Лекция 2. Общие сведения о языке С++

В этой лекции будут рассмотрены основные элементы языка С++: алфавита, переменных, констант, типов данных, операций, стандартных функций. Здесь же будут рассмотрены структура программы и средства ввода вывода данных.

2.1. Алфавит языка

Программа на языке С++ может содержать следующие символы:

- прописные, строчные латинские буквы A, B, C, & , x, y, z и знак подчеркивания;
- арабские цифры от 0 до 9;
- специальные знаки: { } , |, [ ] ( ) + / % * . \ : ? < = > ! & # ~ ; ^
- символы пробела, табуляции и перехода на новую строку.

Из символов алфавита формируют ключевые слова и идентификаторы. *Ключевые слова* это зарезервированные слова, которые имеют специальное значение для компилятора и используются только в том смысле, в котором они определены (операторы языка, типы данных и т.п.). *Идентификатор* - это имя программного объекта, представляющее собой совокупность букв, цифр и символа подчеркивания. Первый символ идентификатора буква или знак подчеркивания, но не цифра. Идентификатор не может содержать пробел. Прописные и строчные буквы в именах различаются, например, ABC, abc, Abc три различных имени. Каждое имя (идентификатор) должно быть уникальным в пределах функции и не должно совпадать с ключевыми словами.

В тексте программы можно использовать *комментарии*. Если текст начинается с двух символов «косая черта» // и заканчивается символом перехода на новую строку или заключен между символами /* и */, то компилятор его игнорирует. Комментарии удобно использовать как для пояснений к программе, так и для временного исключения фрагментов программы при отладке.

2.2. Данные в языке С++

Для решения задачи в любой программе выполняется обработка каких-либо данных. *Данные* могут быть различных типов: целые и вещественные числа, символы, строки, массивы. Данные в языке С++ принято описывать в начале функции. Обязательное описание типов данных позволяет компилятору контролировать допустимость программных кодов.

2.2.1. Типы данных

Типы данных определяют способ хранения чисел или символов в памяти компьютера. Они задают размер ячейки, в которую будет записано то или иное значение, определяя тем самым его максимальную величину или точность задания. Типы языка С++ можно разделить на основные и составные.

К *основным типам данных* языка относят:

- char символьный ;
- int целый ;
- float с плавающей точкой ;
- double двойной точности ;
- bool логический .

Для формирования других типов данных используют основные типы и так называемые спецификаторы. Типы данных созданные на базе стандартных типов с использованием спецификаторов называют *составными типами данных*. В С++ определены четыре *спецификатора типов данных*:

- short короткий ;
- long длинный ;

- signed знаковый ;
- unsigned беззнаковый .

Далее (п.п. 2.2.2. - 2.2.6) будут рассмотрены назначение и описание каждого типа[1].

2.2.2. Символьный тип

Данные типа char в памяти компьютера всегда занимают один байт. Это связано с тем, что обычно под величину символьного типа отводят столько памяти, сколько необходимо для хранения любого из 256 символов клавиатуры. Символьный тип может быть со знаком или без знака. В величинах со знаком signed char можно хранит значения в диапазоне от 128до 127. Соответственно значения переменных типа unsigned char могут находиться в диапазоне от 0 до 255 (табл. 2.1).

Таблица. 2.1. Символьные типы данных

Тип	Диапазон	Размер
char	-128..127	1 байт
unsigned char	0..255	1 байт
signed char	-128..127	1 байт

При работе с символьными данными нужно помнить, что если в выражении встречается *одиночный символ*, он должен быть заключен в одинарные кавычки. Последовательность символов, то есть *строка*, при использовании в выражениях заключается в двойные кавычки.

2.2.3. Целочисленный тип

Переменная типа int в памяти компьютера может занимать либо два, либо четыре байта. Это зависит от разрядности процессора. По умолчанию все целые типы считаются знаковыми, т.е. спецификатор signed можно не указывать. Спецификатор unsigned позволяет представлять только положительные числа.

Диапазоны значений целого типа представлены в таблице 2.2.

