Глава 7 «Функции»

Функция в C++

Функция в программировании - это блок кода, который выполняет определенную задачу и может принимать входные параметры и возвращать значение. Функции позволяют организовать код в логические блоки, многократно использовать код и улучшить читаемость программы.

Определение функции в C++ включает в себя:

1. Тип возвращаемого значения:

Указывает тип данных, который функция будет возвращать после выполнения. Если функция ничего не возвращает, используется тип void.

2. Имя функции:

Произвольное имя, которое используется для вызова функции.

3. Список параметров:

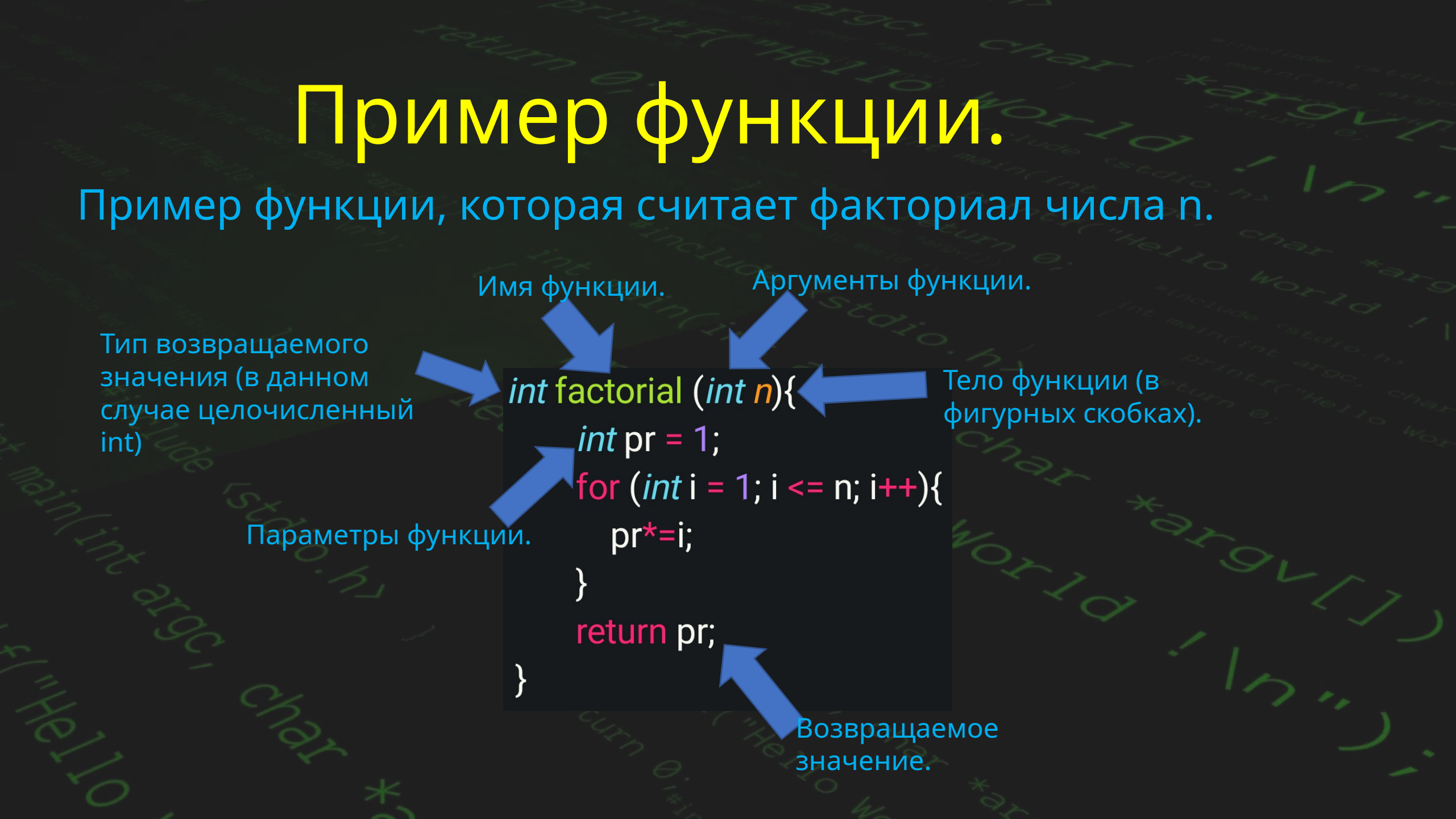
Перечень входных параметров функции, каждый из которых имеет свой тип данных и имя. Если параметров нет, используется пустой список ().

