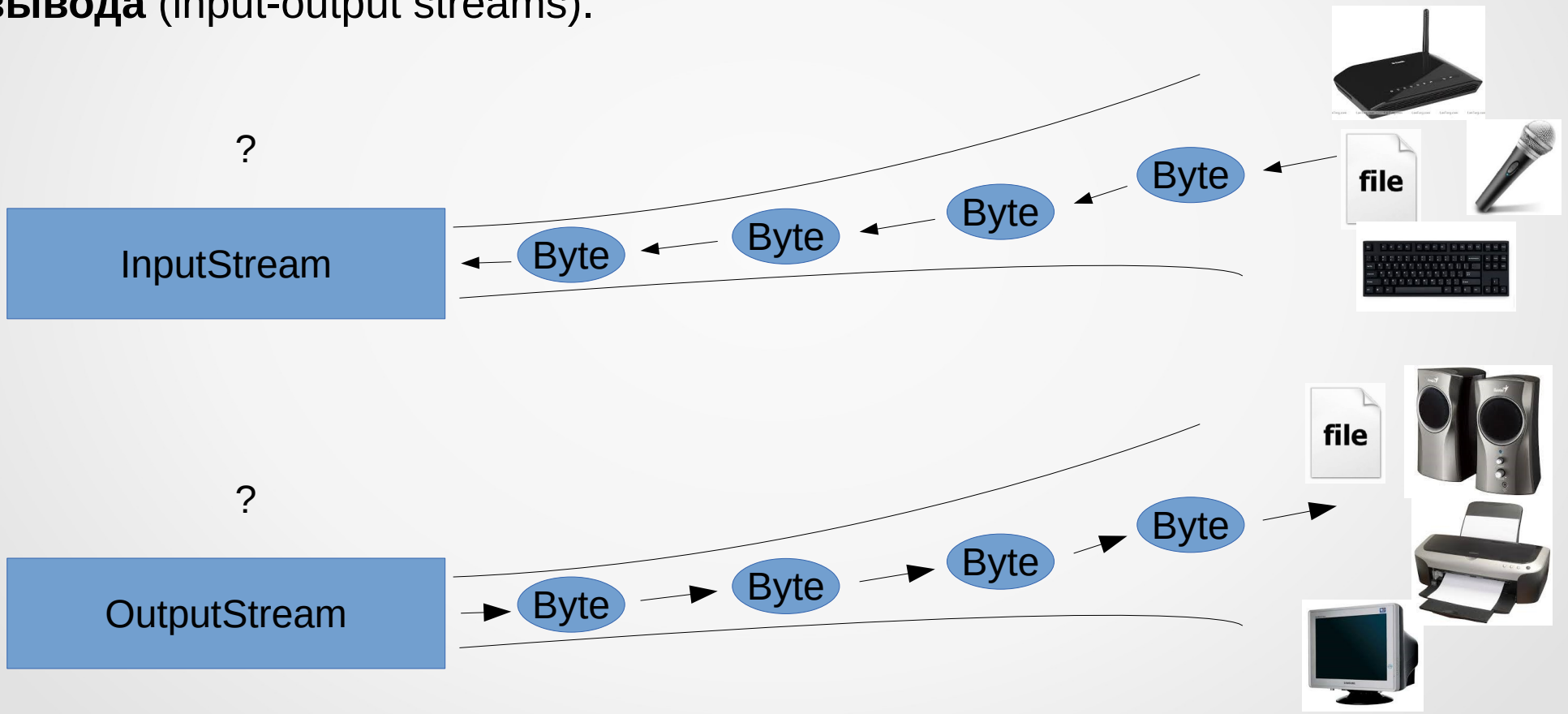
# Лекция 3. Ввод-вывод в языке C++

## План

1. Стандартные потоковые классы ввода-вывода.
2. Манипуляторы. Форматированный ввод-вывод.
3. Файловый ввод-вывод.

# 1. Стандартные потоковые классы ввода-вывода

В языке C++, как и в C, нет встроенных средств ввода-вывода. В C++ для этого используются специальные классы, называемые **потоками ввода-вывода** (input-output streams).



# 1. Стандартные потоковые классы ввода-вывода

**Потоки ввода-вывода** реализованы в библиотеке iostream, описание классов которой можно подключить к программе при помощи директивы

**#include <iostream>**

В iostream определены три стандартных потока:

- **cin** – стандартный поток ввода (аналог **stdin** в языке C);
- **cout** – стандартный поток вывода (аналог **stdout**);
- **cerr** – стандартный поток вывода сообщений об ошибках (**stderr** в C);

Эти потоки определены в **std** – стандартном пространстве имен C++. Для их использования нужно либо подключить это пространство директивой

**using namespace std;**

Либо же в явном виде указывать используемое пространство имен, например, **std::cout**.