Таблица. 2.2. Целые типы данных

Тип	Диапазон	Размер
int	-32767 .. 32767	4 байта
unsigned int	0 .. 65535	4 байта
signed int	-32767 .. 32767	4 байта
short int	-32767 .. 32767	2 байта
long int	-2147483647 .. 2147483647	4 байта
unsigned short int	0 .. 65535	2 байта
signed short int	-32767 .. 32767	2 байта
long long int	$-(2^{63}-1)$ .. $(2^{63}-1)$	8 байт
signed long int	-2147483647 .. 2147483647	4 байта
unsigned long int	0 .. 4294967295	4 байта
unsigned long long int	0 .. $2^{64}-1$	8 байт

2.2.4. Вещественный тип

Внутреннее представление вещественного числа в памяти компьютера отличается от представления целого числа. Число с плавающей точкой представлено в экспоненциальной форме **mE±p**, где **m** мантисса (целое или дробное число с десятичной точкой), **p** порядок (целое число). Для того чтобы перевести число в экспоненциальной форме к обычному представлению с фиксированной точкой, необходимо **мантиссу** умножить на **десять** в степени **порядок**. Например,

-6.42E+2 = $-6.42.10^{2}$ = -642,

3.2E-6 = $3.2.10^{-6}$ =0.0000032

1 В таблицах этой главы приведены стандартные размеров памяти, соответствующие типам данных. Узнать объем занимаемой переменной для конкретного компилятора можно с помощью функции определения размера sizeof (п.1.5.7).

Обычно величины типа float занимают 4 байта, из которых один двоичный разряд отводится под знак, 8 разрядов под порядок и 23 под мантиссу. Поскольку старшая цифра мантиссы всегда равна 1, она не хранится.
Величины типа double занимают 8 байт, в них под порядок и мантиссу отводится 11 и 52 разряда соответственно. Длина мантиссы определяет *точность числа*, а длина порядка его *диапазон*. Спецификатор типа long перед именем типа double указывает, что под величину отводится 10 байт.
Диапазоны значений вещественного типа представлены в таблице 2.3.

Таблица. 2.3. Вещественные типы данных

Тип	**Диапазон**	**Размер**
float	3.4E-38 .. 3.4E+38	4 байта
double	1.7E-308 .. 1.7E+308	8 байт
long double	3.4E-4932 .. 3.4E+4932	8 байт

2.2.5. Логический тип

Переменная типа bool может принимать только два значения true (истина) или false (ложь). Любое значение не равное нулю интерпретируется как true, а при преобразовании к целому типу принимает значение равное 1. Значение false представлено в памяти как 0.

2.2.6. Тип void

Множество значений этого типа пусто. Он используется для определения функций, которые не возвращают значения, для указания пустого списка аргументов функции, как базовый тип для указателей и в операции приведения типов.

2.3. Переменные в языке C++

Переменная поименованный участок памяти ,в котором хранится значение определенного типа. У переменной есть *имя* (идентификатор) и *значение*. Имя служит для обращения к области памяти, в которой хранится значение. Во время выполнения программы значение переменной можно изменить. Перед использованием любая переменная должна быть *описана*:

тип список_переменных;

Например,

```
int a, bc, f;
float g, u, h12;
```

В Си могут обрабатываться *структурированные типы данных*: массивы, строки, структуры, файлы.
По *месту объявления* переменные в языке Си можно разделить на три класса: локальные, глобальные и формальные параметры функции.
Локальные переменные объявляются внутри функции и доступны только в ней. Например:

```
int main()
{
float s; //В функции main определена вещественная переменная s,
s=4.5;   //и ей присвоено значение 4.5.
}
int f1()
{
int s; //В функции f1 описана другая переменная s (типа int),
s=6;   //ей присвоено значение 6.
}
int f2()
{
long int s; //В функции f2 определена еще одна переменная s
s=25;      //(типа long int) и ей присвоено значение 25.
}
```

Глобальные переменные описываются до всех функций и доступны из любого места программы. Например:

```
float s; //Определена глобальная переменная s (типа float).
int main()
{
s=4.5; //В функции main переменной s присваивается значение 4.5.
}
int f1()
{
s=6; //В функции f1 переменной s присваивается значение 6.
}
int f2()
{
s=25; //В функции f2 переменной s присваивается значение 25.
}
```

Формальные параметры функций описываются в списке параметров функции. Работа с функциями подробно описана в главе 4.

