НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Algorithms & Programming
Programming Basics

C/C++ programming

(p.1 – language concepts & flow control)

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

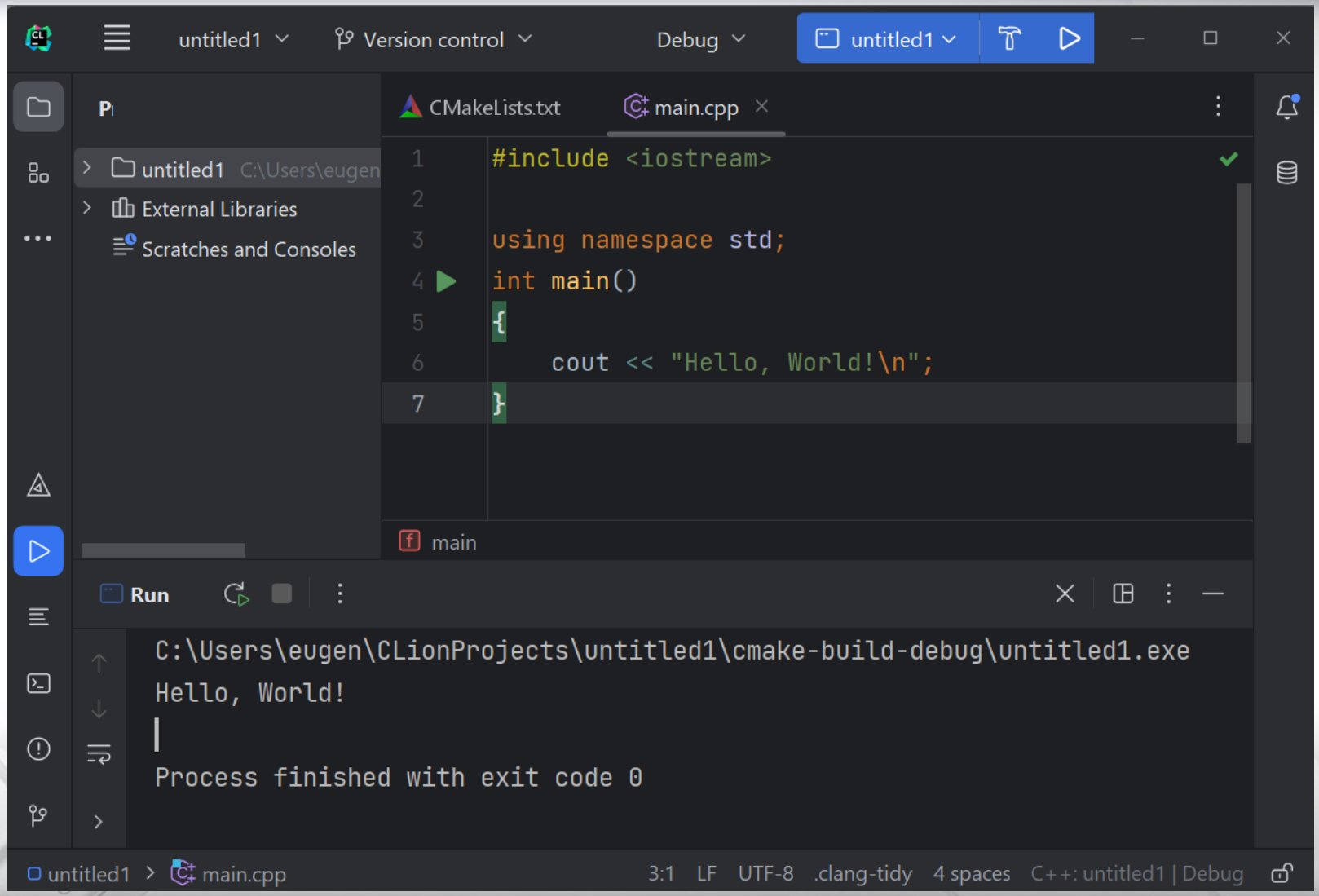
- C++ is a multipurpose programming language
- It can be used for writing every type of software that the programmer wants.
- It can be from a simple program like computing the sum of two numbers up to a full operating system (for example, Windows) and videogames (PS3, PS4, PS5, XBOX).

