

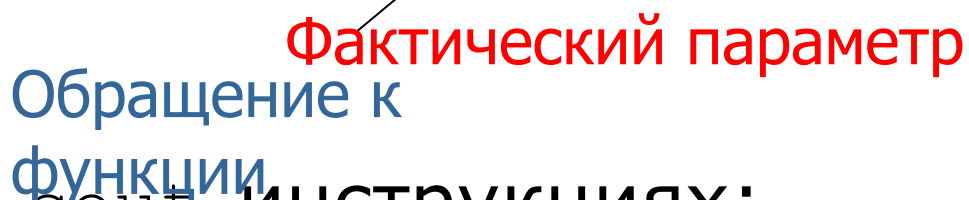
Библиотечные функции

Синтаксис использования функции в программе:

```
the_root = sqrt(9.0);
```

Обращение к
функции

Фактический параметр



Вызов функции в `cout`-инструкциях:

```
cout<<"Длина стороны квадрата, площадь  
которого"<<area<<" , равна"<< (sqrt(area)) ;
```

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

Комментарии

- Однострочные

//.....

- Многострочные

/*
...
*/

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

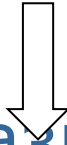
putchar() - обеспечивает вывод одиночного символа без перехода на новую строку.

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

Примеры форматированного вывода

```
int num=5, cost=11000, s=-777;  
float bat=255, x=12.345;  
printf ("на %d студентов %f бутербродов\n", num,  
bat);  
printf ("Значение числа pi равно%f.\n", PI);  
printf ("Любовь и голод правят миром.\n");  
printf ("Стоимость этой вещи %d%s.\n", cost,"  
Руб.");  
printf ("x=%-8.4f s=%5d%8.2f ", x, s, x);  
x=12.3450    s= -777    12.34
```

- Выравнивание по левому краю

8 позиций на целую часть 4 позиции на дробную

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>);

Примеры форматированного ввода

```
int course; // название переменных
float grant;
char name[20]; // строка символов
printf ( "Укажите ваш курс, стипендию,
        ИМЯ"); //может просто быть написана строка символов в
кавычках
scanf ( "%d%f", &course, &grant);
scanf ( "%s", name); //адрес у строк не
пишется (без амперсанда)
```

Первая программа

```
#include <stdio.h>
```

```
void main()
```

```
{  
    printf ("Hello, world!\n");  
}
```

- Включение информации о стандартной библиотеке.
- Определение функции с именем **main**, не получающей никаких аргументов.
- Инструкции **main** заключаются в фигурные скобки.
- Функция **main** вызывает библиотечную функцию **printf** для печати заданной последовательности символов
- **\n** — символ новой строки

Первая программа

```
#include <stdio.h>
int main()
{
    printf("Hello, world!");
    return 0;
}
```

```
#include <stdio.h>
void main()
{
    printf("Hello, world!");
}
```

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|
```

Пример 5 - классы ввода-вывода

```
#include <iostream.h>
int main(){
    int i;
    cout << "Введите целое число\n";
    cin >> i;
    cout << "Вы ввели число" << i << ", спасибо!";
}
```

```
#include <iostream>
using namespace std;
int main(){
    int i;
    cout << "Введите целое число\n";
    cin >> i;
    cout << "Вы ввели число" << i << ", спасибо!";
}
```


Операции C++ (не все!)

Унарные операции

++ -- sizeof ~ ! - + & * new delete
(type)

Бинарные операции

* / % + - << >> < <=
> >= == != & ^ | && || = *= /= %= += -= <<=
>>= &= |= ^= throw ,

Тернарная операция

? :

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
*	адресация
new	выделение памяти
delete	освобождение 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

<code>%=</code>	остаток от деления с присваиванием
<code>+=</code>	сложение с присваиванием
<code>-=</code>	вычитание с присваиванием
<code><<=</code>	сдвиг влево с присваиванием
<code>>>=</code>	сдвиг вправо с присваиванием
<code>&=</code>	поразрядное И с присваиванием
<code> =</code>	поразрядное ИЛИ с присваиванием
<code>^=</code>	поразрядное исключающее ИЛИ с присваиванием
<code>throw</code>	исключение
<code>,</code>	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*.