2.4. Константы в языке C++

Константы это величины, которые не изменяют своего значения в процессе выполнения программы. Оператор описания константы имеет вид:

const тип имя=значение;

Константы в языке C++ могут быть целыми, вещественными, символьными или строковыми. Обычно компилятор определяет тип константы по внешнему виду, но существует возможность и явного указания типа, например,

```
const double pi=3.141592653589793
```

Константа может быть определена до главной функции main. В этом случае применяется директива #define:

```
#define PI 3.141592653589793
int main()
{…
```

2.5. Операции и выражения

Выражение задает порядок выполнения действий над данными и состоит из операндов (констант, переменных, обращений к функциям), круглых скобок и знаков операций: a +b*sin(cos(x)). *Операции* делятся на унарные (например, -c) и бинарные (например, a+b). В табл. 2.4 представлены основные операции языка C++.

Таблица. 2.4. Основные операции языка C++

Операция	**Описание**
Унарные операции	
++	увеличение значения на единицу
--	уменьшение значения на единицу
~	поразрядное отрицание
!	логическое отрицание
-	арифметическое отрицание (унарный минус)
+	унарный плюс
&	взятие адресаа
*	разадресация
(type)	преобразование типа
Бинарные операции	
+	сложение
-	вычитание
*	умножение

Операция	Описание
/	деление
%	остаток от деления
<<	сдвиг влево
>>	сдвиг вправо
<	меньше
>	больше
<=	меньше или равно
>=	больше или равно
==	равно
!=	не равно
&	поразрядная конъюнкция (И)
^	поразрядное исключающее ИЛИ
|	поразрядная дизъюнкция (ИЛИ)
&&	логическое И
||	логическое ИЛИ
=	присваивание
*=	умножение с присваиванием
/=	деление с присваиванием
+=	сложение с присваиванием
-=	вычитание с присваиванием
%=	остаток от деления с присваиванием
<<=	сдвиг влево с присваиванием
>>=	сдвиг вправо с присваиванием
&=	поразрядная конъюнкция с присваиванием
|=	поразрядная дизъюнкция с присваиванием
^=	поразрядное исключающее ИЛИ с присваиванием
Другие операции	
?:	условная операция
,	последовательное вычисление
sizeof	определение размера
(тип)	преобразование типа

Перейдем к подробному рассмотрению основных операций языка.

2.5.1. Операции присваивания

Обычная *операция присваивания* имеет вид:

имя_переменной=значение;

где значение это выражение, переменная, константа или функция. Выполняется операция так. Сначала вычисляется значение выражения указанного в правой части оператора, а затем его результат записывается в область памяти, имя которой указано слева. Например, запись a=b означает, что переменной a присваивается значение b. Если a и b переменные разных типов, происходит преобразование типов: значение в правой части преобразуется к типу переменной левой части. Следует учитывать, что при этом можно потерять информацию или получить другое значение.

В C++ существует возможность присваивания нескольким переменным одного и того же значения. Такая операция называется *множественным присваиванием* и в общем виде может быть записана так:

имя_переменной1= имя_переменной2=..= имя_переменнойN=значение;

Запись a=b=c=3.14159/6; означает, что переменным a, b и c было присвоено одно и то же значение 3.14159/6.

Операции +=, =, *=, /= называют *составным присваиванием.* В таких операциях при

вычислении выражения стоящего справа используется значение переменной из левой части, например так:

```
x+=p; //Увеличение x на p, то же что и x=x+p.
x-=p; //Уменьшения x на p, то же что и x=x-p.
x*=p; //Умножение x на p, то же что и x=x*p.
x/=p; //Деление x на p, то же что и x=x/p.
```

2.5.2. Арифметические операции

Операции +, -, *, / относят к *арифметическим операциям.* Их назначение понятно и не требует дополнительных пояснений.

