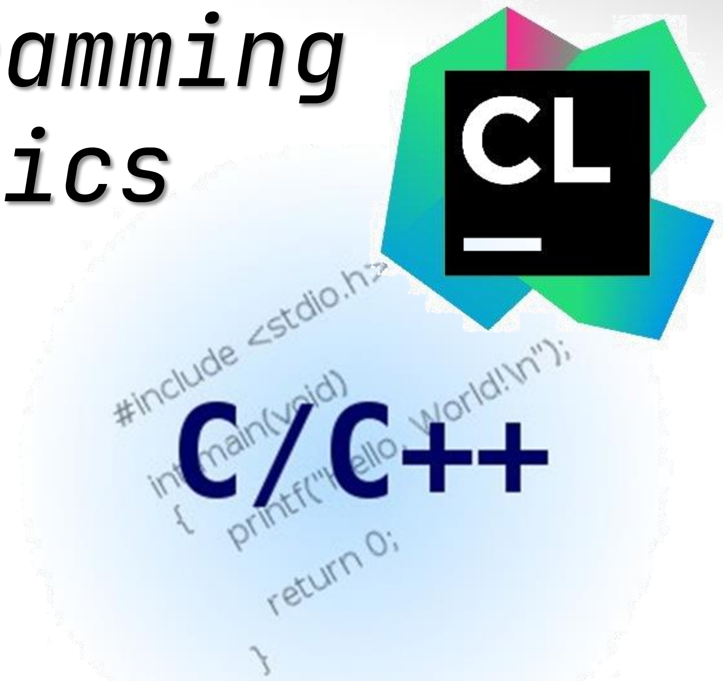
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Algorithms & Programming Programming Basics

C/C++ programming

(p.2 – Functions & Arrays)



```
#include <stdio.h>
int main(void)
{
    printf("Hello, World!\n");
    return 0;
}
```

Yevhen Berkunskyi, NUoS
eugeny.berkunsky@gmail.com
http://www.berkut.mk.ua

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

C++ Arrays

- Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

- To declare an array, define the variable type, specify the name of the array followed by square brackets and specify the number of elements it should store:

```
string cars[4];
```

C++ Arrays

- We have now declared a variable that holds an array of four strings. To insert values to it, we can use an array literal - place the values in a comma-separated list, inside curly braces:

```
string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
```

- To create an array of three integers, you could write:

```
int myNum[3] = {10, 20, 30};
```

Get the Size of an Array

- To get the size of an array, you can use the sizeof() operator:

```
int arr[] = {5, 2, 3, 7, 8};
int size = sizeof(arr);
cout << "size = " << size;
```

Result is:

```
size = 20
```

It is because the sizeof() operator returns the size of a type in bytes.

`int` type is usually 4 bytes, so from the example above,
4 x 5 (4 bytes x 5 elements) = 20 bytes.

Get the Size of an Array

- To get the size of an array, you can use the sizeof() operator:

