Основы языков C/C++

Кафедра ЭО

Петрухин О.М.

Функции

```cpp
#include <iostream>

int main(){
    std::string name1 = "John";
    std::string name2 = "James";
    std::string name3 = "Jack";
    std::string name4 = "Ivan";
    std::string name5 = "Igor";
    std::string name6 = "Boris";

    std::cout << "Hello, " << name1 << "!\n";
    std::cout << "Hello, " << name2 << "!\n";
    std::cout << "Hello, " << name3 << "!\n";
    std::cout << "Hello, " << name4 << "!\n";
    std::cout << "Hello, " << name5 << "!\n";
    std::cout << "Hello, " << name6 << "!\n";
}
```

Что делать, если нужно поприветствовать ещё несколько человек?

Что делать, если теперь вместо `Hello` нужно написать `Goodbye`?

Функции

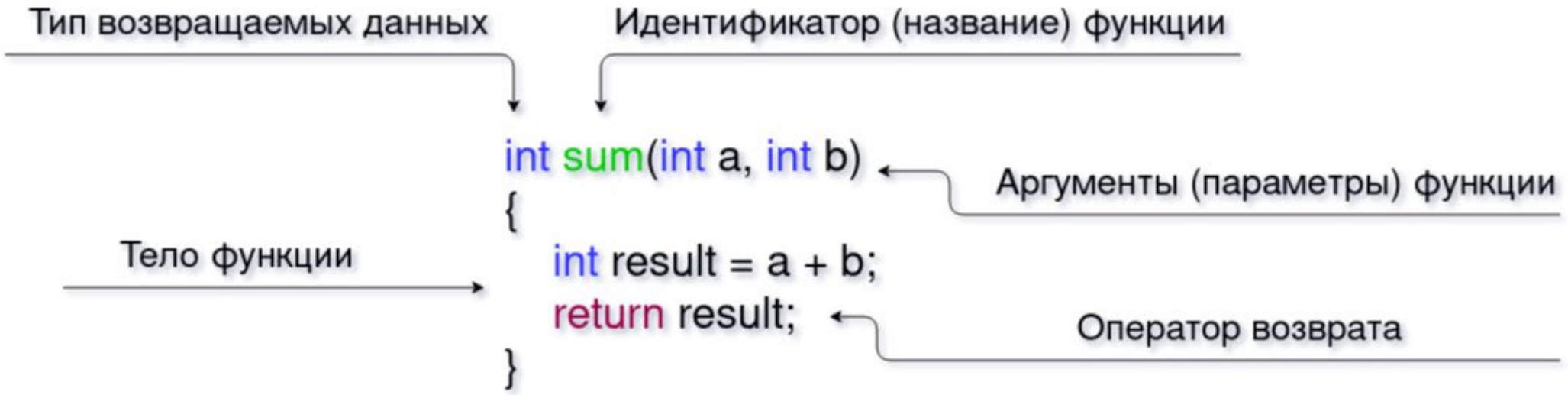
Если код нужно использовать несколько раз, то лучше всего его вывести в подпрограмму — функцию или процедуру. Это позволит значительно сократить объём кода.

```
#include <iostream>

void sayHello(std::string name){
    std::cout << "Hello, " << name << "!\n";
}

int main(){
    std::string name1 = "John";
    std::string name2 = "James";
    std::string name3 = "Jack";
    std::string name4 = "Ivan";
    std::string name5 = "Igor";
    int name6 = "Boris";

    sayHello(name1);
    sayHello(name2);
    sayHello(name3);
    sayHello(name4);
    sayHello(name5);
    sayHello(name6);
}
```

Функции

Функции — это блоки кода, выполняющие определенные операции. Если требуется, функция может определять входные параметры, позволяющие вызывающим объектам передавать ей аргументы. При необходимости функция также может возвращать значение как выходное.

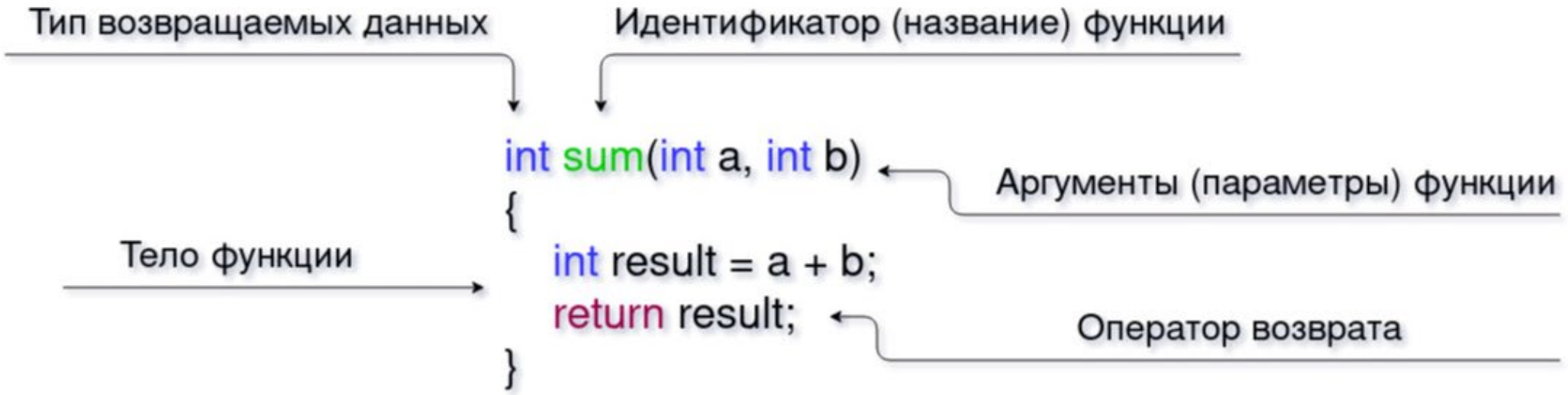
Функции полезны для инкапсуляции основных операций в едином блоке, который может многократно использоваться. В идеальном случае имя этого блока должно четко описывать назначение функции.



