## **Форматирование ввода-вывода средствами языка С.**

### **Функция printf()**

Сегодня мы познакомимся с основными возможностями функции **printf()**. Эта функция появилась в языке программирования C для организации форматного вывода информации. В связи с тем, что C++ был создан на базе языка C, использование этой функции в программах на C++ в некоторых случаях бывает более удобным, чем применение стандартных объектов **cout** и **cin**. В связи с тем, что С++ не за горами, в данном уроке мы перечислим основные особенности использования этой функции.

**Примечание:** Прототип этой функции находится в заголовочном файле stdio.h.

Формат, указываемый при обращении к функции **printf()**, выглядит следующим образом:

|  |  |
| --- | --- |
| printf (Управляющая\_строка,Аргумент1,Аргумент2,...); | |

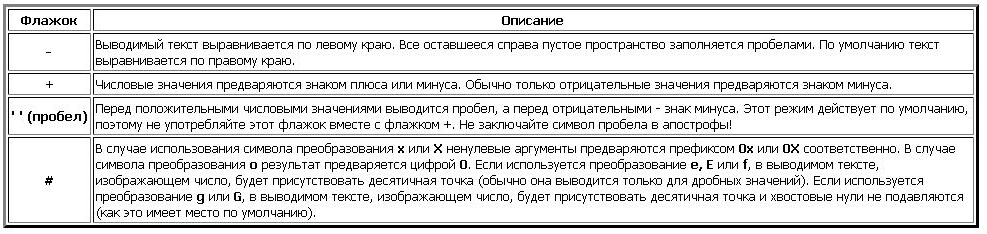
Самым сложным в использовании этой функции является ***Управляющая\_строка*** , задающая формат вывода данных. Помимо обычного текста она содержит ***команды преобразования***, которые начинаются с символа **"%"**, за которым следуют символы и цифры, задающие правила вывода аргументов (***Аргумент1, Аргумент2,...***). Формат определяется следующим образом:

|  |  |
| --- | --- |
| %[флажки][ширина][.точность][F|N|h|l|L] преобразование | |

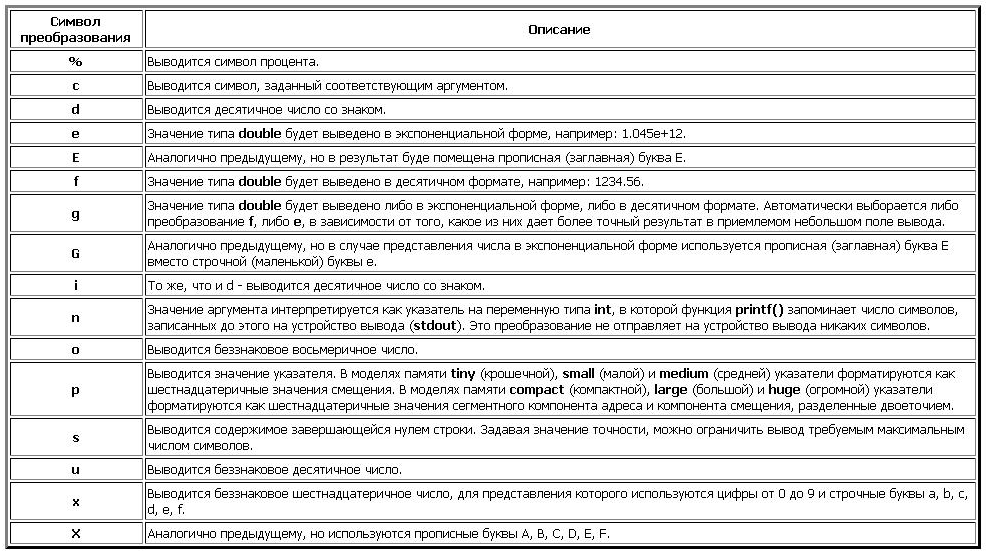
**Примечание:** Отметим, что конструкции в квадратных скобках могут отсутствовать, а символ "|" (вертикальная черта) свидетельствует о том, что здесь может использоваться одна из перечисленных возможностей.