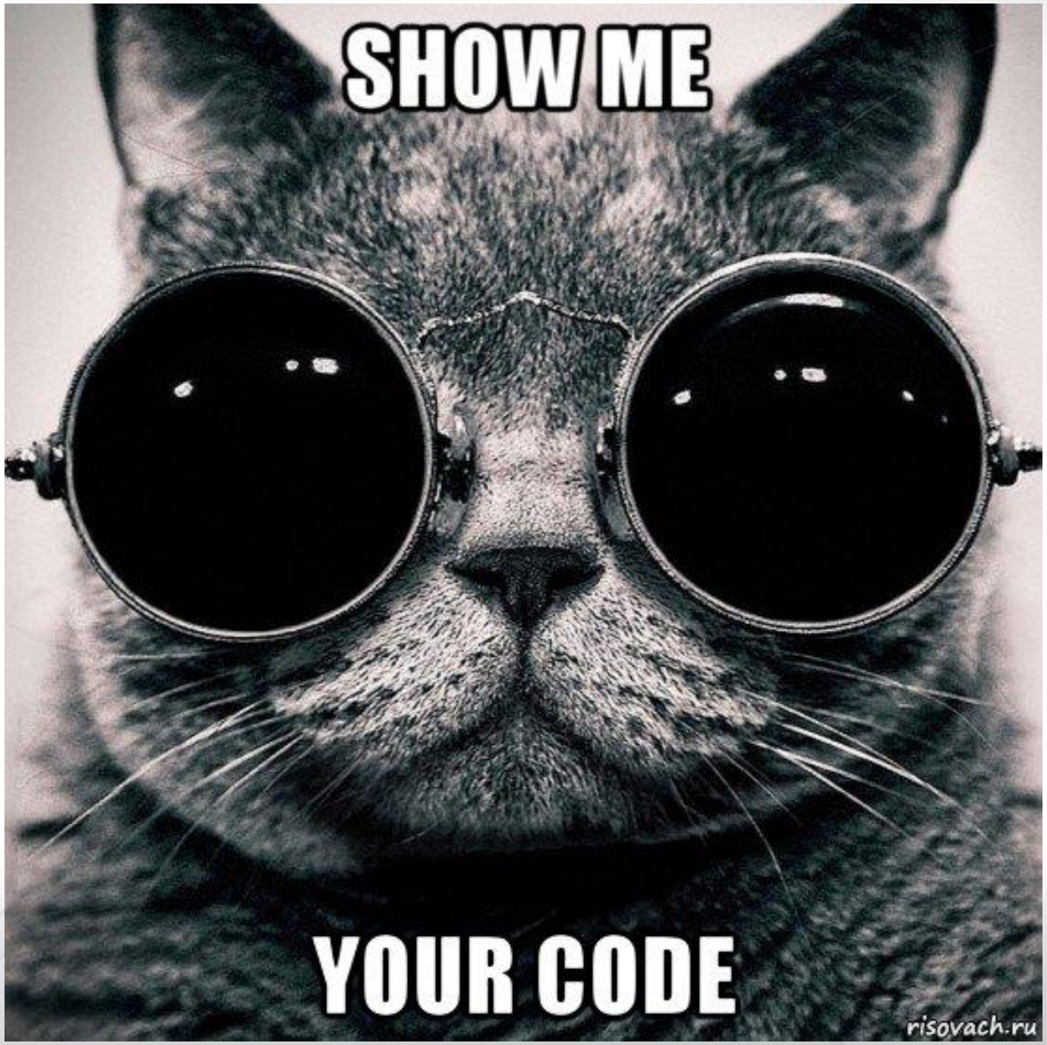
```
int arr[] = {5,2,3,7,8};
int getArrayLength = sizeof(arr) / sizeof(int);
cout << "length = " << getArrayLength;
```

Result is:

```
length = 5
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Let’s code!



C++ Functions

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- Functions are used to perform certain actions, and they are important for reusing code: ***Define the code once, and use it many times***.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Create a Function

- C++ provides some pre-defined functions, such as main(), which is used to execute code. But you can also create your own functions to perform certain actions.

- To create (often referred to as declare) a function, specify the name of the function, followed by parentheses ():

```
void myFunction() {
    // code to be executed
}
```

- **`myFunction()`** is the name of the function
- **`void`** means that the function does not have a return value.
- inside the function (the body), add code that defines what the function should do

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Call a Function

- Declared functions are not executed immediately. They are "saved for later use", and will be executed later, when they are called.

- To call a function, write the function's name followed by two parentheses () and a semicolon ;

- In the following example, `myFunction()` is used to print a text (the action), when it is called.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Call a Function

- In the following example, `myFunction()` is used to print a text (the action), when it is called:

```
#include <iostream>

using namespace std;

void myFunction() {
    cout << “I just got executed\n";
}

int main()
{
    myFunction();
    return 0;
}

// ------ result -----
I just got executed
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Call a Function

- A function can be called multiple times:

```
#include <iostream>

using namespace std;

void myFunction() {
    cout << “I just got executed\n";
}

int main()
{
    myFunction();
    myFunction();
    myFunction();
    return 0;
}

// ------ result -----
I just got executed
I just got executed
I just got executed
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Function Declaration and Definition

A C++ function consist of two parts:

- Declaration: the function's name, return type, and parameters (if any)
- Definition: the body of the function (code to be executed)

```
void myFunction() { // declaration
    // the body of the function
}
```

Note: *If a user-defined function, such as* `myFunction()` *is declared after the* `main()` *function, an error will occur*

Function Declaration and Definition

- However, it is possible to separate the declaration and the definition of the function - for code optimization.

- You will often see C++ programs that have function declaration above main(), and function definition below main(). This will make the code better organized and easier to read:

```cpp
void myFunction(); // declaration

int main()
{
    myFunction(); // call the function
    return 0;
}

void myFunction() { // function definition
    cout << "I just got executed";
}
```

C++ Function Parameters

Parameters and Arguments

- Information can be passed to functions as a parameter. Parameters act as variables inside the function.
- Parameters are specified after the function name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma:

