**08.05.2024 года Ввод/вывод данных в C++**

**Понятие потока. Стандартные потоки ввода/вывода. Форматирование данных при вводе/выводе*.***

**Понятие потока (***stream***)**

В языке программирования **С*++*** *система* *ввода/вывода* данных реализуетсячерез***потоки***. Функционал **потоков ввода/вывода** не определен как часть языка **C++**, а предоставляется стандартной библиотекой **C++** (т. е, находится в [пространстве имён **std**](https://ravesli.com/urok-53-prostranstva-imen/)).

**Поток –** это **абстрактный канал связи**(*логический интерфейс*), который создаётся в программе для обмена данными с внешними устройствами – их хранения и передачи от источника к приёмнику. Чтение/запись данных из потока/в поток – это ***извлечение*** изпотокаи ***вставка*** в поток соответственно**.**

Поток не зависит от конкретного устройства. Система ввода/вывода предоставляет **единый** удобный интерфейс для обмена данными между устройствами.

**Потоки** можно классифицировать **по** **виду устройств**, с которыми они связаны, например, *стандартные* *потоки –* обеспечивают *консольный*ввод/вывод (события, возникающие с *клавиатуры* или на э*кране*); *файловые потоки –* обеспечивают обменинформацией *с файлами* на внешних носителях.

**По направлению передачи данных** различают***потоки ввода*** (входные потоки) – для чтения данных;***потоки вывода***(выходные потоки) *–* для записи данных; ***потоки ввода/вывода*** *–* для чтения/записи данных.

**По способу организации** потоки делятся на ***два типа*** – *текстовые и двоичные* (бинарные)потоки.

**Текстовые потоки** – это *последовательность символов*. При вводе/выводе символов может происходить их преобразование согласно требованиям среды. Например, символ ***новой строки*** (**'\n'**) – может быть заменен парой символов **— *возврата каретки и перевода строки***. Поэтому может и **не быть однозначного соответствия** между символами в *памяти* и символами на внешнем *устройстве*.

**Двоичные потоки** – это *последовательность байтов*. При вводе/выводе не происходит преобразования символов. Двоичные потокипозволяют осуществить пересылку больших по объёму данных с высокой скоростью.

Вместо традиционных функций **Cи: printf()**, **scanf()** – для **в/ы** данных с консоли; **fprintf()**, **fscanf()** – для файлов, в языке программирования С++ используются **потоковые классы**, что даёт значительные преимущества. Одним из аргументов в *пользу потоков* является **простота использования**.

В **С++** стандартная библиотека ввода/вывода (*система ввода/вывода)* организована как **иерархия классов**. При подключении заголовочного файла <**iostream**> программа получает доступ ко всем [**класс**](https://ravesli.com/urok-113-klassy-obekty-i-metody-klassov/)**ам** библиотеки **iostream**, в основе которой находится класс **ios**, содержащий множество функций и переменных *для управления основными* ***операциями******над потоками***.

Класс **ios** является базовым по отношению к классам **istream, ostream** и **iostream,** которыепредставляют собой **классы потоков*****ввода, вывода и ввода/вывода***соответственно (***потоковые классы***).

В языке **C++** **поток** представляет собой ***объект*** некоторого ***потокового класса***.

**Стандартные потоки ввода/вывода**

Для реализации стандартного ввода/вывода стандартная библиотека **C++** по умолчанию предоставляет:

* **cin** *–* стандартный поток ввода (клавиатура);
* **cout** – стандартный поток вывода (экран);
* **cerr** – стандартный поток ошибок.

При вводе/выводе происходит **автоматический перевод** из числового типа данных в строку при выводе на экран и из строки в числовой тип при вводе с клавиатуры.

Для записи данных в поток применяется *операция вставки* (**<<**). Для чтения данных из потока – *операция извлечения*(**>>**). Например:

int a;

cin>>a; // ввод с клавиатуры

cout<<"a = "<<a<<endl; // вывод на экран

Операции ***извлечения***из потокаи***вставки*** в поток в качестве результата своего выполнения формируют соответственно ***ссылки*** на объект типа **istream** или **ostream**, что позволяет создавать *цепочки операций*:

int a, b, c;

cin>>a>>b>>c; // **5 6 7 или 5 6 7**

cout<<a<<b<<c<<endl;

Данные *при вводе* из потока должны разделяться ***пробелами*** (или другими пробельными символами – *табуляция, перевод строки, новая страница*).

*Числовые значения* можно вводить в **10**-ой или **16**-ой системе счисления (с префиксом **0х**), со знаком или без знака:

int n;

cout<<"Enter number – **10** c/c: "; cin>>n; // **26**

cout<<"Enter number – **16** c/c: "; cin>>hex>>n; // **0X1A** или **0x1a**

*Операции << и >>* определены для всех встроенных типов данных, что позволяет автоматически выполнять ввод/вывод в соответствии с типом величин, а также для некоторых классов стандартной библиотеки.

При вводе символьных массивов (строк) операцией >> извлечение символов из потока происходит до ближайшего пробельного символа, например, пробела. Вместо него в строку заносится *нуль-символ* '**\0**' – признак конца строки.