C/C++

- Through the course we will be using the GNU C++ compiler, G++, along with the IDE CLion.
- All of the code will be written in standard C++.
- CLion can be downloaded from https://www.jetbrains.com/clion/
- Compilers can be downloaded from
  - https://cygwin.com/install.html
  - https://sourceforge.net/projects/mingw-w64/

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

IDE & Compilers



Libraries

- The full extension of C++ has libraries for many purposes: input and output, math, memory management, time, file processing, string processing and many more features.
- However, when we start to code, we must define which of these libraries we want to use in our program.
- Since 2003, C++ also implemented a feature named **namespaces**, which are another way to organize all of the stuff inside the language.

The *main* function

- In C++, all programs must contain a function (or a piece of code) named **main**.
- The purpose of this function is to indicate the compiler where will the program start its execution.

```
#include <iostream>

int main() {
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Input and Output (I/O)

- Almost every program in C++ needs an input and an output. There are some exceptions to the rule, but normally, there is always I/O.
- The input is all the data that enters the computer, like numbers, words.
- The output are the data that is showed to the screen (also named console).
- The library used in C++ for this operations is named **iostream**.

Data Types and variables

- When we work with a program, there are many types of data we can use.
- For example: numbers, text, and some special types of data. All of these are called Data Types.
- In C++ we have 3 categories: numeric, alphabetic and binary.

Data Types and variables

- In the numeric category we have 3 types of data:
  - **int –** Represent integer numbers
  - **float** – Represents numbers with 7 decimal positions after the point
  - **double** – Represents numbers with 15/16 decimal positions after the point
- There are more specific numeric data types but for the purposes of this course, these three are enough.

Data Types and variables

- For the alphabetic category we have two types of data:
  - **char** – Store one alphanumeric digit
  - **string** – Store a sequence of alphanumeric digits

Data Types and variables

- And in the binary category, there is only one data type:
  - **bool** – Store a true or false. Zero or One.
- The bool data type may sound very useless at first but it has its handy applications when programming some specific routines.

Data Types and variables

- In C++ there is a concept named **variable**.
- A variable can store information of any of these mentioned types. Imagine a **variable like a box** that **can hold some value** for a certain time and **you can do** any **operation with that value**.
- In C++, before using variables, we must declare them. How? Like this:

```
<datatype> <nameOfVariable>
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Data Types and variables

```
int numberA;
int numberB;
int result;
```

- You can name the variables in the way that you want, but it is recommendable to give them a name similar to the use it will have.
- How can we assign a value?
- We use the = operator:

```
numberA = 10;
numberB = 5;
result = numberA + numberB;
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Data Types and variables

- The complete program would look like this:

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

int main() {
    int numberA;
    int numberB;
    int result;

    numberA = 10;
    numberB = 5;
    result = numberA + numberB;

    cout << result;
    return 0;
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Data Types and variables

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

int main() {
    int numberA;
    int numberB;
    int result;
    cout << "Input first value: ";
    cin >> numberA;
    cout << "Input second value: ";
    cin >> numberB;
    result = numberA + numberB;
    cout << "The sum is: " << result;
    return 0;
}
```

String I/O

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string name;
    cout << "What's your name? ";
    cin >> name;
    cout << "Goodbye " << name;
    return 0;
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Escape Sequences

In C++, there are special sequences that allow the programmer a better distribution of the information that is displayed.
The sequences are:

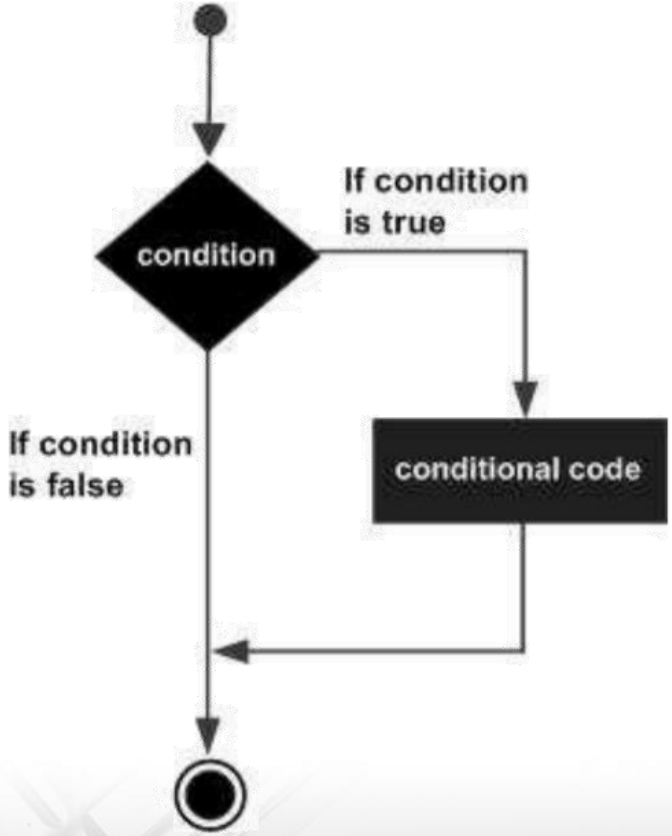
\n – End of line
\t – Insertion of tabular space