```
void functionName(parameter1, parameter2, parameter3) {
    // code to be executed
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Example

```
#include <iostream>

using namespace std;

void myFunction(string name) {
    cout << name << " Dow\n";
}

int main()
{
    myFunction("John");
    myFunction("Liam");
    myFunction("Jane");
    return 0;
}
```

When a parameter is passed to the function, it is called an argument.
So, from the example above: name is a parameter, while John, Liam and Jane are arguments.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Parameters

In C++, parameters are passed to a function in one of the following ways:

- By value
- By reference
- By pointer

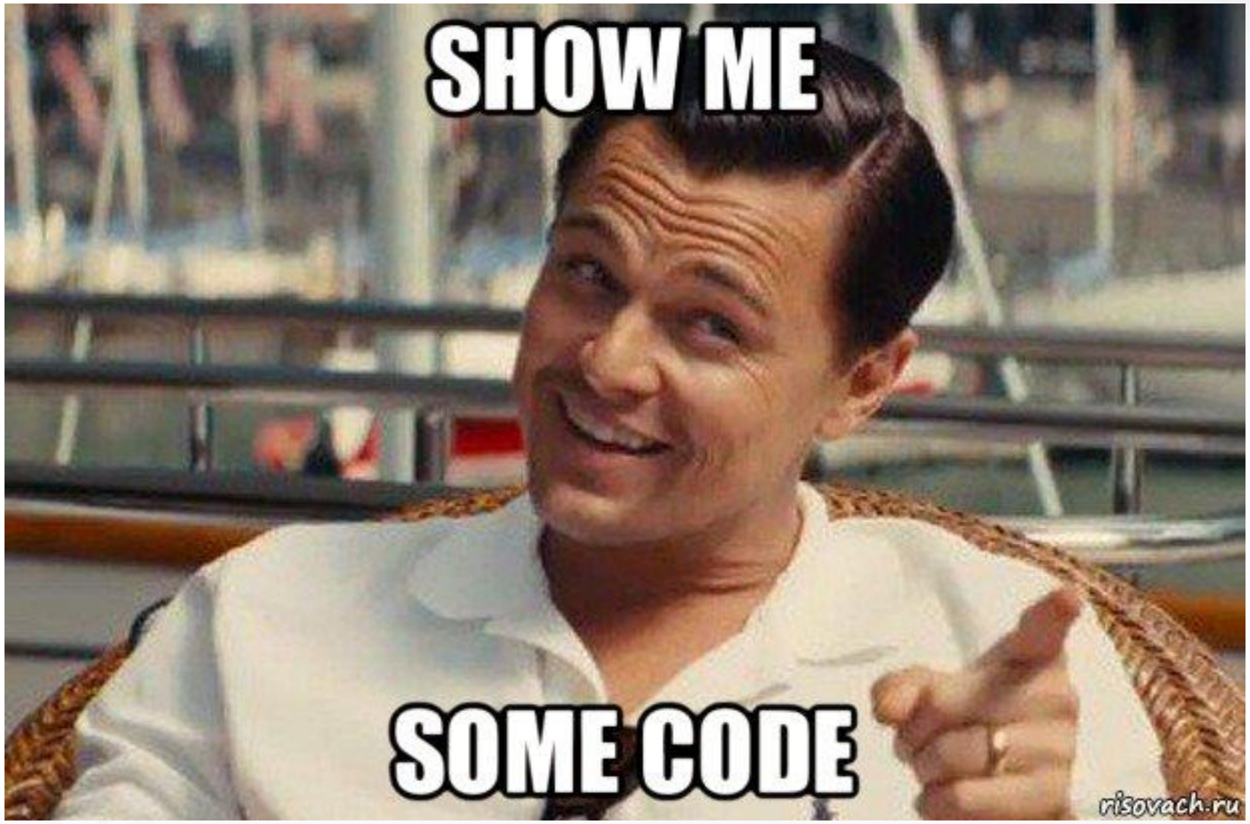
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

By value

```
#include <iostream>

using namespace std;

void swapNums(int x, int y) {
    int t = x; x = y; y = t;
}

int main() {
    int firstNum = 5;
    int secondNum = 7;
    cout << "Before swap: \n";
    cout << firstNum << " " << secondNum << "\n";
    swapNums(firstNum,secondNum);
    cout << "After swap: \n";
    cout << firstNum << " " << secondNum << "\n";
    return 0;
}
```