Операции инкремента ++ и *декремента* -- так же причисляют к арифметическим, так как они выполняют увеличение и уменьшение на единицу значения переменной. Эти операции имеют две формы записи *префиксную* (операция записывается перед операндом) и *постфиксную* (операция записывается после операнда). Так, например оператор p=p+1; можно представить в префиксной форме ++p; и в постфиксной p++;. Эти формы отличаются при использовании их в выражении. Если знак декремента (инкремента) предшествует операнду, то сначала выполняется увеличение (уменьшение) значения операнда, а затем операнд участвует в выражении. Например,

```
x=12;
y=++x; //В переменной y будет храниться значение 13.
```

Если знак декремента (инкремента) следует после операнда, то сначала операнд участвует в выражении, а затем выполняется увеличение (уменьшение) значения операнда:

```
x=12;
y=x++; //Результат – число 12 в переменной y.
```

Остановимся на *операциях целочисленной арифметики*. Операция *целочисленного деления* / возвращает целую часть частного (дробная часть отбрасывается) в том случае если она применяется к целочисленным операндам, в противном случае выполняется обычное деление: 11/4=2 или 11.0/4=2.75. Операция *остаток от деления* % применяется только к целочисленным операндам: 11%4 = 3.

К *операциям битовой арифметики* относятся следующие операции: &, |, ^, ~, <<, >>. В операциях битовой арифметики действия происходят над двоичным представлением целых чисел.

Арифметическое И (&). Оба операнда переводятся в двоичную систему, затем над ними происходит логическое поразрядное умножение операндов по следующим правилам:

1&1=1,
1&0=0,
0&1=0,
0&0=0.

Например, если A=13 и B=23, то их двоичное представление соответственно A=0000000000001101 и B=0000000000010111. В результате логического умножения A&B получим 0000000000000101 или 5 в десятичной системе счисления. Таким образом, A&B=13&23=5.

```
& 0000000000001101
  0000000000010111
  0000000000000101
```

Логическое умножение A&B

Арифметическое ИЛИ (|). Здесь также оба операнда переводятся в двоичную систему, после чего над ними происходит логическое поразрядное сложение операндов по следующим правилам:

1|1=1,
1|0=1,
0|1=1,

0|0=0.
Например, результат логического сложения чисел A=13 и B=23 будет равен A|B=31 (рис. 2.2).

```
| 0000000000001101
  0000000000010111
  0000000000011111
```

Логическое сложение A|B

Арифметическое исключающее ИЛИ (^). Оба операнда переводятся в двоичную систему, после чего над ними происходит логическая поразрядная операция ^ по следующим правилам:
1^1=0,
1^0=1,
0^1=1,
0^0=0.
Арифметическое отрицание (~). Эта операция выполняется над одним операндом. Применение операции ~ вызывает побитную инверсию двоичного представления числа ~13=14.

```
    0000000000001101
~a  1111111111110010
```

Арифметическое отрицание

Сдвиг влево (M<<L). Двоичное представление числа M сдвигается влево на L позиций. Рассмотрим операцию 17<<3. Число 17 в двоичной системе имеет вид 10001. При сдвиге его на 3 позиции влево получим 10001000. В десятичной системе это двоичное число равно 136. Итак, 17<<3 =136. Заметим, что сдвиг на один разряд влево соответствует умножению на два, на два разряда умножению на четыре , на три умножению на восемь .Таким образом , операция M<<L эквивалентна умножению числа M на 2 в степени L.

Сдвиг вправо (M>>L). В этом случае двоичное представление числа M сдвигается вправо на L позиций, что эквивалентно целочисленному делению числа M на в степени L. Например, 25>>1=12, 25>>3= 3.

