# *Система ввода-вывода C++*

Система ввода-вывода в C++ действует через потоки (streams).

Поток ввода-вывода – это логическое устройство, которое выдает и принимает информацию пользователя. Поток связан с физическим устройством с помощью системы ввода-вывода C++. Когда запускается программа на C++ автоматически открываются четыре потока:

|  |  |  |
| --- | --- | --- |
| Поток | Значение | Устройство |
| cin | стандартный ввод | клавиатура |
| cout | стандартный вывод | экран |
| cerr | стандартная ошибка | экран |
| clog | буферируемая версия cerr | экран |

В C++ ввод-вывод обеспечивается подключением к программе заголовочного файла <iostream>. В этом файле определены наборы иерархий классов, поддерживающие операции ввода-вывода. Классы ввода-вывода начинаются с системы классов-шаблонов.

Система ввода-вывода строится на двух связанных, но различных иерархиях классов-шаблонов.

Первая является производной от класса нижнего уровня basic\_streambuf.

Вторая является производной от класса basic\_ios. Это класс ввода-вывода верхнего уровня, который обеспечивает форматирование, контроль ошибок и информацию о состоянии потока ввода-вывода.

Класс basic\_ios является базовым для производных классов basic\_istream, basic\_ostream, basic\_iostream.

|  |  |
| --- | --- |
| **Класс** | **Назначение** |
| basic\_ios | Обеспечивает операции ввода-вывода общего назначения |
| basic\_istream | Поддерживает операции ввода |
| basic\_ostream | Поддерживает операции вывода |
| basic\_iostream | Поддерживает операции ввода-вывода |
| basic\_ifstream | Поддерживает файловые операции ввода |
| basic\_ofstream | Поддерживает файловые операции вывода |
| basic\_fstream | Поддерживает файловые операции ввода-вывода |
| basic\_istringstream | Поддерживает строковые операции ввода |
| basic\_ostringstrearm | Поддерживает строковые операции вывода |
| basic\_stringstream | Поддерживает строковые операции ввода-вывода |

#### Иерархия классов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ios | -> | istream | -> | ifstream istringstream |
|  | -> | ostream | -> | ofstream ostringstream |
|  | -> | iostream | -> | iofstream iostringstream |

#### [Класс ios](http://mycpp.ru/cpp/scpp/cppd_iocpp_ios.htm)

Назначение: обеспечивает операции ввода-вывода общего назначения  
Заголовок: <iostream>  
Иерархия: базовый класс классов ввода-вывода

#### [Класс ostream](http://mycpp.ru/cpp/scpp/cppd_iocpp_ostream.htm)

Назначение: поддерживает операции вывода  
Заголовок: <iostream>  
Иерархия: ios -> ostream

#### [Класс istream](http://mycpp.ru/cpp/scpp/cppd_iocpp_istream.htm)

Назначение: поддерживает операции ввода  
Заголовок: <iostream>  
Иерархия: ios -> istream

##### ***Методы класса ios***

bad — Проверка потока на ошибку   
clear— Очищает флаги состояния, связанные с потоком   
eof — Проверка конца файла  
exceptions — Работа с флагами которые вызывают исключения   
fail — Проверка на присутствие ошибки ввода-вывода  
fill — Проверка на присутствие ошибки ввода-вывода  
flags — Работа с флагами   
good — Проверка на отсутствие ошибок ввода-вывода   
precision — Возвращает или устанавливает текущее значение точности  
rdstate — Возвращает состояние соответствующего потока  
setf — Устанавливает флаги форматирования  
setstate — Устанавливает состояние соответствующего потока  
sync\_with\_stdio — Позволяет использовать стандартную С-подобную систему ввода-вывода  
unsetf — Очистка одного или нескольких флагов форматирования   
width — Работа с шириной поля

**Форматированный ввод-вывод**

Каждый поток ввода-вывода связан с набором флагов формата (format flags). Флаги управляют способом форматирования информации и представляют собой битовые маски (bitmasks). Эти маски объявлены в классе ios как данные перечислимого типа fmtflags.