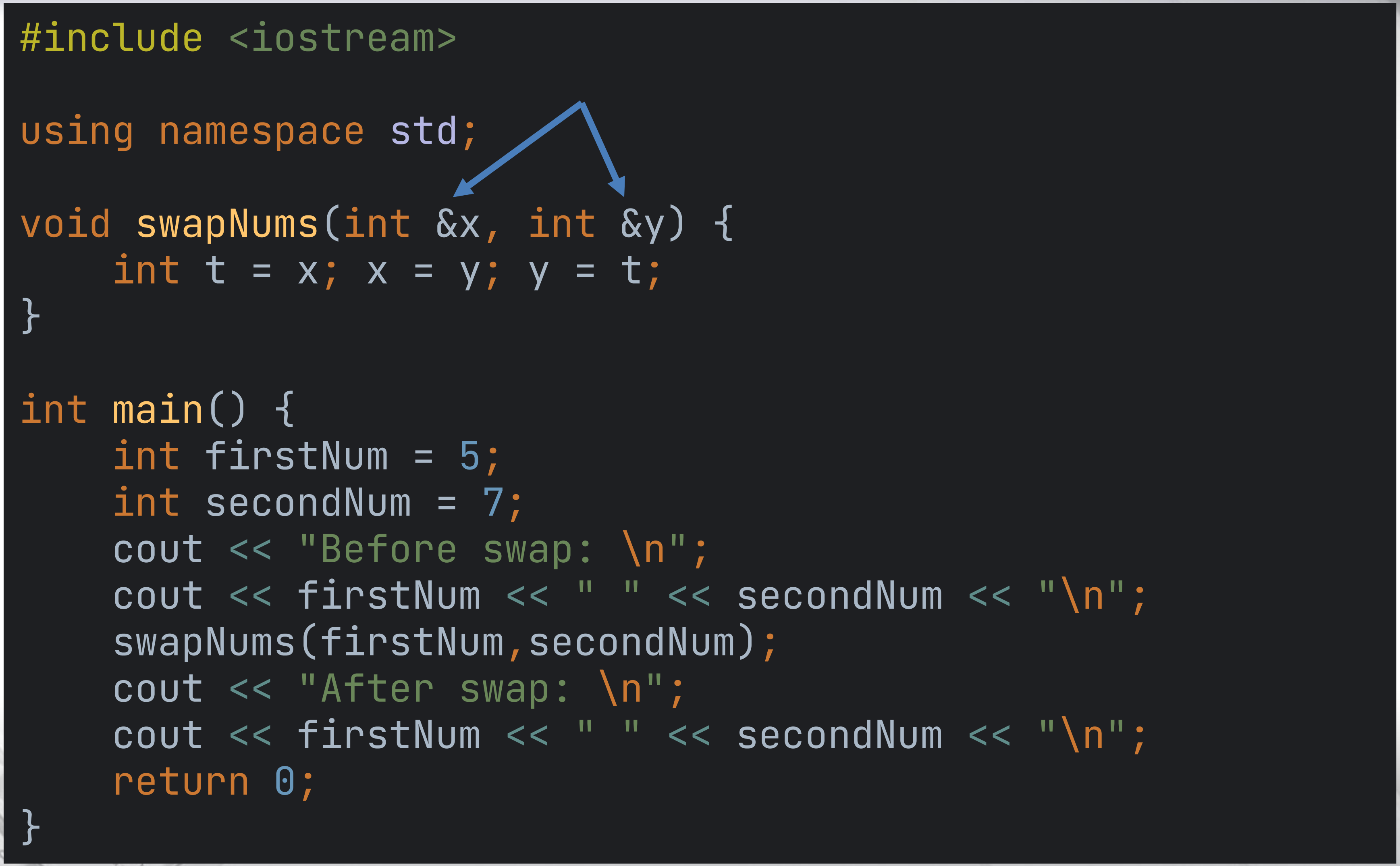
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Let’s see



