**Функции.** Функция - это набор операторов, объединённый одним названием. Функция может возвращать значение. Функции позволяют выделить блок кода и назначить ему имя, а затем этот блок многократно имени вызывать в различных частях программы по присвоенному.

Структура функции:  
[<класс памяти>][<тип>]<имя функции>([список параметров])  
{  
 <тело функции>  
}

Первая строка представляет заголовок функции. Вначале указывается возвращаемый тип функции. Если функция не возвращает никакого значения, то используется тип void. Затем идет имя функции, которое представляет произвольный идентификатор. К именованию функции применяются те же правила, что и к именованию переменных. После имени функции в скобках идет перечисление параметров. Функция может не иметь параметров, в этом случае указываются пустые скобки.

После заголовка функции в фигурных скобках идет тело функции, которое содержит выполняемые инструкции.

Для возвращения результата функция применяет оператор return. Если функция имеет в качестве возвращаемого типа любой тип, кроме void, то она должна обязательно с помощью оператора return возвращать какое-либо значение.

**Пример 1:**1 #include <iostream>

2 int square(int a, int b)  
3 {  
4 return a \* b;  
5 }

6 int main()  
7 {  
8 std::cout << square(4, 5);  
9 }

**Что здесь происходит?**

1. Мы используем директиву (указание) include, чтобы подключить библиотеку iostream. Эта библиотека входного и выходного потока. Попросту говоря, библиотека функций для ввода и вывода.
2. Мы объявляем функцию square. Это целочисленная функция, которая в качестве параметров принимает два целых числа.
3. Открываем тело функции.
4. Пишем нужные операторы. В данном случае - оператор return. Эта функция будет возвращать произведение двух чисел, которые она получила в качестве параметров.
5. Закрываем тело функции.
6. Объявляем функцию main (то есть, это место, откуда начнётся наша программа).
7. Открываем тело функции main.
8. Пишем нужные операторы. В данном случае мы выводим результат работы функции square.
9. Закрываем тело функции main (программа завершается).

**Как это работает?**

1. Компилятор ищет функцию main. Находит её на 6.
2. Проходим 7
3. Чтобы выполнить строку 8 нужно перейти к функции square
4. Идём на строку 2
5. Проходим 3
6. Выполняем умножение на строке 4
7. И по оператору return возвращаемся в main
8. Выводим то, что у нас вернулось через return

**Важно.** Пользовательские функции должны быть хотя бы объявлены (т.е., нужно написать тип функции, её имя и аргументы) до функции main. Если, к примеру, мы сначала вызовем, а потом объявим функцию, то мы получим ошибку на этапе компиляции.

**Возвращаемое значение**.

1. Функции с типом данных void ничего не возвращают в main
2. Для всех остальных функций (кроме main) нужно прописывать оператор возвращения return
3. Тип функции - тип того, что вернётся в main (или другую функцию - смотря откуда вызывали)

**Тело функции.** Это операторы, которые она выполняет. Тело функции находится в фигурных скобках.

**Важно.** Если тип функции - не void, то в теле функции обязательно должен быть указан оператор return.

**Вызов функции.** Когда запускается программа на языке C++, то обычно запускается функция main. Никакие другие функции, определенные в программе, автоматически не выполняются. Для выполнения функции ее необходимо вызвать. Чтобы вызвать функцию, нужно позвать её по имени и, если необходимо, указать параметры. square(4,5) в строке 8 - это вызов функции square, 4 и 5 - нужные фактические параметры.

**Формальные параметры**. Это то, что функции нужно узнать, чтобы она работала. В строке 2 int a и int b - формальные параметры.

**Фактические параметры**. Это то, что мы передаём функции при вызове. При выполнении операторов функции значения фактических параметров становятся на место формальных параметров. В строке 8 числа 4 и 5 - это фактические параметры.

**Перегрузка функций.** Это когда у двух и более функций одинаковые имена, но разное количество аргументов или их типы. У перегруженных функций могут быть разные типы.

Пример объявления перегруженных функций:

int func(int x, int y, int z){return 0;}

int func(char a, char b){return (int)a;}

char func(char a, char b, char c){return a;}

**Рекурсия.** Это когда функция вызывает саму себя.

Пример:

#include <iostream>

using namespace std;

int factorial(int n)

{

if (n> 1)

{

return n \* factorial(n - 1); //функция вызывает саму себя

}

else return 1;

}

int main()

{

int n = 0; cout<<"Enter n: ";

cin>> n;

cout<< factorial(n);

}

**Что здесь происходит?**

Допустим, n = 3. Когда мы вызываем функцию factorial(n), мы сразу проверяем условие n>1. Если условие истинно, то мы переходим по первой ветке. Там у нас есть оператор return n \* factorial(n-1). Чтобы его выполнить нам нужно зайти в функцию ещё раз. Теперь наше n = 2. Аналогично мы идём дальше, пока не дойдём до n = 0. Тогда функция начинает собираться и на выходе мы получаем 3\*(3-1)\*((3-1)-1) = 3\*2\*1 = 6, т.е. факториал 3.