2.5.3. Логические операции

В C++ определены следующие *логические операции* ||, &&, !. Логические операции выполняются над логическими значениями true (истина) и false (ложь). В языке C++ ложь 0, истина любое значение ≠ 0. В табл. 2.5 приведены результаты логических операций.

Таблица. 2.5. Логические операции

A	B	!A	A&&B	A||B
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

2.5.4. Операции отношения

Операции отношения возвращают в качестве результата логическое значение. Таких операций шесть: >, >=, <, <=, ==, !=. Результат операции отношения логическое значение true (истина) или false (ложь).

2.5.5. Условная операция

Для организации разветвлений в простейшем случае можно использовать *условную операцию* ? :. Эта операция имеет три операнда и в общем виде может быть представлена так:

условие ? выражение1 : выражение2;

Работает операция следующим образом. Если условие истинно (не равно 0), то результатом будет выражение1, в противном случае выражение2. Например, операция y=x<0 ? x : x; записывает в переменную y модуль числа x.

2.5.6. Операция преобразования типа

Для приведения выражения к другому типу данных в C++ существует *операция преобразования типа*:

(тип) выражение;

Например, в результате действий x=5; y=x/2; z=(float) x/2; переменная y примет значение равное 2 (результат целочисленного деления), а переменная z = 2.5.

2.5.7. Операция определения размера

Вычислить размер объекта или типа в байтах можно с помощью *операции определения размера*, которая имеет две формы записи:

sizeof (тип) или **sizeof выражение**

В качестве примера рассмотрим программу, вычисляющую размеры основных типов данных в байтах. Результаты вычислений выводятся на экран (рис. 2.1).

```
#include "stdafx.h"
#include "iostream"
using namespace std;
int main()
{
int i=3;              //Определение целочисленной переменной.
double d=0.2;  //Определение вещественной переменной.
//Вычисление размеров различных типов данных:
cout<<"Size char:"<<sizeof (char)<<"\n";
cout<<"Size int:"<<sizeof (int)<<"\n";
cout<<"Size short int:"<<sizeof (short int)<<"\n";
cout<<"Size long int:"<<sizeof (long int)<<"\n";
cout<<"Size long long int:"<<sizeof (long long int)<<"\n";
cout<<"Size float:"<<sizeof (float)<<"\n";
cout<<"Size double:"<<sizeof (double)<<"\n";
cout<<"Size long double:"<<sizeof (long double)<<"\n";
//Размер памяти, отведенной под строку "STROKA".
cout<<"Size STROKA:"<<sizeof "STROKA"<<"\n";
//Размер памяти, отведенной под целочисленную переменнцю i.
cout<<"Size i:"<<sizeof i<<"\n";
//Размер памяти, отведенной под значение выражения i+d.
cout<<"Size i+d:"<<sizeof (i+d)<<"\n";
return 0;
}
```

На рис. 2.1. видно, сколько памяти отводится под тот или иной тип данных. Остановимся на определении размера объектов. Так, объект STROKA состоит из 6 символов и занимает 7 байтов, т.е. по одному байту на каждую букву и один байт на символ окончания строки (п. 2.2.2). Под переменную i отведено 4 байта. Действительно, тип переменной int, занимает в памяти 4 байта. Последний объект это выражение i+d. Его значение занимает в памяти 8 байт. Это связано с тем, что тип double более длинный (8 байтов) по сравнению с типом int (4 байта) и значение результата было преобразовано к более длинному типу[2].