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

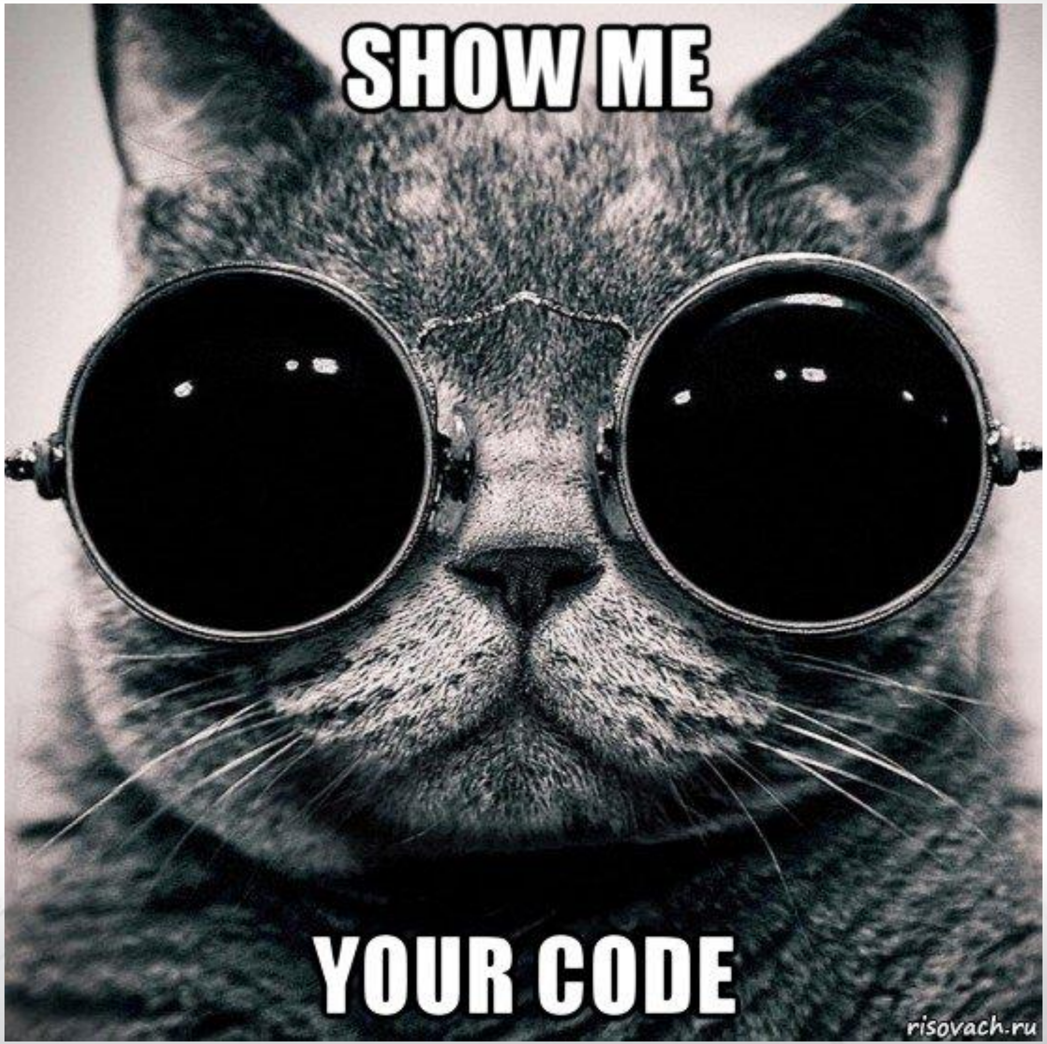
By reference



```
#include <iostream>

using namespace std;

void swapNums(int &x, int &y) {
    int t = x; x = y; y = t;
}

int main() {
    int firstNum = 5;
    int secondNum = 7;
    cout << "Before swap: \n";
    cout << firstNum << " " << secondNum << "\n";
    swapNums(firstNum,secondNum);
    cout << "After swap: \n";
    cout << firstNum << " " << secondNum << "\n";
    return 0;
}
```