##### **Элементы формата.**

1. **Формат** (команда преобразования) начинается с обязательного знака процента. Чтобы вставить в выводимый текст сам знак процента, его нужно указать дважды: **%%**.
2. **Флажки** задают правила выравнивания, знаки "+" и "-", десятичную точку, хвостовые нули и префиксы для восьмеричных и шестнадцатеричных значений. Флажки не являются обязательными. Если они присутствуют, то могут состоять из одного или нескольких указанных в таблице ниже символов.



1. **Ширина** - задает минимальную ширину поля вывода (в символьных позициях).
   * Любое свободное пространство обычно заполняется пробелами
   * Но если значение ширины начинается с цифры **0**, эти добавляемые пробелы заменяются нулями.
   * В качестве ширины можно также указать символ \* ("звездочка"), что приводит к использованию значения следующего аргумента типа **int** в качестве ширины поля.
   * В этом случае задается два значения: целое значение, задающее минимальную используемую ширину, и значение, выводимое в этом поле.
   * Значение, задающее ширину, должно располагаться ***перед*** выводимым значением.
   * Если указана нулевая ширина, то вывод производится в поле переменной ширины и дополняется ведущими нулями.
   * Можно задавать меньшую ширину, чем требуется, для представления значения.
   * Ширина поля вывода увеличивается по мере необходимости, чтобы обеспечить вывод значения без отсечения.
2. **Точность** - если в этом месте форматной строки присутствует точка, то следующее за ней значение представляет точность, используемую для представления форматируемого результата. Интерпретация точности зависит от типа форматируемого элемента.
   * За точкой должно следовать целое значение. Это значение по умолчанию равно 0, что означает отсутствие компонента точности.
   * Для символов преобразования **d, i, o, u, x** и **X** по умолчанию это значение равно 1.
   * Для символов **e, E** и **f** по умолчанию это значение равно 6.
   * Для **g** и **G** - переменному числу значащих цифр.
   * Для символов **s** и **c** - полному числу символов.
   * Для символов преобразования **g** и **G** точность представляет максимальное число значащих цифр сформатированного результата.
   * Для преобразований **e, E** и **f** точность равна числу используемых десятичных позиций, а последняя цифра округляется.
   * В случае использования **s** точность указывает максимальное число используемых символов строки.
   * Точность никак не влияет на результат с использованием символа преобразования **c** - всегда выводится один символ.
   * Для преобразований **d, i, o, u, x** и **X** выводится столько цифр, сколько указано точностью, дополненных, при необходимости, слева цифрами 0
3. Один из нескольких ***модификаторов*** позволяет указать характеристики, связанные с размером выводимого значения. Требуемый тип данных выбирается соответствующим ***символом преобразования:***
   * **F** - дальний (far-) указатель;
   * **N** - ближний (near-) указатель;
   * **h** - значение типа short int;
   * **l** - значение типа long;
   * **L** - значение типа long double.
4. ***Преобразование*** - задает тип соответствующего аргумента. Программист сам несет ответственность за правильность типа этого аргумента. Например, если используется символ преобразования **g**, то аргумент должен быть значением с плавающей точкой. В таблице 2 перечислены все допустимые символы преобразования, причем в каждой отдельной команде можно использовать ***только один*** из указанных символов.



### **Функция scanf().**

А, теперь, рассмотрим основные особенности применения функции **scanf()**, предназначенной для ввода данных. Так же, как и для функции **printf()**, для функции **scanf()** указываются управляющая строка и следующий за ней список аргументов. Обращение к этой функции имеет вид:

scanf (Управляющая\_строка,&Имя1,&Имя2,...,&ИмяN);  
  
Имя1,Имя2,...,ИмяN - это имена переменных, значения которых надо ввести. Смысл символа "&" перед именем будет раскрыт ниже.  
Управляющая\_строка - это строка символов, которая задает количество и типы вводимых переменных. Делается это так: в формате указывается символ %, за которым следует буква, определяющая тип вводимой переменной.