# 1. Стандартные потоковые классы ввода-вывода

В классе, реализующем стандартный поток вывода (**ostream**) перегружена побитовая операция сдвига влево языка С («<<»). Здесь она используется для вывода информации (данных) в поток. Например, команда

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

выведет на экран (по-умолчанию) текст «x=10.5». Т. е. сначала в поток будет отправлена цепочка байтов, образующих строку «x=», а затем – число «10.5».

Аналогичным образом в классе, реализующем стандартный поток ввода (**istream**), перегружена операция «>>», имеющая теперь новый смысл: извлечение из входного потока данных. Например, в результате выполнения следующего кода:

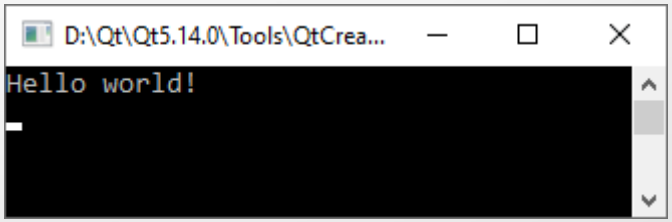
```
int a, b, c;

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

будет предпринята попытка последовательного извлечения из входного потока (клавиатуры) трех целых чисел и присвоения их значений переменным a, b и c соответственно.

# 1. Стандартные потоковые классы ввода-вывода

Пример использования стандартного потока вывода:

```
#include <iostream>

using namespace std;    // Подключение пространства имен std

int main()
{
    // Вывод в стандартный поток вывода (экран)
    cout << "Hello world!" << endl;
    return 0;
}
```



# 1. Стандартные потоковые классы ввода-вывода

Пример использования потоков с явным указанием пространства имен:

```
#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;
}
```

**Примечание. Вывод в поток std::endl приводит к переходу на новую строку.**

# 1. Стандартные потоковые классы ввода-вывода

Таким образом, вывод информации в поток осуществляется в общем виде следующим образом:

**cout << выражение_1 << выражение_2 << ... << выражение_n;**

где выражение_i – это некоторое выражение, переменная или константа.

Аналогично, ввод информации из потока осуществляется выражением вида:

**cin >> переменная_1 >> переменная_2 >>...>> переменная_n;**

**Примечание 1. При множественном вводе из потока вводимые данные должны быть разделены пробелом, символом табуляции ('\t') или символом конца строки ('\n').**

**Примечание 2. Для ввода строки, содержащей символы пробела, нужно использовать специальный манипулятор потока getline().**

# 2. Манипуляторы. Форматированный ввод-вывод

Для задания параметров форматирования потока в C++ используются специальные функции, называемые **манипуляторами** (manipulators), которые могут включаться в выражения ввода-вывода.

**Таблица 1. Наиболее распространенные потоковые манипуляторы**

| Манипулятор | Описание |
|---|---|
| endl | Помещение в выходной поток символа конца строки |
| dec | Вывод чисел в десятичном формате |
| oct | Вывод чисел в восьмеричном формате |
| hex | Вывод чисел в шестнадцатеричном формате |
| width(), setw() | Задание ширины поля вывода |
| fill(), setfill() | Заполнение пустых знакомест заданным значением символа |
| precision() | Задание количества значащих цифр в числе (или после запятой) в зависимости от использования fixed |
| fixed | Индикатор того, что установленная точность относится к количеству знаков после запятой |
| showpos | Вывод знака «+» для положительных чисел |

# 2. Манипуляторы. Форматированный ввод-вывод

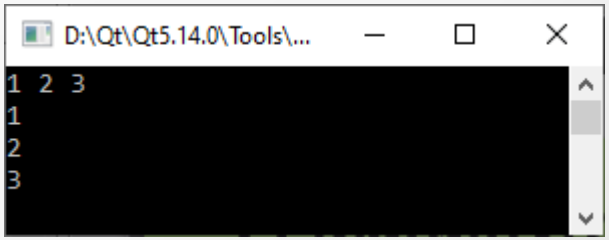
**Таблица 1 (продолжение)**

| Манипулятор | Описание |
|---|---|
| scientific | Вывод числа в научной нотации |
| get() | Считывание символа из входного потока |
| getline() | Считывание строки из входного потока |

**Пример использования манипулятора 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 работы программы**

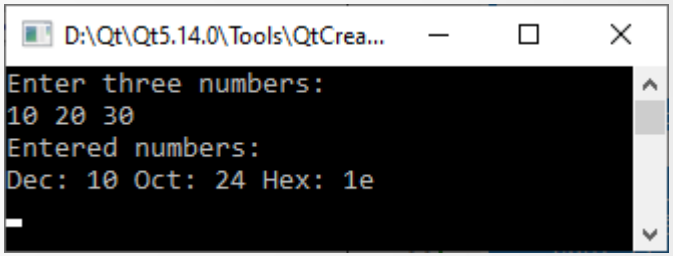


# 2. Манипуляторы. Форматированный ввод-вывод

**Примеры использования манипуляторов dec, oct и 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 работы программы**

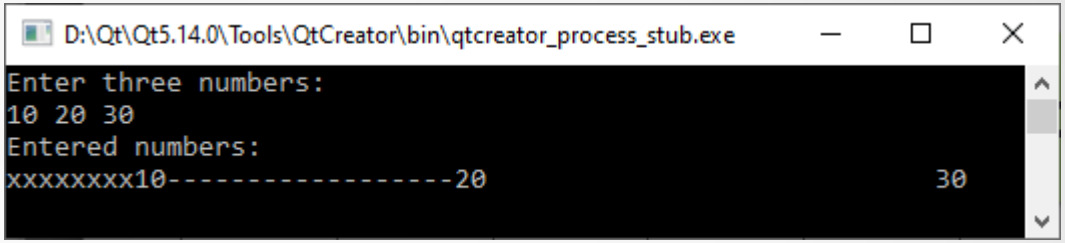


# 2. Манипуляторы. Форматированный ввод-вывод

**Примеры использования манипуляторов width(), setw(), fill() и 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 работы программы**



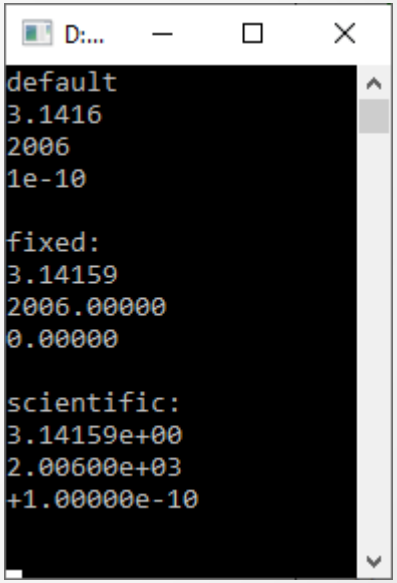

**Примечание. Для использования манипуляторов setw() и setfill() необходимо подключить заголовочный файл iomanip**

# 2. Манипуляторы. Форматированный ввод-вывод

**Примеры использования манипуляторов precision(), fixed, showpos и 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 работы программы**

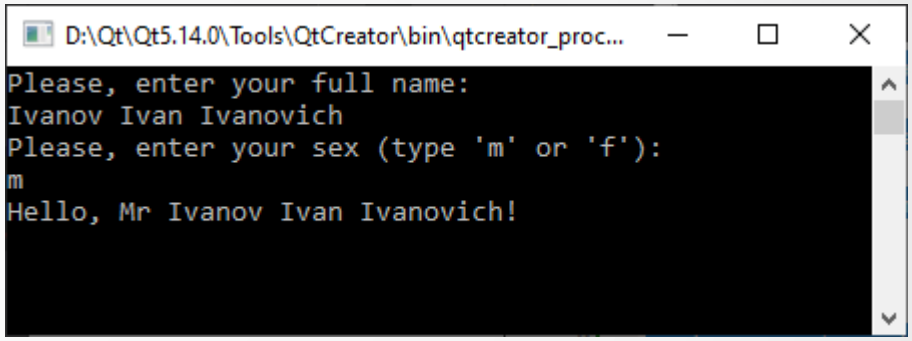


# 2. Манипуляторы. Форматированный ввод-вывод

**Примеры использования манипуляторов getline() и 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 работы программы**



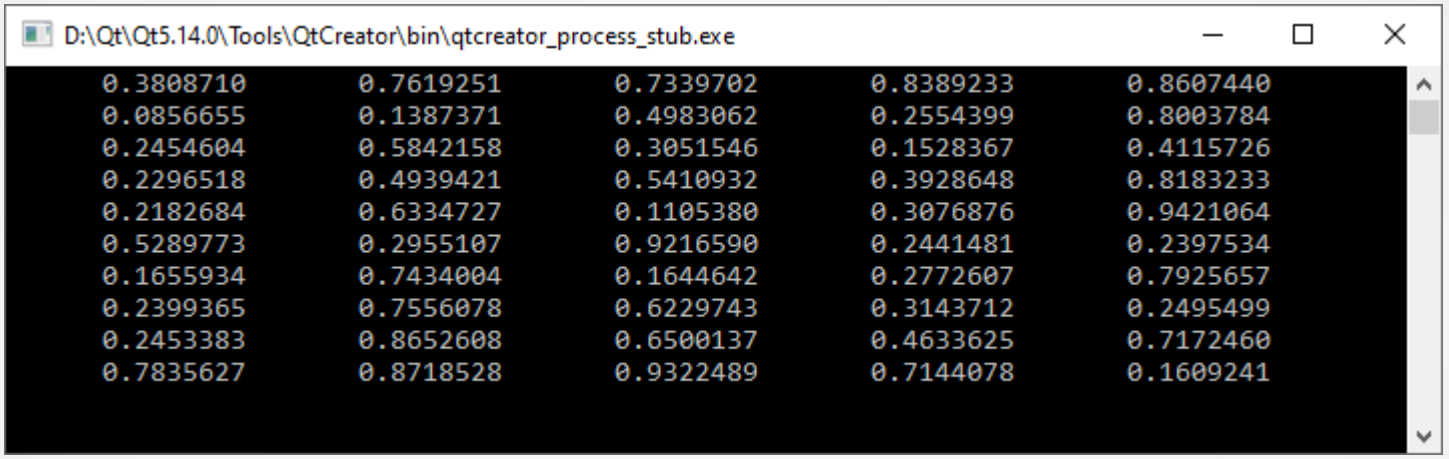
# 2. Манипуляторы. Форматированный ввод-вывод

**Пример форматированного вывода с помощью потоковых манипуляторов**

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

using namespace std;

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

    // Инициализация генератора случайных чисел
    srand(unsigned(time(0)));
    // Заполнение массива случайными значениями
    for (int i = 0; i < 10; i++)
        for (int j = 0; j < 5; j++)
            array[i][j] = double(rand()) /  RAND_MAX;
    // Форматированный вывод массива в виде таблицы
    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. Манипуляторы. Форматированный ввод-вывод

**Результат работы вышеприведенной программы**



# 3. Файловый ввод-вывод

В C++ есть три основных потоковых класса файлового ввода-вывода:
- **ifstream** – реализация чтения файла (наследник класса istream);
- **ofstream** – реализация записи в файл (наследник класса ostream);
- **fstream** – реализация чтения-записи файла (производный от класса iostream).

Для использования их в программе необходимо воспользоваться следующей директивой препроцессора:

```
#include <fstream>
```

Объявить потоки в программе можно, например, следующим образом:

```
ifstream in;        // Файловый ввод
ofstream out;       // Файловый вывод
fstream io;         // Файловый ввод-вывод
```

Связать поток с конкретным файлом можно двумя способами.
1. С помощью метода **open():**

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

Здесь **filename** задает имя файла, а необязательный параметр **mode** режим открытия файла.

# 3. Файловый ввод-вывод

2. В конструкторе непосредственно при создании потокового объекта, например, так:

**ofstream out("data.txt");**

В приведенном примере при создании объекта out будет автоматически создан (перезаписан) файл с названием "data.txt" (в текущей папке).

Проверить успешность открытия потока можно при помощи метода **is_open():**

**bool is_open() const;**

В случае какой-либо ошибки данный метод вернет значение **false**.

После завершения работы с файловым потоком его нужно закрыть. Делается это при помощи метода **close()**:

**void close();**

**Примечание. При удалении объекта потокового класса в деструкторе ранее открытый файл будет автоматически закрыт. Однако, правила хорошего тона в программировании требуют явного вызова метода close().**

# 3. Файловый ввод-вывод

Рассмотрим следующий пример вывода в файл текстовой информации.

```
#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;
}
```

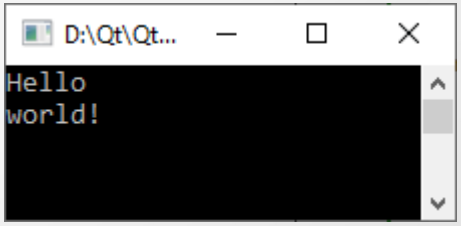
В данном примере для вывода в файл используется уже знакомая нам перегруженная операция вывода в поток «<<».

# 3. Файловый ввод-вывод

Для чтения из файла текстовой информации воспользуемся классом 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 работы программы**




**Примечание. Здесь метод eof() возвращает значение true, когда достигнут конец файла, а fail() – при возникновении ошибки (например, чтения).**

# 3. Файловый ввод-вывод

Для чтения-записи двоичной информации в файлах используются методы **read()** и **write()**.

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

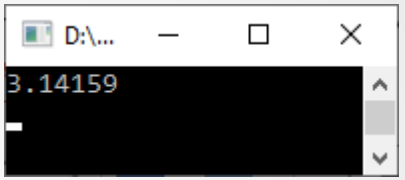
Здесь **buffer** – указатель на массив размерностью не менее **len** – байт. В случае read() в buffer осуществляется считывание len байт из входного потока, а в случае write() из области памяти, на которую указывает buffer, осуществляется вывод len байт в выходной поток.

# 3. Файловый ввод-вывод

**Пример чтения-записи двоичных данных**

```
#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 работы программы**




**Содержимое файла data.dat**