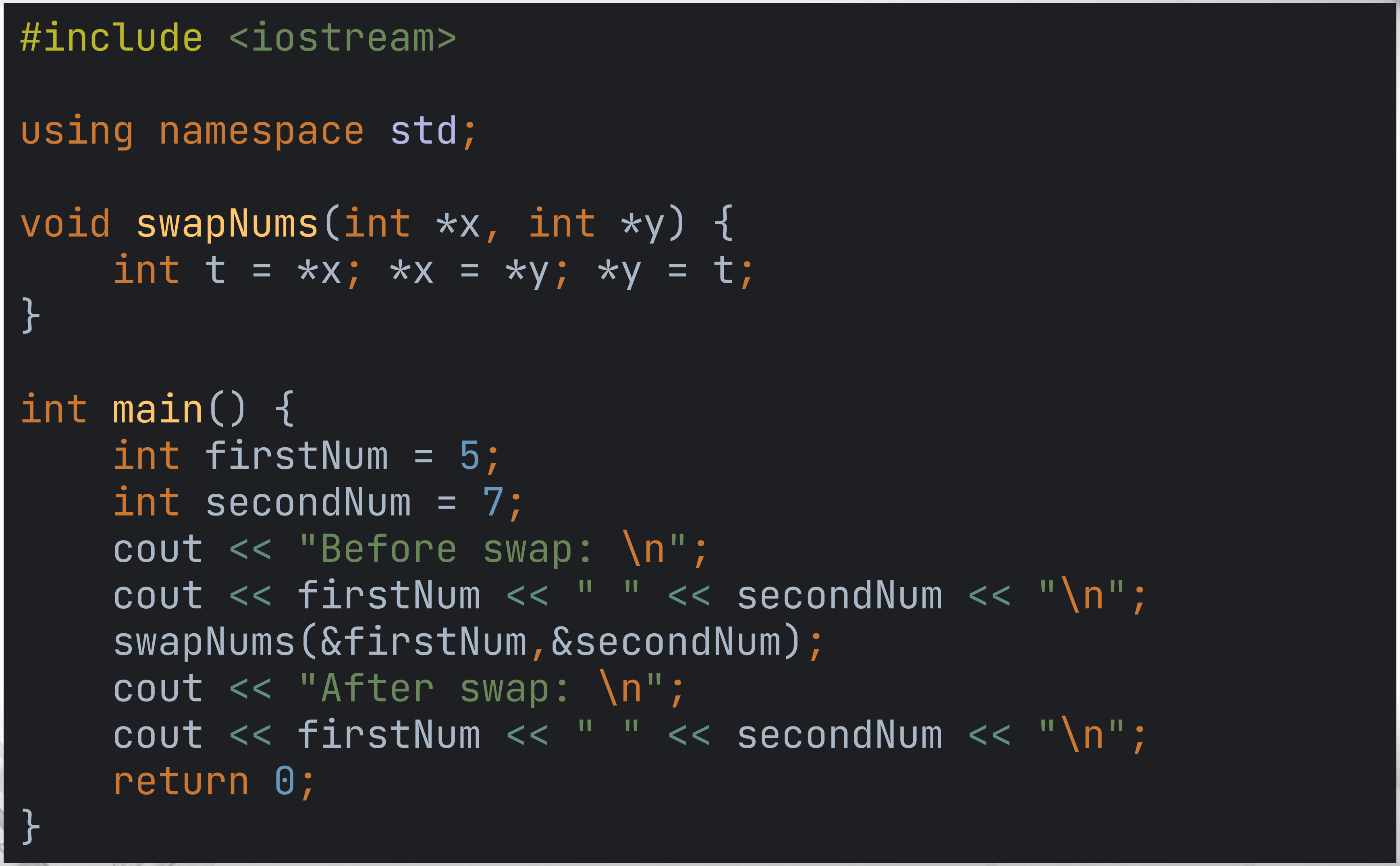
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Let’s code!



