УНИВЕРСИТЕТ СИНЕРГИЯ

РАЗРАБОТКА НА C++

Урок 1.
Функции в программировании

Функция

Функция - это набор операторов, объединенный одним названием. Программисты пишут функции для сокращения кода, а также для структурирования кода в своих проектах.

Например, вместо строчки:
int a = 5 * 5 * 5 * 5 * 5 * 5 * 5 * 5 * 5 * 5 * 5 * 5 * 5 * 5 * 5 * 5 * 5 * 5;

Можно написать:
int a = pow(5, 18);

Так программистам легче избежать ошибок, так как если функция протестирована, то при выявлении ошибки не нужно возвращаться к этому блоку кода

Структура функции

[<тип>] <имя функции> ([список параметров])

{

    <тело функции>

}

тип (int, double, string и т.д.)

имя (может быть любым, но должно начинаться с буквы либо символа “_”)

тело функции (исполняемый код)

Список параметров

Список параметров это любые значения или переменные, которые мы хотим использовать в функциях. Например функция которая складывает числа

```
int sum(int a, int b){
        return a+b;          //ключевое слово return “возвращает” значение
}
```

int func(){return 0;} - это тоже правильный вариант функции

Int func{return 0;} - а это неправильно, так как даже при отсутствии параметров нужно указать круглые скобки (так мы показываем компилятору, что это функция)

Возвращаемое значение

Часто функции производят различные расчеты (или обработку данных) и в зависимости от задумки программиста, после выполнения инструкции вместо функции будет располагаться значение

Например:

```
cout << sum(5, 5) << endl;
```

Во время выполнения этой строчки, функция будет заменена на:

```
cout << 10 << endl;
```

т.е. Функция sum “вернула” значение 10

Возвращаемое значение

Функция должна возвращать значение в зависимости от своего типа. Значение возвращается с помощью **оператора return**. На этом же операторе завершается выполнение функции.

Существует специальный тип данных, для которого можно не указывать оператор return. Это **тип void.** Функции типа void не возвращают значение.

Однако можно написать return; чтобы просто завершить функцию void.

Объявление и реализация функции

В языке C++ функции должны находиться выше в коде программы, чем место из которых их вызывают:

```
void func(){}


int main() {
	func(); //все окей! func выше main

	func2(); //main не видит эту функцию выше (это ошибка!)

}

void func2(){}
```

Объявление и реализация функции

Но если мы скажем компилятору что функция существует, то ошибки не будет.

Объявление функции может выглядеть так:
int func(int a, char b);

Мы сказали компилятору что функция func **существует** и **указали** какие **параметры** её нужны для выполнения

Реализацию, то есть тело функции, можно объявить в другом месте программы или даже в другом файле (позже научимся так делать)

Примеры функций

```
#include <iostream>

//объявление и реализация

int square(int a, int b) {

        return a * b;

}

int main() {

        std::cout << square(4, 5);

}
```

```
#include <iostream>

void hello(); // объявление функции

int main() {

        hello();

}

void hello() { // реализация функции

        std::cout << “Hello, world!”;

}
```

Вызов функции

Когда запускается программа на языке C++, то обычно запускается функция main. Значит для написания программы на C++ нужна **минимум одна функция**

Никакие другие функции, определенные в программе, автоматически не выполняются. Для выполнения функции ее необходимо вызвать. Всё выполняется последовательно!

```
int a = sum(sum(5, 5), 10);
        //сначала компилятор считает sum(5, 5) = 10
        //потом:
int a = sum(10, 10);
        //в конце:
int a = 20;
```

Чтобы вызвать функцию, нужно позвать её по имени и, если необходимо, указать фактические параметры.

Параметры

Формальные параметры. Это то, что функции нужно узнать, чтобы она работала. Т.е. это то, что мы указываем в заголовке функции.

```
int sum (int a, int b)

void print(string s)

double func(bool a, int b)
```

Фактические параметры. Это то, что мы передаем функции при вызове. При выполнении операторов функции значения фактических параметров становятся на место формальных параметров.

```
int number = sum(5, -18);

print(“Hello World!”);

double d = func(true, number);
```

Перегрузка функций

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

Рекурсия. Это когда функция вызывает саму себя.

```
void func()
{
    func(); //функция вызывает саму себя
}
```

При вызове такой функции мы уйдем в **бесконечную рекурсию**, так как функция будет снова и снова запускать себя.

Если используем рекурсию нужно продумать как мы из нее выйдем!

Рекурсия

```
int factorial(int n){
    if (n > 1){
        return n * factorial(n - 1);
    }
    else return 1;
}
```

```
int main(){
        cout << factorial(5) << endl;
}
```

```
// factorial(5)                    (5>1) true
// 5 * factorial(4)                (4>1) true
// 5 * 4 * factorial(3)            (3>1) true
// 5 * 4 * 3 * factorial(2)        (2>1) true
// 5 * 4 * 3 * 2 * factorial(1) (1>1) false
// 5 * 4 * 3 * 2 * 1    (return произойдет только когда все функции будут выполнены)
```