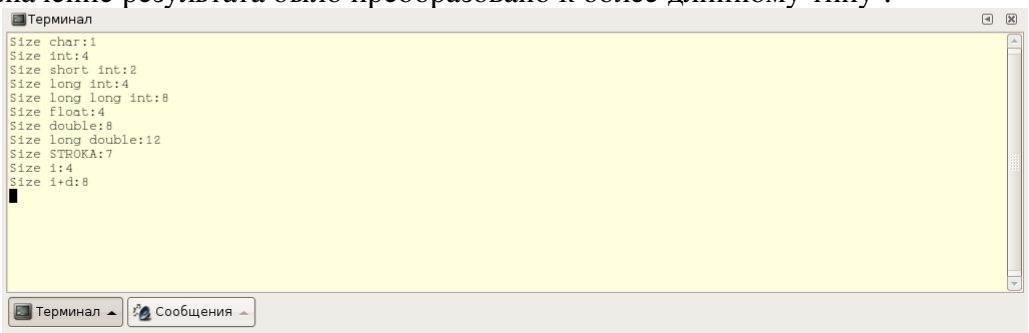

Рис. 2.1. Результаты работы программы, определяющей размеры объектов

2 В записи операции sizeof (i+d) были использованы скобки. Это связано с тем, что операция приведения типа имеет более высокий приоритет, чем операция сложения.

2.6. Стандартные функции

В C++ определены *стандартные функции* над арифметическими операндами (табл. 2.6).

Таблица. 2.6. Стандартные математические функции

Обозначение	**Действие**
abs(x)	Модуль целого числа
fabs(x)	Модуль вещественного числа
sin(x)	Функция синус
cos(x)	Функция косинус
tan(x)	Функция тангенс
atan(x)	Арктангенс x в диапазоне $-\pi/2$ до $\pi/2$
exp(x)	Экспонента, e^x
log(x)	Натуральный логарифм (x>0)
log10(x)	Десятичный логарифм (x>0)
sqrt(x)	Корень квадратный (x≥0)
pow(x,y)	x в степени y

Определенную проблему представляет применение функции pow(x,y), которая возводит x в степень y. При программировании выражений, содержащих возведение в степень, надо внимательно проанализировать значения, которые могут принимать x и y, так как в некоторых случаях возведение x в степень y невыполнимо.

Так, ошибка возникает, если x отрицательное число, а y дробь. Предположим, что y правильная дробь вида *k/m*. Если знаменатель *m* четный, это означает вычисление корня четной степени из отрицательного числа, а значит, операция не может быть выполнена. В противном случае, если знаменатель *m* нечетный, можно воспользоваться выражением z= pow(fabs(x),y).

2.7. Структура программы

Программа на языке C++ состоит из *функций*, *описаний* и *директив процессора*.

Одна из функций должна обязательно носить имя main. Элементарное *описание функции* имеет вид:

тип_результата имя_функции (параметры)
{
оператор1;
оператор2;
..
операторN;
}

Здесь, тип_результата это тип того значения, которое функция должна вычислить (если функция не должна возвращать значение, указывается тип void), имя_функции имя, с которым можно обращаться к этой функции, параметры список аргументов функции (может отсутствовать), оператор1, оператор2,.., операторN операторы, представляющие тело функции, они обязательно заключаются в фигурные скобки и каждый оператор заканчивается точкой с запятой[3]. Как правило программа на C++ состоит из одной или нескольких, не вложенных друг в друга функций.

Основному тексту программы предшествуют *директивы процессора*, которые в общем виде выглядят так:

#include <имя_файла>

Каждая такая строка дает компилятору команду присоединить программный код, который хранится в отдельном файле с расширением .h. Такие файлы называют файлами заголовков.

3 Подробно работу с функциями рассмотрим в четвертой лекции.

С их помощью можно выполнять ввод-вывод данных, работать с математическими функциями, преобразовывать данные, распределять память и многое другое. Например, описание стандартных математических функций находится в заголовочном файле math.h.
Общую структуру программы на языке С++ можно записать следующим образом:
директивы процессора
описание глобальных переменных
тип_результата main(параметры)
{
операторы главной функции
}
тип_результата имя1(параметры1)
{
операторы1;
}
тип_результата имя2(параметры2)
{
операторы2;
}
.....
тип_результата имяN(параметрыN)
{
операторыN;
}

2.8. Ввод и вывод данных

Ввод-вывод данных в языке С++ осуществляется либо с помощью *функций ввода-вывода* в стиле С, либо с использованием *библиотеки классов* С++. Преимущество объектов С++ в том, что они легче в использовании, особенно если ввод-вывод достаточно простой. Функции ввода-вывода унаследованные от С более громоздкие, но подходят для задач с форматированным выводом данных.