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

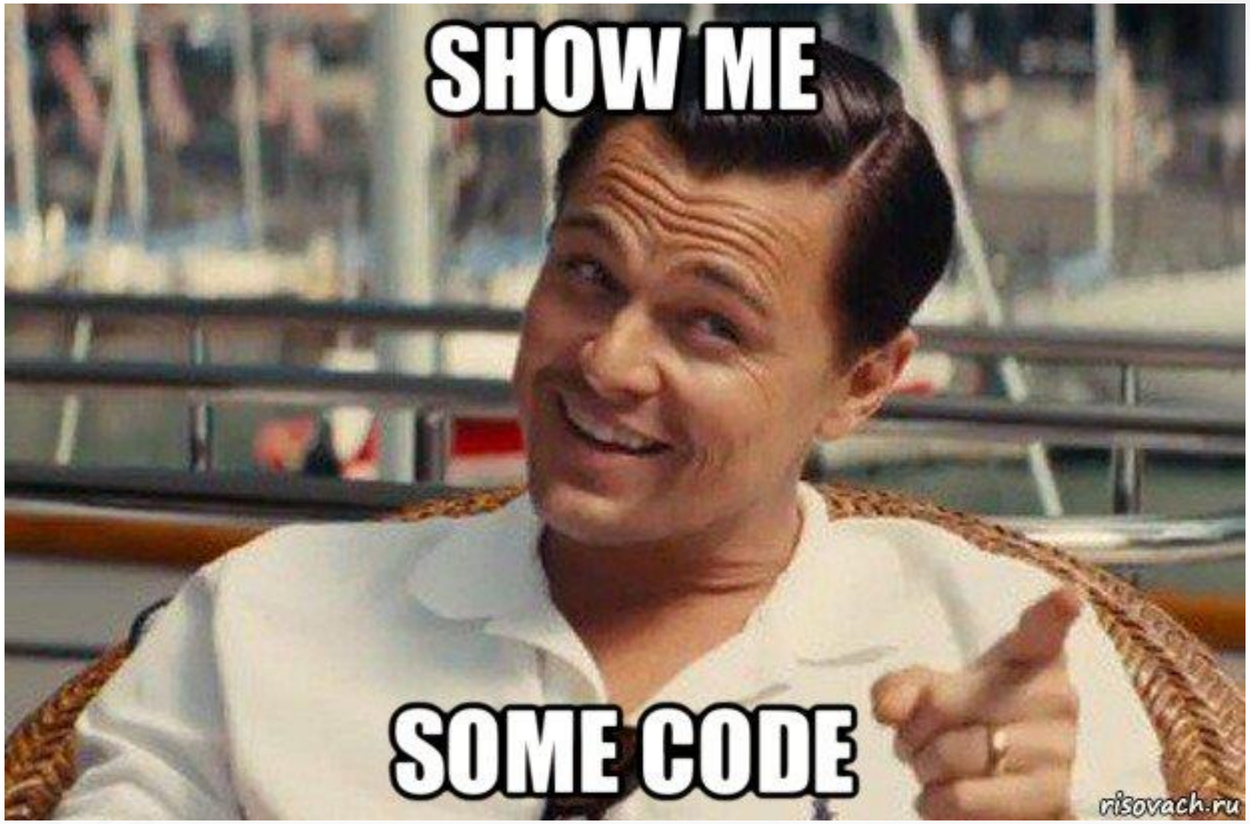
By pointer



```cpp
#include <iostream>

using namespace std;

void swapNums(int *x, int *y) {
    int t = *x; *x = *y; *y = t;
}

int main() {
    int firstNum = 5;
    int secondNum = 7;
    cout << "Before swap: \n";
    cout << firstNum << " " << secondNum << "\n";
    swapNums(&firstNum,&secondNum);
    cout << "After swap: \n";
    cout << firstNum << " " << secondNum << "\n";
    return 0;
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Let’s see



C++ Function Overloading

- In C++, two or more functions can have the same name if the number and/or type of arguments passed is different.
- These functions having the same name but different arguments are known as overloaded functions. For example:

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

int test(int a) { return a; }

double test(double a) { return a; }

int test(int a, int b) { return a + b; }
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

C++ Function Overloading

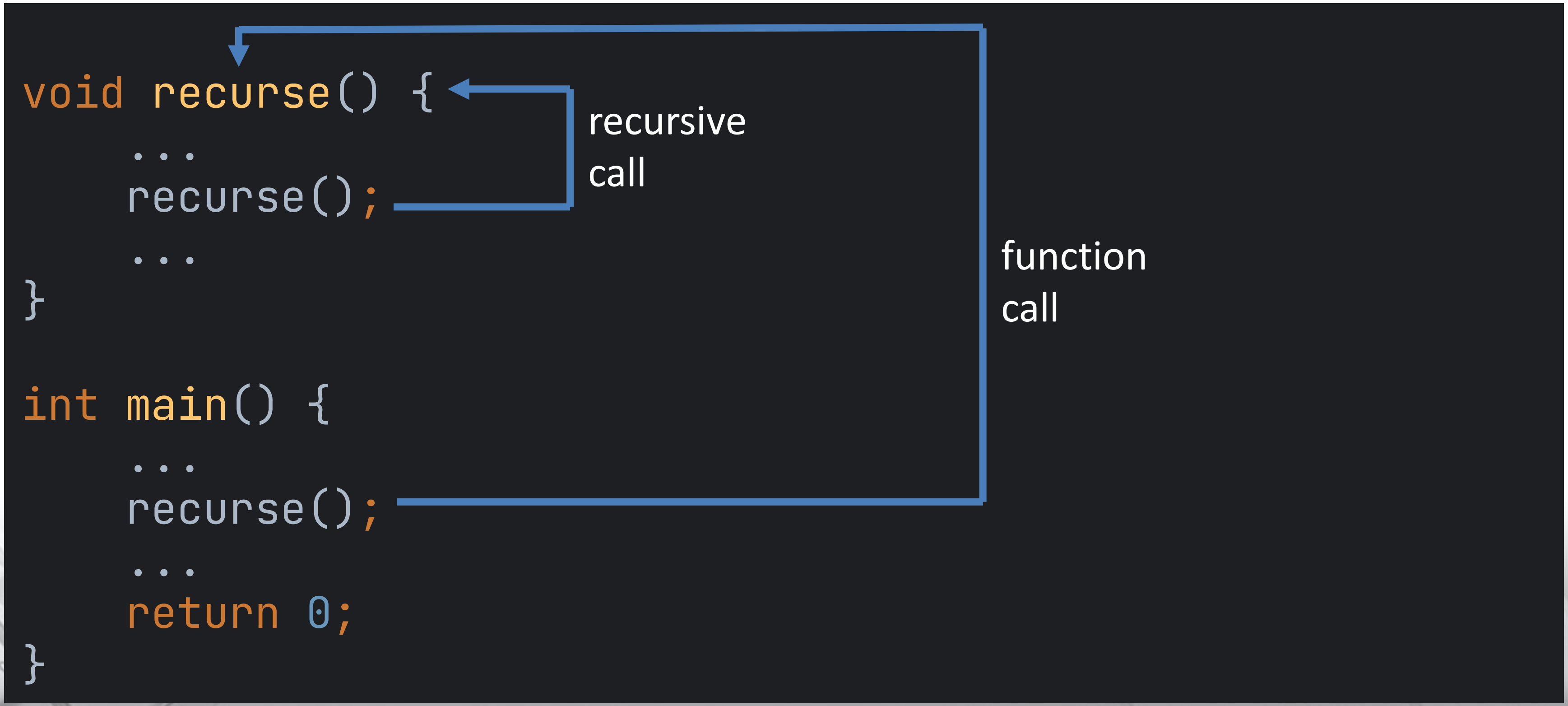
- Overloaded functions may or may not have different return types but they must have different arguments. For example