4. Тело функции:

Блок кода, заключенный в фигурные скобки {} и содержащий инструкции, которые выполняет функция.

Пример функции.

Пример функции, которая считает факториал числа n.




```
int factorial (int n){
    int pr = 1;
    for (int i = 1; i <= n; i++){
        pr*=i;
    }
    return pr;
}
```

Вызов функции в основной программе.

Чтобы вызвать функцию в основной программе (внутри main) необходимо указать **её имя** и присвоить значение переменной такого типа, **который указан как возвращаемый тип в данной функции**. При несовпадении код не скомпирируется и будет ошибка.

```
#include <iostream>
using namespace std;

int factorial (int n){
        int pr = 1;
        for (int i = 1; i <= n; i++){
            pr*=i;
        }
        return pr;
 }
int main()
{
    int n;
    n = factorial(6);
    cout << n;
    return 0;
}
```

Вывод
720

В данном примере создается целочисленная переменная n, которой присваивается значение, которое возвращает функция при аргументе = 6

Передача аргументов в функцию

Чтобы передать аргументы в функцию, необходимо после её имени в () указать необходимое число аргументов (скачала указывается тип, а потом имя аргумента)

```
int func(){
    return 0;
}
```

Функция, которая не получает на вход аргументов, и возвращает целочисленное значение

```
void func(int x, char a){
}
```

Функция, которая получает на вход целочисленный и символьный аргументы, и возвращает пустое значение

```
double func(double x, double y, int n){
    return 0.0;
}
```

Функция, которая получает на вход 2 вещественных и 1 целочисленный аргумент, и возвращает вещественное значение

Виды передачи аргументов

В C++, при передаче аргументов в функцию, существует два основных способа: **по значению и по ссылке**. При передаче **по значению** в функцию передается **копия аргумента**, а при передаче **по ссылке** передается **адрес** исходного аргумента (через символ **&**), позволяя функции изменить его значение

```
int func(double a, int& b){
        return 0;
}
```

В эту функцию аргумент a передается по значению, а b – по ссылке

Функции. Пример задачи.

```cpp
#include <iostream>
#include <cmath>
using namespace std;

double func(double x, double y){
    double z;
    z = sqrt(x*x + y*y);
    return z;
}

int main()
{
    for (double i = 1; i <= 3; i++){
        for (double j = 1; j <= 3; j++){
            double rass = func(i, j);
            cout << "От начала
            координат до точки"
            << i << " " << j << " = " <<
            rass << endl;
        }
    }
    return 0;
}
```

Вывод

```
От начала координат до точки 1
1 = 1.41421
От начала координат до точки 1
2 = 2.23607
От начала координат до точки 1
3 = 3.16228
От начала координат до точки 2
1 = 2.23607
От начала координат до точки 2
2 = 2.82843
От начала координат до точки 2
3 = 3.60555
От начала координат до точки 3
1 = 3.16228
От начала координат до точки 3
2 = 3.60555
От начала координат до точки 3
3 = 4.24264
```

Программа, которая вычисляет расстояние от начала координат до всех точек с целочисленными координатами от 1 до 3. Функция func содержит формулу для вычисления расстояния. В качестве аргументов координаты точки x и y.

Ошибки при работе с функциями

Самые частые ошибки при работе с функциями:

1) Несовпадение типов переменных возвращаемой функцией и присвоения значения.

```
#include <iostream>
using namespace std;

int f(int n){
    int summ = 0;
    for (int i = 0; i < n; i++){
        summ+=i;
    }
    return summ;
}

int main()
{
    double x = f(4);
    return 0;
}
```

В данном примере возвращаемый тип функции int, а присваиваем переменной типа double, ошибка.

2) Несовпадение типов переменных в аргументах функции.

```
#include <iostream>
using namespace std;

int f(int n){
    int summ = 0;
    for (int i = 0; i < n; i++){
        summ+=i;
    }
    return summ;
}

int main()
{
    int x = f(4.3);
    return 0;
}
```

В данном примере тип аргумента функции int, а передаём вещественное значение double, ошибка.

Ошибки при работе с функциями

3) Недостаток или переизбыток аргументов в функции.

```
#include <iostream>
using namespace std;

int f(int x, int y){
    return x + y;
}

int main(){
    int n = f(4);
}
```

В этом примере у функции f два аргумента, а передаем мы только 1, нехватка аргументов, ошибка