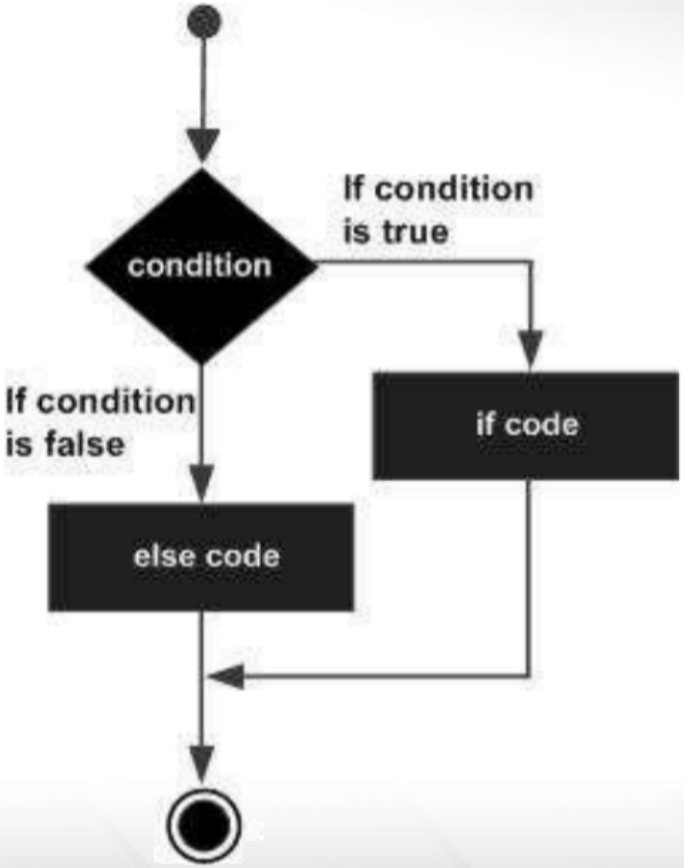
With these two sequences, you can design a well distributed interface.

```
cout << "Here is a line\n" << "Here is another line\t"
<< "I am after a tab space";
```

Program Flow

if… if…else…


condition
If condition is true
conditional code
If condition is false


condition
If condition is true
if code
If condition is false
else code

if statement

Structure of **if** with curly braces:

```
if ( condition) {
   // statements for execute when condition is true
}
```

Example

```
if ( 7 > 6 ) {
   cout << ”seven is more than six”;
}
```

- Without curly braces, there is one statement in body, only first statement.
- If you need many statements in the body of if statement, you should use curly braces.

if statement

Sometimes, when condition is false, it will be convenient to execute some code, different to code when condition is true

```
if ( condition ) {
    /* execute this code, if condition is true */
}
else {
    /* execute this code, if condition is false */
}
```