Функции

Функции — это блоки кода, выполняющие определенные операции. Если требуется, функция может определять входные параметры, позволяющие вызывающим объектам передавать ей аргументы. При необходимости функция также может возвращать значение как выходное.

Функции полезны для инкапсуляции основных операций в едином блоке, который может многократно использоваться. В идеальном случае имя этого блока должно четко описывать назначение функции.



Особенности вызова функций

Важно учесть: чтобы можно было вызвать функцию, о ней должно быть известно, то есть должна быть известна её **сигнатура** (имя функции, её параметры, возвращаемый тип). Для этого на момент вызова функции она уже должна быть объявлена **раньше**. Такой код не скомпилируется

```
#include <iostream>

int main()
{
    hello("Dima");
    return 0;
}

void hello(std::string name){
    std::cout << "Hello, " << name << std::endl;
}
```

Особенности вызова функций

Чтобы решить эту проблему, можно следить за правильным порядком объявления функций:

```
#include <iostream>

void hello(std::string name) {
    std::cout << "Hello, " << name << std::endl;
}

int main(int argc, char** argv)
{
    hello("Dima");
    return 0;
}
```

Однако если функций много, это может стать сложной задачей

Особенности вызова функций

Поэтому в C++ существует механизм объявления функции без тела. То есть описывается только сигнатура функции, а тело функции может быть описано другом месте. Скомпилируется такой код:

```
#include <iostream>

void hello(std::string name);

int main(){
    hello("Dima");
    return 0;
}
void hello(std::string name){
    std::cout << "Hello, " << name << std::endl;
}
```

Здесь функция `hello` сначала **объявляется**, а **определяется** уже после использования.

Определение функции состоит из объявления, а также текста, который представляет собой весь код между фигурными скобками

Возвращаемое значение

До этого мы постоянно использовали `void`, то есть ничего не возвращали.

Чаще всего функции что-то возвращают — результат проделанных действий. Например, объём не печатают на экран, а обычно возвращают, чтобы дальше его использовать.

Чтобы функция вернула какое-либо значение, нужно выполнить два условия:

1. Указать возвращаемый тип данных (не `void`)
2. В функции написать слово `return` и через пробел значение или переменную. Тип значения должен совпадать с типом из предыдущего пункта

Значение, которое возвращает функция, можно записать в переменную соответствующего типа

Возвращаемое значение

Примеры:

```cpp
#include <iostream>

int calcSquare(int a, int b);

int main()
{
    int vol = calcSquare(4, 2);
    std::cout << vol << std::endl;
    return 0;
}

int calcSquare(int a, int b)
{
    int square = a * b;
    return square;
}
```

```cpp
#include <iostream>

int min(int x, int y);

int main() {
    // Найти минимальное из 4 чисел
    int a, b, c, d;
    std::cout << "Введите 4 числа: ";
    std::cin >> a >> b >> c >> d;

    std::cout << "Минимальное число: " <<
    min(a, min(b, min(c, d))) << std::endl;
    return 0;
}

int min(int x, int y) {
    return x < y ? x : y;
}
```

Область видимости

Всё, что происходит в функции, в ней же и остаётся. То есть функции могут работать только с теми значениями, которые в них передали:

```
#include <iostream>

void printHello(){
    // Переменная name объявлена в другой функции, поэтому у
    // printHello() нет к ней доступа
    // Запуск такого кода приведёт к ошибке
    std::cout << "Hello, " << name << "!\n";
}

int main(){
    std::string name = "Igor";

    printHello();
}
```

Область видимости

Если переменная создаётся внутри какого-либо блока {}, то она будет доступна только в этом и во всех вложенных блоках; такие переменные называются **локальными.**

Если вы хотите, чтобы какая-нибудь переменная была доступна везде, то её нужно объявить за пределами каких-либо блоков; такие переменные называются **глобальными:**

```
#include <iostream>

//Глобальные переменные создаются вне функций
std::string name;

void printHello(){
    std::cout << "Hello, " << name << "!\n";
}

int main(){
    name = "Igor";

    printHello();
}
```

Как работают аргументы

Рассмотрим, что происходит, когда вы передаёте аргумент в функцию. Допустим, есть вот такой код:

```
#include <iostream>

void sum(int a){
    a = a + 500;
}

int main(){
    int a = 5;

    sum(a);

    std::cout << a << "\n";
}
```

Какое значение a будет выведено на экран?

КАИ
Крылья для жизни!

Спасибо за внимание!