Сочетание ***%буква*** является ***спецификацией преобразования***. При вводе мы будем пользоваться следующими спецификациями:

* **%d** - ввести целое число;
* **%c** - ввести один символ;
* **%s** - ввести строку символов.

Спецификации преобразования должны соответствовать количеству и типу вводимых переменных.

При применении функции **scanf()** мы. Необходимо запомнить два правила.

1. Если нужно ввести некоторое значение и присвоить его переменной одного из основных типов, то перед именем переменной требуется писать символ &, т. е передавать переменную в функцию по ссылке.
2. Если требуется ввести значение ***строковой переменной***, то использовать символ **&*не нужно***.

При обращении к функции **scanf()** выполнение программы приостанавливается, и нужно ввести значения для указанных переменных, после чего работа программы продолжается.

Остановимся более подробно на содержимом управляющей строки. Она содержит спецификации преобразования, которые используются для непосредственной интерпретации входной последовательности.

Управляющая строка может содержать:

* пробелы ' ', символы табуляции '\t' и перевода строки '\n', которые просто игнорируются;
* обычные символы (кроме символа "%"), которые предполагаются совпадающими с очередными (отличными от символов, перечисленных в предыдущем пункте) символами входного потока;
* ***спецификации преобразования***, состоящие из
  + %,
  + необязательного символа "подавления" присваивания "\*",
  + необязательного числа, задающего максимальную ширину поля и
  + символа преобразования.

***Спецификация преобразования*** управляет преобразованием очередного входного поля. Результат обычно помещается в переменную, на которую ссылается соответствующий параметр. Однако, если присутствует символ "\*", то входное поле просто пропускается, и никакого присваивания не производится.

Перечислим основные символы преобразования:

* **d** - десятичное число;
* **x** - шестнадцатеричное число;
* **o** - восьмеричное число;
* **c** - символ;
* **s** – строка символов

### **Пример**

|  |
| --- |
| #include <iostream> #include<stdio.h> using namespace std;  void main () {  int data,month,year;  char name[15],town[15];    printf ("What is your name? ");  scanf ("%s",name);  printf ("Enter day, month and year of your birthday.\n Date (20 12 2004): ");  scanf ("%d %d %d",&data,&month,&year); // ввод через пробел  printf ("Where do you live?");   scanf ("%s",town);  printf ("Now, we know you!!!\n");  printf ("Your name is %s .\n",name);  printf ("You live in %s your birthday (%d.%d.%d)\n\n",town,data,month,year); } |

## **Форматирование ввода-вывода средствами языка С++.**

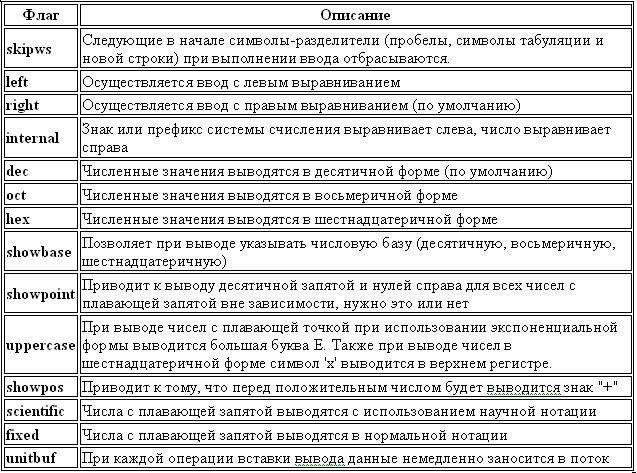
В предыдущей теме урока мы рассмотрели, как можно использовать функцию printf() для форматирования текста, а можно сделать то же самое при помощи средств форматирования ввода/вывода языка С++. Мы рассмотрим целых два способа.

### **Способ первый**

Данный способ предусматривает использование функций для установки определенных флагов форматирования, которые перечислены в ios\_base

|  |
| --- |
| enum {  skipws=0x0001, left=0x0002, right=0x0004, internal=0x0008, dec=0x0010, oct=0x0020, hex=0x0040, showbase=0x0080, showpoint=0x0100, uppercase=0x0200, showpos=0x0400, scientific=0x0800, fixed=0x1000, unitbuf=0x2000, stdio=0x4000  }; |

Рассмотрим назначение каждого из флагов.