2.8.1. Функции ввода- вывода

Функция

printf(строка форматов, список выводимых переменных)

выполняет форматированный *вывод* переменных, указанных в списке, в соответствии со строкой форматов.
Функция

scanf(строка форматов, список адресов вводимых переменных)

выполняет *ввод* переменных, адреса которых указанны в списке, в соответствии со строкой форматов.
Строка форматов содержит символы, которые будут выводиться на экран или запрашиваться с клавиатуры и так называемые спецификации. *Спецификации* это строки, которые начинаются символом % и выполняют управление форматированием:

% флаг ширина . точность модификатор тип

Параметры флаг, ширина, точность и модификатор в спецификациях могут отсутствовать. Значения параметров спецификаций приведены в табл. 2.7.

Таблица 2.7. Символы управления

Параметр	Назначение
Флаги	
-	Выравнивание числа влево. Правая сторона дополняется пробелами. По умолчанию выравнивание вправо.

Параметр	Назначение
+	Перед числом выводится знак «+» или «-»
Пробел	Перед положительным числом выводится пробел, перед отрицательным «-»
#	Выводится код системы счисления: 0 - перед восьмеричным числом, 0x (0X) перед шестнадцатеричным числом.
Ширина	
n	Ширина поля вывода. Если n позиций недостаточно, то поле вывода расширяется до минимально необходимого. Незаполненные позиции заполняются пробелами.
0n	То же, что и n, но незаполненные позиции заполняются нулями.
Точность	
ничего	Точность по умолчанию
n	Для типов e, E, f выводить n знаков после десятичной точки
Модификатор	
h	Для d, i, o, u, x, X тип short int.
l	Для d, i, o, u, x, X тип long int.
Тип	
c	При вводе символьный тип char, при выводе один байт.
d	Десятичное int со знаком.
i	Десятичное int со знаком.
o	Восьмеричное int unsigned.
u	Десятичное int unsigned.
x, X	Шестнадцатеричное int unsigned, при x используются символы a-f, при X - A-F.
f	Значение со знаком вида [-]dddd.dddd.
e	Значение со знаком вида [-]d.ddde[+|-]ddd.
E	Значение со знаком вида [-]d.ddddE[+|-]ddd.
g	Значение со знаком типа e или f в зависимости от значения и точности.
G	Значение со знаком типа e или F в зависимости от значения и точности.
s	Строка символов.

Кроме того, строка форматов может содержать некоторые специальные символы, которые приведены в табл. 2.8.

Таблица 2.8. Специальные символы

Символ	Назначение
\b	Сдвиг текущей позиции влево.
\n	Перевод строки.
\r	Перевод в начало строки, не переходя на новую строку.
\t	Горизонтальная табуляция.
\'	Символ одинарной кавычки.
\"	Символ двойной кавычки.
\?	Символ ?.

Первой строкой программы, в которой будут применяться функции ввода-вывода языка C, должна быть директива #include <stdio.h>. Заголовочный файл stdio.h содержит описание