```
// error!!!

int test(int a) { return a; }

double test(int b) { return 1.0*b; }
```

Here, both functions have the same name, the same type, and the same number of arguments. Hence, the compiler will throw an error.

C++ Recursion

- A function that calls itself is known as a recursive function.
- And, this technique is known as recursion.



```
void recurse() {
    ...
    recurse();
    ...
}

int main() {
    ...
    recurse();
    ...
    return 0;
}
```

C++ Recursion

- The recursion continues until some condition is met.
- To prevent infinite recursion, `if...else` statement (or similar approach) can be used where one branch makes the recursive call and the other doesn't.

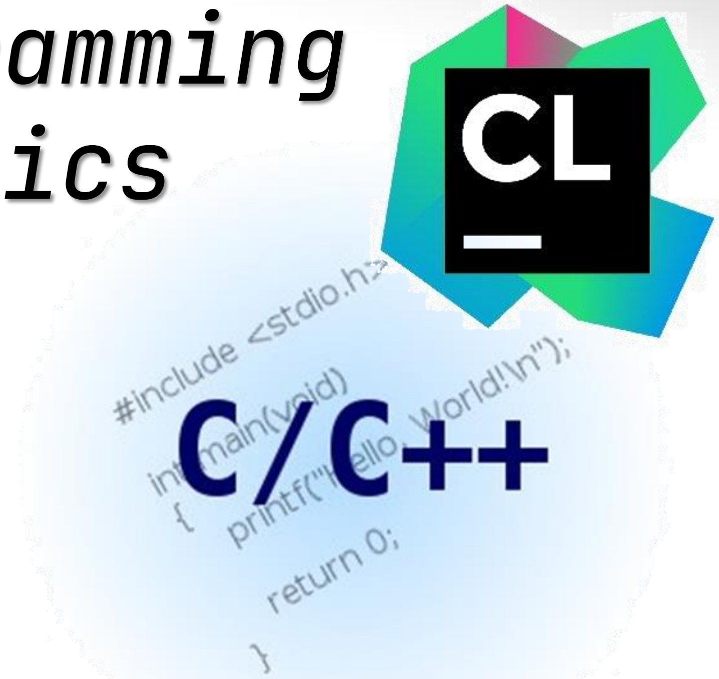
Example: Factorial

```cpp
#include <iostream>

using namespace std;

long long factorial(int);

int main() {
    int n;

    cout << "Enter a non-negative number: ";
    cin >> n;
    long long result = factorial(n);
    cout << "Factorial of " << n << " = " << result;

    return 0;
}

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

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Algorithms & Programming Programming Basics

C/C++ programming
(p.2 – Functions & Arrays)



```
#include <stdio.h>
int main(void)
{
    printf("Hello, World!\n");
    return 0;
}
```

Yevhen Berkunskyi, NUoS
eugeny.berkunsky@gmail.com
http://www.berkut.mk.ua