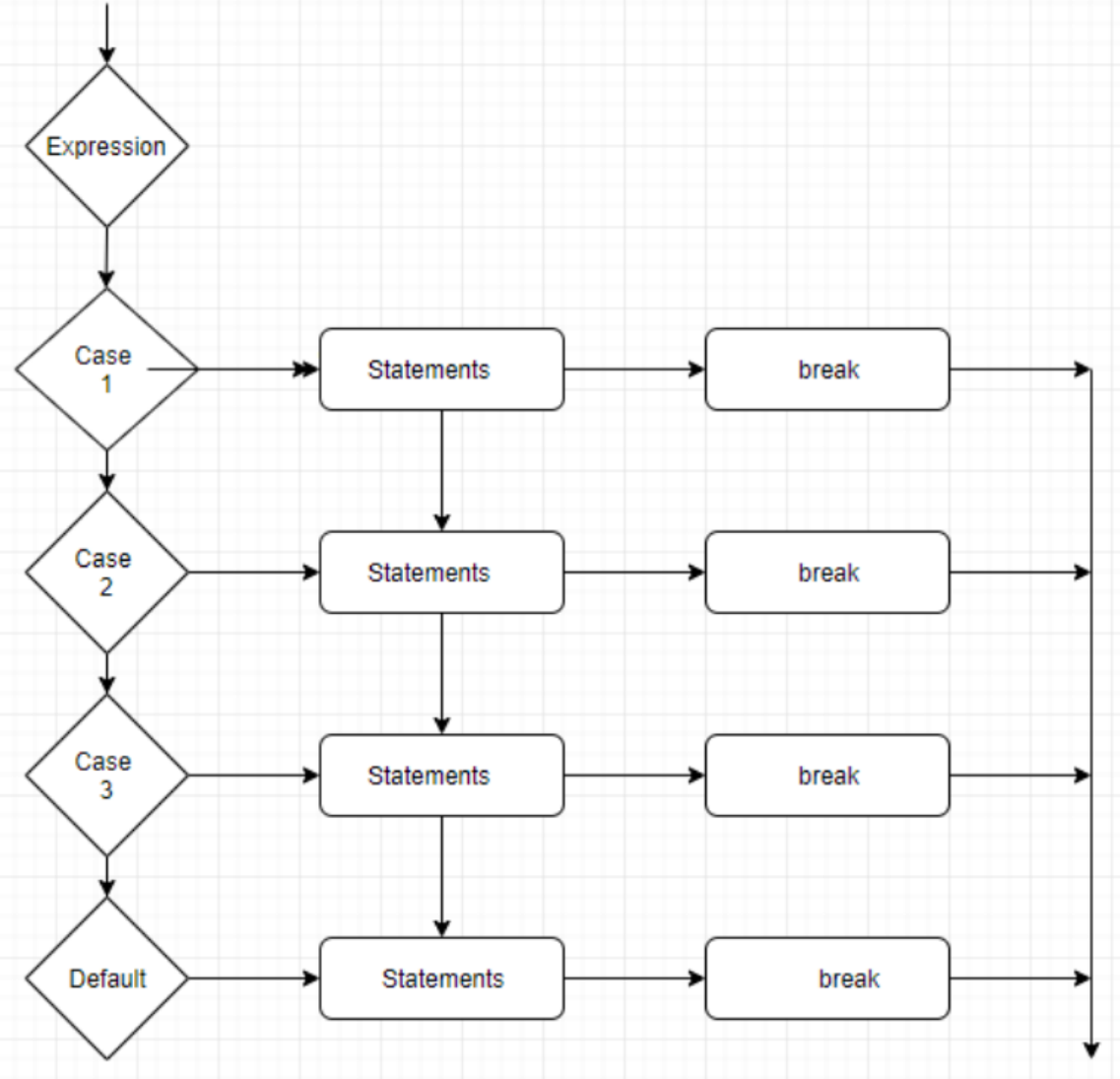
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Nested if statements

```
int age;
cout << "How old are you ? ";
cin >> age;
if ( age < 100 ) {
    cout << "You are so young!\n";
} else if ( age == 100 ) {
    cout << "Youth is over\n";
} else {
    cout << "Don't live so many years\n";
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

switch - case



switch - case

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

switch – case example

```
int day = 4;
switch (day) {
  case 1: cout << "Monday";
    break;
  case 2: cout << "Tuesday";
    break;
  case 3: cout << "Wednesday";
    break;
  case 4: cout << "Thursday";
    break;
  case 5: cout << "Friday";
    break;
  case 6: cout << "Saturday";
    break;
  case 7: cout << "Sunday";
    break;
}
```

C++ While Loop

- The while loop loops through a block of code as long as a specified condition is true:

```
while (condition) {
  // code block to be executed
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

C++ While Loop example

- In the example below, the code in the loop will run, over and over again, as long as a variable (i) is less than 5:

```
int i = 0;
while (i < 5) {
  cout << i << "\n";
  i++;
}
```

C++ Do/While Loop

- The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

```
do {
  // code block to be executed
} while (condition);
```

C++ Do/While Loop

- The example below uses a do/while loop. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

```
int i = 0;
do {
  cout << i << "\n";
  i++;
} while (i < 5);
```

C++ For Loop

- When you know exactly how many times you want to loop through a block of code, use the for loop instead of a while loop:

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

C++ For Loop

- When you know exactly how many times you want to loop through a block of code, use the for loop instead of a while loop:

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

Statement 1 is executed (one time) before the execution of the code block.

Statement 2 defines the condition for executing the code block.

Statement 3 is executed (every time) after the code block has been executed.