функций ввода-вывода.
Рассмотрим работу функций на примере следующей задачи.
ЗАДАЧА 2.1. Зная a, b, c длины сторон треугольника ,вычислить площадь Sи периметр P этого треугольника.
Входные данные*:* a, b, c. Выходные данные*:* S, P.
Для вычисления площади применим формулу Герона: $S=\sqrt{r(r-a)(r-b)(r-c)}$, где r - полупериметр.
Далее приведены две программы для решения данной задачи и результаты их работы. Сравните работу функций printf и scanf в этих программах.

```
//ЗАДАЧА 2.1. Вариант первый

#include <stdio.h>
#include <math.h>
int main()
{
float a,b,c,S,r;      //Описание переменных.
//В операторах scanf для вычисления адреса
//переменной применяется операция &.
printf("a="); //Вывод на экран символов a=.
scanf("%f",&a);            //Запись в переменную a
                           //значениия введенного с клавиатуры.
printf("b=");              //Вывод на экран символов b=.
scanf("%f",&b);            //Запись в переменную b значениия
                           //введенного с клавиатуры.
printf("c=");              //Вывод на экран символов c=
scanf("%f",&c);            //Запись в переменную c значениия
                           //введенного с клавиатуры.
r=(a+b+c)/2;               //Вычисление полупериметра.
S=sqrt(r*(r-a)*(r-b)*(r-c));     //Вычисление площади треугольника.
//Вывод символов S=, значения S и символа табуляции \t.
//Спецификация %5.2f означает, что будет выведено вещественное число,
//под которое отводится 5 знаков, причем 2 из них после точки.
printf("S=%5.2f \t",S);
//Вывод символов p=, значения выражения 2*r
//и символа окончания строки \n.
printf("p=%5.2f \n",2*r);
//Оператор printf("S=%5.2f \t p=%5.2f \n",S,2*r);
//выдаст тот же результат.
return 0;}


//ЗАДАЧА 2.1. Вариант второй

#include <stdio.h>
#include <math.h>

int main()
{
float a,b,c,S,r;
printf("Vvedite a, b, c \n");    //Вывод на экран строки символов.
scanf("%f%f%f",&a,&b,&c);        //Ввод значений.
r=(a+b+c)/2;
S=sqrt(r*(r-a)*(r-b)*(r-c));
printf("S=%5.2f \t p=%5.2f \n",S,2*r);     //Вывод на экран результатов.
return 0;
}
```

2.8.2. Объектно-ориентированные средства ввода-вывода.

Описание объектов для управления *вводом-выводом* содержится в заголовочном файле iostream.h. При подключении этого файла с помощью директивы #include <iostream.h> в программе автоматически создаются объекты-потоки[4] cin для *ввода* с клавиатуры и cout для *вывода* на экран, а так же операции помещения в поток << и чтения из потока >>.

Итак, с помощью объекта cin и операции >> можно присвоить значение любой переменной. Например, если переменная i описана как целочисленная, то команда cin>> i; означает, что в переменную i будет записано некое целое число, введенное с клавиатуры. Если нужно ввести несколько переменных, следует написать cin>>x>>y>>z;.

Объект cout и операция << позволяют вывести на экран значение любой переменной или текст. Текст необходимо заключать в двойные кавычки, кроме того, допустимо применение специальных символов \t и \n. Запись cout<<i; означает вывод на экран значения переменной i. А команда cout<<x<<"\t"<<y; выведет на экран значения переменных x и y разделенные символом табуляции.

Рассмотрим следующую задачу.

ЗАДАЧА 2.2. Известны плотность ρ, высота h и радиус основания R цилиндрического слитка, полученного в металлургической лаборатории. Найти объем V, массу m и площадь S основания слитка.

Входные данные*:* ρ, h, R. Выходные данные*:* S, V, m.

Учитывая, что $S=2\pi R$, $V=\pi R^2 h$ и $m=\rho V$ составим текст программы:

```
#include "stdafx.h"
#include <iostream>
using namespace std;
#define pi 3.14159          //Определение константы.
int main()
{double R,h,r,S,V,m;  //Описание переменных.
cout<<"R=";           //Вывод на экран символов R=.
cin>>R;               //Ввод значения переменной R.
cout<<"h=";           //Вывод на экран символов h=.
cin>>h;               //Ввод значения переменной h.
cout<<"r=";           //Вывод на экран символов r=.
cin>>r;               //Ввод значения переменной r.
S=2*pi*R;             //Расчет площади.
V=pi*R*R*h;           //Вычисление объема.
m=r*V;                //Определение массы.
cout<<"S="<<S;  //Вывод на экран символов S= и значения переменной S.
cout<<"\n V="<<V;     //Вывод на экран с новой строки V= и значения V.
cout<<"\n m="<<m;     //Вывод на экран с новой строки m= и значения m.
return 0;
}
```

4 Поток - виртуальный канал связи, создаваемый в программе для передачи данных.