Набор флагов формата – тип **fmtflags**:

|  |  |
| --- | --- |
| **Флаг** | **Назначение** |
| **boolalpha** | значения булевского типа выводятся в виде true и false; |
| **dec** | вывод в десятичном виде; |
| **fixed** | вывод числа с плавающей точкой; |
| **hex** | вывод в шестнадцатеричном виде; |
| **internal** | для заполнения поля вывода происходит вставка пробелов между знаком числа и его первой цифрой; |
| **left** | выравнивание по левому краю (флаг установлен); |
| **right** | выравнивание по правому краю; |
| **oct** | вывод в восьмеричном виде; |
| **scientific** | числа с плавающей точкой выводятся в научной нотации; |
| **showbase** | вывод основания системы счисления; |
| **showpoint** | вывод десятичной точки и незначащих нулей при отображении десятичных чисел; |
| **showpos** | вывод знака перед положительным числом; |
| **skipws** | установлен – при вводе данных из потока разделители (пробелы, знаки табуляции, переход на новую строку) игнорируются. |
| **unitbuf** | установлен – после каждой операции вставки буфер очищается; |
| **uppercase** | прописные буквы **e** и **x** при выводе в научном формате и в шестнадцатеричной системе счисления. |

**Замечание**: По умолчанию используется выравнивание по - правому краю и данные выводятся в десятичной системе счисления.

**Замечание**: Поля **dec, oct, hex** называют общим именем **basefield.**

Поля **left, right, internal** называют **adjustfield.**

***Установка флага***

Для установки флага формата используют функцию **setf()**. Эта функция является членом класса **ios**, и влияет на конкретный поток.

В одном вызове можно установить сразу несколько флагов (используют оператор OR).

***Основная форма:***

fmtflags setf(fmtflags флаги);

***Сброс флагов***

Противоположной к функции setf() является функция unsetf(). Эта функция-член класса ios сбрасывает один или несколько флагов формата.

***Основная форма:***

void unsetf(fmtflags флаги);

**Пример**: // *Установка флагов*

#include<iostream>

using namespace std;

int main()

{

int iVar = 100;

cout << iVar << "\n";

cout.unsetf(ios::dec);

cout.setf(ios::hex | ios::showbase);

cout << iVar << "\n";

cout.unsetf(ios::hex);

cout.setf(ios::oct);

cout << iVar << "\n";

cout.unsetf(ios::oct);

cout << iVar << "\n";

system("pause");

return 0;

}

Результаты выполнения программы:

100

0x64

0144

100

**Замечание**: Обратить внимание на использование функции **unsetf().**

**Перегруженная форма функции setf()**

Функция **setf()** имеет перегруженную форму:

fmtflags setf(fmtflags флаги1, fmtflags флаги2);

В этом случае изменяются только флаги, заданные параметром *флаги2*. Сначала они сбрасываются, а затем устанавливаются в соответствии с со статусами флагов, заданных параметром *флаги1*.

**Пример**: *// Перегруженная форма. Результат, как и в предыдущем примере.*

#include<iostream>

using namespace std;

int main()

{

int iVar = 100;

cout << iVar << "\n";

cout.setf(ios::showbase);

cout.setf(ios::hex, ios::basefield);

cout << iVar << "\n";

cout.setf(ios::oct, ios::basefield);

cout << iVar << "\n";

cout.setf(ios::dec, ios::basefield);

cout << iVar << "\n";

system("pause");

return 0;

}

**Функции width(), precision() и fill()**

Кроме флагов формата в классе ios определены три функции-члена. Эти функции устанавливают параметры формата: ширину поля, точность и символ заполнения: **width(), precesion(), fill().**

По умолчанию при выводе любого значения, оно занимает столько позиций, сколько выводится. Однако, с помощью функции **width()** можно задать минимальную ширину поля.

Прототип функции:

**streamsize width(streamsize w);**

Тип данных **streamsize** определен в заголовочном файле<iostream>.

