Функции — конспект темы

Функция, вас вызывают!

Функция — фрагмент кода, который вызывают из другого места программы. Каждому такому фрагменту присваивают уникальное имя — название функции.

Синтаксис функции:

```
void <имя функции>(<тип1> <аргумент1>, <тип2> <аргумент2>,) {
    <тело функции>
}
```

Тип возвращаемого значения указывают всегда — даже если функция ничего не возвращает. Каждое слово в названии функции пишут с большой буквы.

Функция логирования принимает целое число (id пользователя) и строку (поисковый запрос) и выводит отладочную строку вида «запрос сделан пользователем с таким id»:

```
#include <iostream>

using namespace std;

void PrintLogMessage(int user_id, string query) {
    cout << "Query "s << query << " was done by the user with id = "s << user_id << endl;
}

int main() {
    PrintLogMessage(237, "GET_STATUS"s);
}
```

Функция `PrintLogMessage` ничего не возвращает, поэтому перед названием — ключевое слово `void`.

Возврат значения и ошибки

Чтобы вернуть результат выполнения функции, используют ключевое слово `return`. Результат вернётся в место вызова функции:

```
#include <iostream>
#include <string>

using namespace std;

string Concatenate(string left, string right) {
    return left + " "s + right;
}

int main() {
    string str1 = "In C++"s;
    string str2 = "we trust"s;
    cout << Concatenate(str1, str2) << endl;
}
```

Этот код вычисляет результат работы функции `Concatenate` от аргументов `str1` и `str2`, а затем подставляет его в место вызова. Программа выведет:

```
In C++ we trust
```

`return` завершает выполнение функции.

Типичные ошибки:

1. Забыли вернуть значение из функции.

```
#include <iostream>

using namespace std;

int AddFive(int x) {
    x += 5;
}

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

2. Неправильный тип или количество аргументов.
3. Забыли объявить функцию, объявили её после применения или опечатались в названии:

```
#include <iostream>

using namespace std;

int Addfive(int x) {
    x += 5;
    return x;
}

int main() {
    cout << AddFive(5) << endl; // Функция с этим названием объявлена ниже
    cout << Addive(5) << endl; // А в вызове этой функции опечатка в названии
}

int AddFive(int x) {
    x += 5;
}
```

4. Недостижимый код.

```
#include <iostream>

using namespace std;

int ProcessValue(int value, int threshold) {
    if (value > threshold)
        value = value / 2;
        return value; // код далее в этой функции недостижим для исполнения
    if (value < threshold)
        value = value * 3;
        return value;
}

int main() {
    int value, threshold;
    cin >> value >> threshold;
    cout << endl << ProcessValue(value, threshold) << endl;
}
```

Автор кода забыл фигурные скобки для `if`-блока. Поэтому выполнение этой функции всегда заканчивается на первом `return`.

Зачем нужны функции

- С ними легче читать код.

- Функции позволяют переиспользовать код и избежать его дублирования.
- Функции делят код на логические фрагменты и облегчают поиск проблем.
- Функции позволяют разбить большую задачу на несколько более простых.
- Функции — это красиво!