Управляющие последовательности

\a	**7**	**Звуковой сигнал**
\b	8	Возврат на шаг
\f	C	Перевод страницы (формата)
\n	**A**	**Перевод строки**
\r	D	Возврат каретки
\t	**9**	**Горизонтальная табуляция**
\v	**B**	**Вертикальная табуляция**
\\	5C	Обратная косая черта
\'	27	Апостроф
\"	22	Кавычка
\?	3F	Вопросительный знак
\0ddd		Восьмеричный код символа
\0xdd	dd	Шестнадцатиричный код символа

Диапазоны для IBM PC-совместимых

Тип	Диапазон значений	Размер(байт)
bool	true и false	1
signed char	-128 … 127	1
unsigned char	0 … 255	1
signed short int	-32 768 … 32 767	2
unsigned short *int*	0 … 65 535	2
signed long int	-2 147 483 648 … 2 147 483 647	4
unsigned long int	0 … 4 294 967 295	4
float	3.4e-38 … 3.4e+38	4
double	1.7e-308 … 1.7e+308	8
long double	3.4e-4932 … 3.4e+4932	10

Стандартные математические функции

Математическая функция	Имя функции в языке С	Математическая функция	Имя функции в языке С
корень(x)	**sqrt(x)**	**arcsin(x)**	**asin(x)**
|x|	**fabs(x)**	**arccos(x)**	**acos(x)**
e^x	**exp(x)**	**arctg(x)**	**atan(x)**
x^y	**pow(x,y)**	**arctg(x/y)**	**atan2(x,y)**
ln(x)	**log(x)**	**sh(x)=1/2 (e^x-e^{-x})**	**sinh(x)**
lg_{10}(x)	**log10(x)**	**ch(x)=1/2 (e^x+e^{-x})**	**cosh(x)**
sin(x)	**sin(x)**	**tgh(x)**	**tanh(x)**
cos(x)	**cos(x)**	**Ост. от деления x на y**	**fmod(x,y)**
tg(x)	**tan(x)**	**Наим. целое, которое >=x**	**ceil(x)**
		Наиб. целое, которое <=x	**floor(x)**

Файлы библиотечных функций (директивы препроцессора)

#include <stdio.h> - подключение файла с объявлением стандартных функций файлового ввода-вывода;

#include <conio.h> - функции работы с консолью;

#include <graphics.h> - графические функции;

#include <math.h> - математические функции.

#include<iostream.h> - подключение библиотеки потокового ввода-вывода

Функции вывода информации

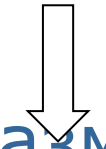
putchar() - обеспечивает вывод одиночного символа без перехода на новую строку.

puts() - используется для вывода строки символов с переходом на начало новой строки.

printf() - форматированный вывод данных.

Формат:

printf (<управляющая строка>, <спис. арг.>);



% <флаг><размер поля . точность>
спецификация

Форматы функции печати (спецификация)

Формат	Тип выводимой информации
d	десятичное целое число
c	один символ
s	строка символов
e	число с плавающей точкой (экспоненциальная запись) 1.2E+21
f	число с плавающей точкой (десятичная запись)
u	десятичное число без знака
o	восьмеричное число без знака
x	шестнадцатеричное число без знака

Функции ввода информации

getch () ввод одиночных символов.

gets () ввод строки символов до нажатия клавиши ENTER.

scanf форматированный ввод информации любого вида.

Формат:

scanf (<управляющая строка>, <список адресов>);

Пример 1 - простейшая программа

```
#include <stdio.h>
int main(){
   int i;
   printf("Введите целое число\n");
   scanf("%d", &i);
   printf("Вы ввели число %d, спасибо!", i);
}
```

```
#include <cstdio>
using namespace std;
int main(){
   int i;
   printf("Введите целое число\n");
   scanf("%d", &i);
   printf("Вы ввели число %d, спасибо!", i);
}
```

Пример 2 - целые форматы

```
#include <stdio.h>
int main(){
    int int1 = 45, int2 = 13;
    printf("int1 = %d| int2 = %3d| int2 = %-4d|\n",
        int1, int2, int2);
    printf("int1 = %X| int2 = %3x| int2 = %4o|\n",
        int1, int2, int2);
}
```

```
int1 = 45| int2 =  13| int2 = 13  |
int1 = 2D| int2 =   d| int2 =   15|
```

Пример 3 - вещественные форматы

```
#include <stdio.h>
int main(){
float f = 3.621;
double dbl = 2.23;
printf("f = %f| f = %4.2f| f = %6.1f|\n", f, f, f);
printf("f = %g| f = %e| f = %+E|\n", f, f, f);
printf("dbl = %5.2lf| dbl = %e| dbl = %4.1G|\n",
      dbl, dbl, dbl);
}
```

```
f = 3.621000| f = 3.62| f =    3.6|
f = 3.621| f = 3.621000e+000| f = +3.621000E+000|
dbl =  2.23| dbl = 2.230000e+000| dbl =    2|
```

Пример 4 - форматы символов и строк

```
#include <stdio.h>
int main(){
char ch = 'z', *str = "ramambahari";
printf("ch = %c| ch = %3c|\n", ch, ch);
printf("str = %14s|\nstr = %-14s|\nstr = %s|\n",
      str, str, str);
}
```

```
ch = z| ch =   z|
str =    ramambahari|
str = ramambahari   |
str = ramambahari|
```

Приоритеты операций

Операция	Краткое описание
Унарные операции	
::	доступ к области видимости
.	выбор
->	выбор
[ ]	индексация
( )	вызов функции
<тип>( )	конструирование
++	постфиксный инкремент
– –	постфиксный декремент
typeid	идентификация типа
dynamic_cast	преобразование типа с проверкой на этапе выполнения
static_cast	преобразование типа с проверкой на этапе компиляции
reinterpret_cast	преобразование типа без проверки
const_cast	константное преобразование типа

Приоритеты операций

sizeof	размер объекта или типа
– –	префиксный декремент
++	префиксный инкремент
~	поразрядное отрицание
!	логическое отрицание
–	арифметическое отрицание (унарный минус)
+	унарный плюс
&	взятие адреса
*	разадресация
new	выделение памяти
delete	освобожение памяти
(<тип>)	преобразование типа

Приоритеты операций

.*	выбор
->*	выбор
Бинарные и тернарная операции	
*	умножение
/	деление
%	остаток от деления
+	сложение
–	вычитание
<<	сдвиг влево
>>	сдвиг вправо

<	меньше
<=	меньше или равно
>	больше
>=	больше или равно
==	равно
!=	не равно
&	поразрядная конъюнкция (И)
^	поразрядное исключающее ИЛИ
|	поразрядная дизъюнкция (ИЛИ)
&&	логическое И
||	логическое ИЛИ
? :	условная операция (тернарная)
=	присваивание
*=	умножение с присваиванием
/=	деление с присваиванием

%=	остаток отделения с присваиванием
+=	сложение с присваиванием
–=	вычитание с присваиванием
<<=	сдвиг влево с присваиванием
>>=	сдвиг вправо с присваиванием
&=	поразрядное И с присваиванием
|=	поразрядное ИЛИ с присваиванием
^=	поразрядное исключающее ИЛИ с присваиванием
throw	исключение
,	последовательное вычисление

Операции выполняются в соответствии с *приоритетами*. Для изменения порядка выполнения операций используются круглые скобки. Если в одном выражении записано несколько операций одинакового приоритета, унарные операции, условная операция и операции присваивания выполняются *справа налево*, остальные — *слева направо.*