Кроме флагов, существует так же набор функций, которые устанавливают и сбрасывают эти флаги.

1. **long setf(long flags);** - используется для установки флагов и принимает в качестве параметра рассмотренные выше флаг или флаги, соединенные между собой при помощи побитового ИЛИ. Она возвращает предыдущее значение флага. Рассмотрим пример:

|  |
| --- |
| cout.setf(ios::hex); cout.setf(ios::showpos); cout<<123<<" "<<123.45<<"\n";  cout<<67<<" "<<678.9<<"\n"; выведет на экран  +123 +123.45  +67 +678.9  точно такого же эффекта мы добьемся после выполнения следующего кода cout.setf(ios::hex | ios::showpos); cout<<123<<" "<<123.45<<"\n";  cout<<67<<" "<<678.9<<"\n"; |

**2. long unsetf(long flags);** - функция возвращает значение предыдущей установки флага и сбрасывает флаги, определяемые параметром flags. Пример:

|  |
| --- |
| cout.setf(ios::showpos | ios::hex); cout<<123<<" "<<123.45<<"\n"; cout.unsetf(ios::showpos | ios::hex); cout<<123<<" "<<123.45<<"\n"; выведет на экран  +123 +123.45  123 123.45 |

Кроме флага форматирования также можно установить ширину поля потока, символ для заполнения и число цифр после десятичной запятой. Для этого используются следующие функции:

**int width(int len);** - устанавливает ширину поля и возвращает текущую ширину. Данная функция воздействует только на один вывод информации.

**char fill(char ch);** - устанавливает текущий символ заполнения и возвращает предыдущий символ заполнения. По умолчанию используется пробел

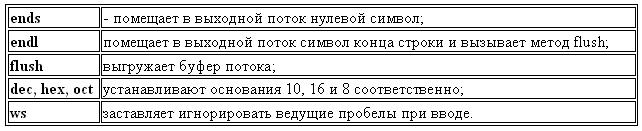
**int precision(int num);** - устанавливает точность чисел с плавающей точкой. Эта функция определяет количество цифр после десятичной запятой или обозначает количество выводимых цифр. Точность, установленная по умолчанию, равна 6.

|  |
| --- |
| cout.width(10); cout.fill('\*'); cout<<123<<"\n"; cout.width(10); cout<<456<<"\n\n";  cout<<"Using precision\n"; float price1 = 20.405f; float price2 = (float)1.9+8.0/9.0; cout<<price1<<"\n"<<price2<<"\n\n"; cout.precision(2); cout<<price1<<"\n"<<price2<<"\n\n"; cout.setf(ios::scientific); cout<<price1<<"\n"<<price2<<"\n\n";  выведет на экран   \*\*\*\*\*\*\*123  \*\*\*\*\*\*\*456   Using precision  20.405  2.78889   20  2.8   2.04e+001  2.79e+000 |

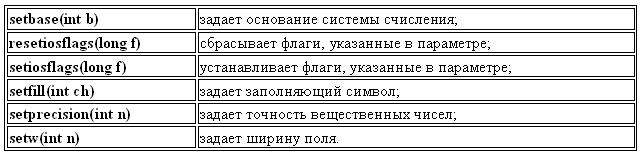
### **Второй способ. Использование манипуляторов.**

**Манипуляторы** - специальные функции, которые позволяют изменять флаги потока. Существуют манипуляторы с параметрами и без. Если Вы используете манипуляторы с параметрами, нужно подключить файл *iomanip.h*.

**Манипуляторы без параметров:**



**Манипуляторы с параметрами:**

****

##### **Пример использования:**

|  |
| --- |
| **cout<<setw(5)<<setfill('\*')<<22<<endl; cout<<hex<<11<<endl; Результат работы:  \*\*\*22  b** |