C++ For Loop

- The example below will print the numbers 0 to 4:

```
for (int i = 0; i < 5; i++) {
  cout << i << "\n";
}
```

This example will only print even values between 0 and 10:

```
for (int i = 0; i <= 10; i = i + 2) {
  cout << i << "\n";
}
```

The foreach Loop

- There is also a "for-each loop" (introduced in C++ version 11 (2011), which is used exclusively to loop through elements in an array (or other data sets):

```
for (type variableName : arrayName) {
  // code block to be executed
}
```

- Example

```
int myNumbers[] = {10, 20, 30, 40, 50};
for (int i : myNumbers) {
  cout << i << "\n";
}
```

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:

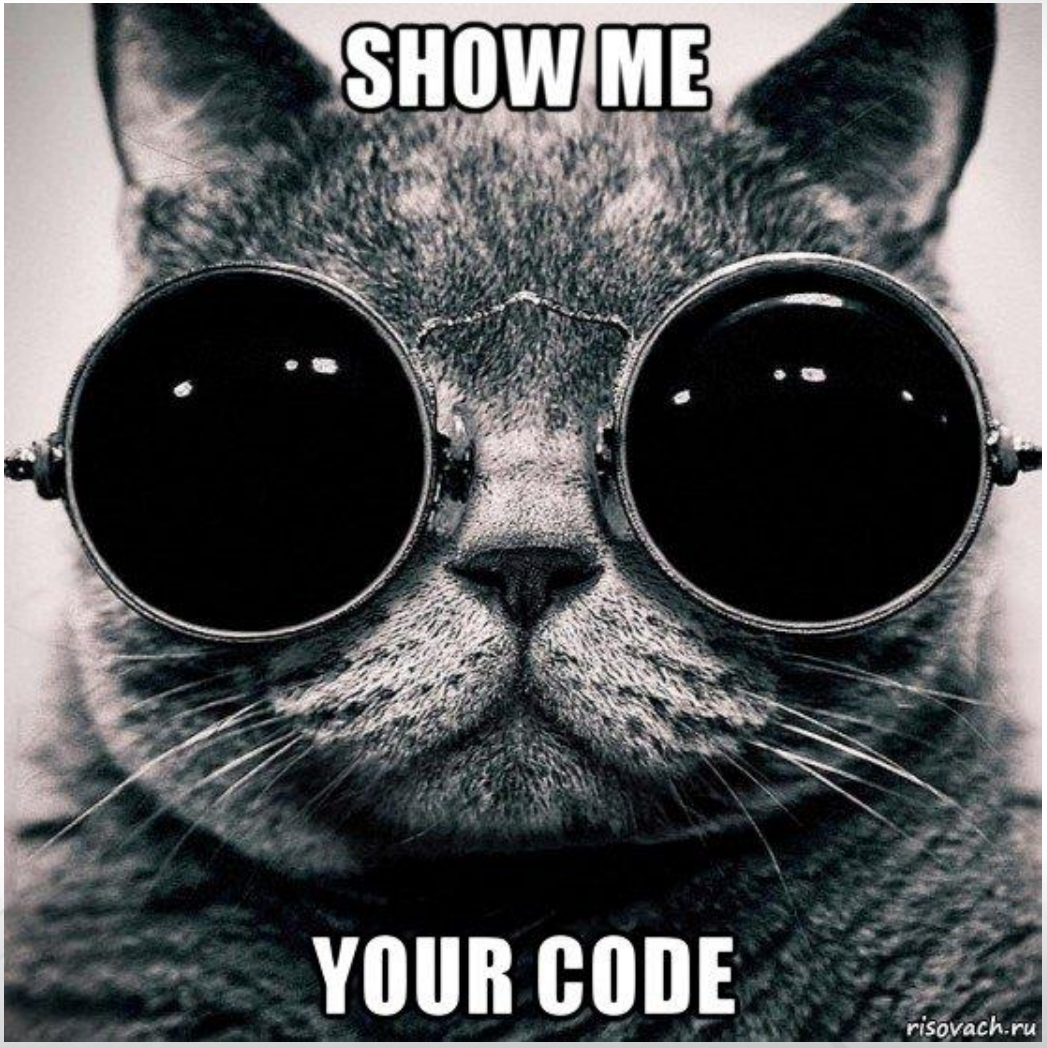
```
string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
```

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

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

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Let’s code!



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Questions?

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Algorithms & Programming
Programming Basics

C/C++ programming

(p.1 – language concepts & flow control)

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