По умолчанию при выводе значений с плавающей точкой точность равна шести знакам. Однако с помощью функции **precesion()** это число можно изменить.

Прототип функции **precesion()**

**streamsize precesion(streamsize p);**

Точность (число цифр после запятой) задается параметром **p**, а возвращает функция прежнее состояние точности.

По умолчанию, если требуется заполнить свободное поле, используются пробелы. Однако, с помощью функции **fill()**, можно изменить символ заполнения.

Прототип функции **fill()**:

**char fill(char ch);**

После вызова функции **fill()** символ **ch** становится новым символом заполнения.

***Пример:***

#include<iostream>

using namespace std;

int main()

{

// cout.width(10);

cout << 12.12345 << "\n";

cout.precision(4);

cout.width(10);

cout << 12.12345 << "\n";

cout.fill('\*');

cout.width(10);

cout << 12.12345 << "\n";

system("pause");

return 0;

}

**Манипуляторы ввода-вывода**

В системе ввода-вывода имеется еще один способ форматирования информации. Это использование специальных функций—манипуляторов ввода-вывода.

Манипуляторы, в отличие от функций-членов класса ios, могут располагаться внутри инструкции ввода-вывода.

Для доступа к манипуляторам с параметрами (таким, как функция setw()), необходимо включить в программу заголовок <iomanip>. В этом заголовке нет необходимости при использовании манипуляторов без параметров.

**Пример**:

cout<<oct<<100<<hex<<100;

cout<<setw(10)<<100; // ширина поля = 10, 16-ый формат

|  |  |  |
| --- | --- | --- |
| Манипулятор | Назначение | Ввод/вывод |
| dec | установка флага dec | ввод/вывод |
| endl | вывод символа новой строки и очистка потока | вывод |
| hex | установка флага hex | вывод |
| left | установка флага left |  |
| oct | установка флага oct |  |
| right | установка флага right |  |
| setbase(int основание) | задание основания системы счисления |  |
| setfill(int ch) | задание символа заполнения |  |
| setiosflags(fmtflags f) | установка флагов, заданных параметром f |  |
| setprecesion(int p) | задание числа цифр точности равным p |  |
| setw(int w) | задание ширины поля равным w позиций |  |

Если с помощью манипулятора надо установить конкретные флаги формата, надо использовать функцию setiosflags(). Этот манипулятор реализует ту же функцию, что и функция член setf().

Для сброса флагов формата надо использовать манипулятор resetiosflags(). Он эквивалентен функции unsetf().

***Перегрузка операторов "<<" и ">>"***

Оператор "**<<**" – ***оператор вставки*** (вставляет данные в поток).

Оператор "**>>**" – ***оператор извлечения*** (извлекает данные из потока).

***Шаблон создания пользовательской функции вставки:***

ostream &operator<<(ostream &stream, имя\_класса &ob)

{

*// Тело функции вставки*

return stream;

}

**Замечание**: Функция вставки не является членом класса, поэтому, для того чтобы она могла работать с закрытыми членами класса, необходимо объявить ее дружественной этому классу.

***Шаблон создания пользовательской функции извлечения:***

istream &operator>>(istream &stream, имя\_класса &ob)

{

*// Тело функции извлечения*

return stream;

}

***При мер***: *Перегруженный ввод и вывод комплексного числа*

#include <iostream>

using namespace std;

class complex

{

double re, im;

public:

complex(double, double);

complex(): re(0.0), im(0.0) {}

friend ostream& operator << (ostream &stream, complex &ob);

friend istream& operator >> (istream &stream, complex &ob);

};

ostream &operator << (ostream &stream, complex &z)

{

cout << "Вывод комплексного числа: ";

stream << z.re << " " << z.im << endl;

return stream;

}

istream &operator >> (istream &stream, complex &z)

{

cout << "Ввод комплексного числа: \n";

stream>>z.re;

stream >> z.im;

cout << endl;

return stream;

}

int main()

{

setlocale(LC\_ALL, "Russian");

complex z;

cin >> z;

cout<< z;

system("pause");

return 0;

}