Однако часто пользовательский ввод всё же нужен со всеми пробельными символами. В этом случае можно использовать, например, функции **get()** или **getline()** класса **istream (см. тему 10, Пример 4).**

**Форматирование данных при вводе/выводе**

В потоковых классах форматирование можно выполнить с помощью **флагов*,* манипуляторов и специальных функций форматирования**.

**Флаги форматирования**. В **С++** каждый поток связан с набором *флагов форматирования*, которые управляют процессом форматирования. Если флаг формата установлен, реализуется соответствующая ему *функция*.

Флаги устанавливаются функцией **setf(),** а сбрасываются функцией **unsetf()**. Функции являются членами класса **ios.**

Наиболее часто используемые **флаги форматирования**:

**skipws** при вводе пробельные символы игнорируются;

**left** при выводе выравнивание по левому краю поля;

**rigth** при выводе выравнивание по правому краю поля;

**dec**  десятичная система счисления (по умолчанию);

**hex**  шестнадцатеричная система счисления;

**scientific** вывод вещественных чисел в форме мантиссы и порядка с

**6** знаками после точки;

**fixed** вывод вещественных чисел в обычной форме с **6**

десятичными знаками после точки;

**showpoint** вывод вещественных чисел с десятичной точкой и

дробной частью, по умолчанию всего **6** знаков;

Так как флаги форматирования являются определениями класса **ios**, обычно к ним обращаются через ***операцию разрешения области видимости класса* (::)** например, **ios::skipws**:

char str[20];

cout << "Enter char-array: ";

cin.setf(ios::skipws);

cin >> str; // введём, например,111

cout<<"str = "<<str<<endl; //str = 111 (без начальных пробелов)

**Манипуляторы форматирования.** Помимо флагов, изменяющих состояние потока, можно использовать *манипуляторы* (специальные функции), реализованные в заголовочных файлах **<istream>, <ostream> и <iomanip>.** Манипуляторы можно помещать в поток вместе с данными. Например, **endl**, который посылает символ перехода на новую строку в поток:

cout << "\*\*\*\*\*\*" << endl;

Многие манипуляторы повторяют названия флагов. Наиболее часто используемые **манипуляторы**:

**endl** при выводе перейти на новую строку;

**dec** вывод чисел в *десятичной* системе счисления;

**hex**  вывод целых чисел в **16**-ой системе счисления;

**setw**(int w) задаёт ширину поля вывода, равную **w**;

**ws** пропуск начальных пробелов;

**left** при выводе выравнивание по левому краю поля;

**right** при выводе выравнивание по правому краю поля;

**setfill(**char ch**)** устанавливает символ заполнения **ch**;

**setprecision(**int p**)** задаёт число знаков после десятичной точки,

равное **p** позициям, если вывод в форме мантиссы

и порядка. Иначе – общее количество цифр.

Манипулятор **endl** осуществляет переход на новую строку и производит **сброс буфера вывода** (следовательно и всех форматов при использовании манипуляторов). Поэтому, если требуется **сохранить форматы** при переходе на новую строку, нужно использовать управляющий символ **'\n'**.

Манипуляторы действуют только **на данные**, которые следуют **за ними** в потоке, а не на данные, которые находятся перед ними.

**Функции форматирования**.Также в классе **ios** определены **функции-члены**, которые устанавливают параметры формата, например:

* **width(**int len) – устанавливает ширину поля вывода;
* **precision(**int num**) –** устанавливает точность при выводе, т.е. число цифр после

точки, если вывод в виде мантиссы и порядка, иначе – общее количество цифр;

* **fill(**char ch**) –** задаёт символ заполнения поля вывода:

int x = 1234567890; int y = 45;

cout.setf(ios::right); cout.width(20);

cout.fill(**'\*'**); cout<<x<<endl; // \*\*\*\*\*\*\*\*\*\*1234567890

cout.width(20); cout<<y<<endl; // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*45

// **Пример 1.** Форматирование данных при выводе.

#include <iostream>

using namespace std;

#include <iomanip>

int **main()**{

cout<<123.23<<" "<<"ABCDEF"<<" "<<100<<endl; // по умолчанию

// меняем формат с помощью флагов форматирования

cout.unsetf(ios::dec); // требуется не для всех компиляторов

cout.setf(ios::hex|ios::scientific); // вывод в форме мантиссы и порядка

cout<<123.23<<" "<<"ABCDEF"<<" "<<100<<endl;

// использование манипуляторов форматирования

cout<<hex<<100<<" "; // вывод в **16**-ой с/с

cout<<oct<<10<<endl; // вывод в **8**-ой с/с

cout<<setw(15)<<"ABCDEF"<<endl; // выравнивание вправо

cout<<setprecision(2)<<1234.5678<<endl;

// использование функций форматирования

cout<<left<<"ABCDEF"<<endl; // выравнивание влево

cout.width(10);

cout.precision(4);

cout<<123.456789<<endl;

}

**Результат:**

123.23 ABCDEF 100

1.232300e+02 ABCDEF 64

64 12

ABCDEF // выравнивание вправо по умолчанию, ширина поля **15**

1.23e+03

ABCDEF

1.2346e+02