НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Algorithms & Programming
Programming Basics

C/C++/Kotlin programming

(p.6 – Files + Structures / Data Classes)

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

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Text & Binary files

- Text files
  - contain text representation of information
  - for writing / reading you need to convert
  - can be viewed / read in text editor
- Binary files
  - contain data in the same format as in memory
  - no conversion needed
  - a program is needed to view / read

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Writing structures to files

- To write structure to a file in C++, you can follow these steps:
  1. Define the structure that you want to write to the file.
  2. Open a file stream using the ofstream class.
  3. Check if the file stream is open.
  4. Write the structure to the file using the write function.
  5. Close the file stream.

Let’s look on this steps deeper

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Writing structure to file

1. Define the structure that you want to write to the file

```
struct Person {
    char name[50];
    int age;
    double height;
};
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Writing structure to file

2. Open a file stream using the ofstream class

```
ofstream outFile(“file.txt");
```

3. Check if the file stream is open

```
if (!outFile.is_open()) {
    cerr << "Error: Cannot open file\n";
    return 1;
}
```

Writing structure to file

4. Write the structure to the file using the write function

```
Person person = {"John", 30, 1.8};
outFile.write(reinterpret_cast<char*>(&person),
              sizeof(person));
```

Note that we use `the reinterpret_cast` operator to convert the pointer to the structure into a pointer to a char.

5. Close the file stream

```
outFile.close();
```

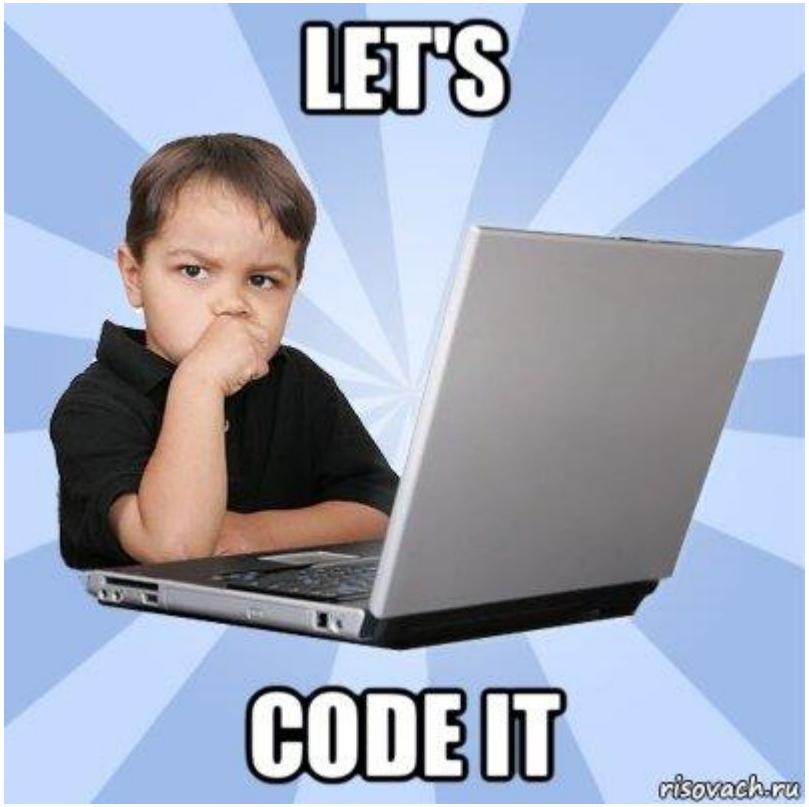
Writing structure to file

- Note that this method writes the entire structure to the file as a binary data. If you want to write the structure in a human-readable format, you can use the << operator with the file stream object, like this:

```
outFile << "Name: " << person.name << "\n";
outFile << "Age: " << person.age << "\n";
outFile << "Height: " << person.height << "\n";
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Demo



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Reading structure from file

Reading a structure from a file in C++ involves the following steps:

1. Define the structure that you want to read from the file.
2. Open the file in read mode using the ifstream class
3. Check if the file was opened successfully. If it was not, then you can't read from the file.
4. Read the data from the file and store it in a variable of the structure type.
5. Do something with the data that you just read
6. Close the file when you're done reading from it

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Reading structure from file

1. Define the structure that you want to read from the file

```
struct Person {
    char name[50];
    int age;
    double height;
};
```

Reading structure from file

2. Open the binary file in read mode using the `ifstream` class. For example, if the binary file is called "`people.bin`", you can open it as follows

```
ifstream inFile("people.bin", ios::binary);
```

Note the ios::binary flag, which specifies that we are reading a binary file

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Reading structure from file

3. Check if the file was opened successfully. If it was not, then you can't read from the file.

```
if (!inFile.is_open()) {
    cout << "Unable to open file" << endl;
    return 1;
}
```

4. Read the data from the file and store it in a variable of the structure type

```
Person p;
inFile.read((reinterpret_cast<char*>)&p, sizeof(Person));
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Reading structure from file

5. Do something with the data that you just read

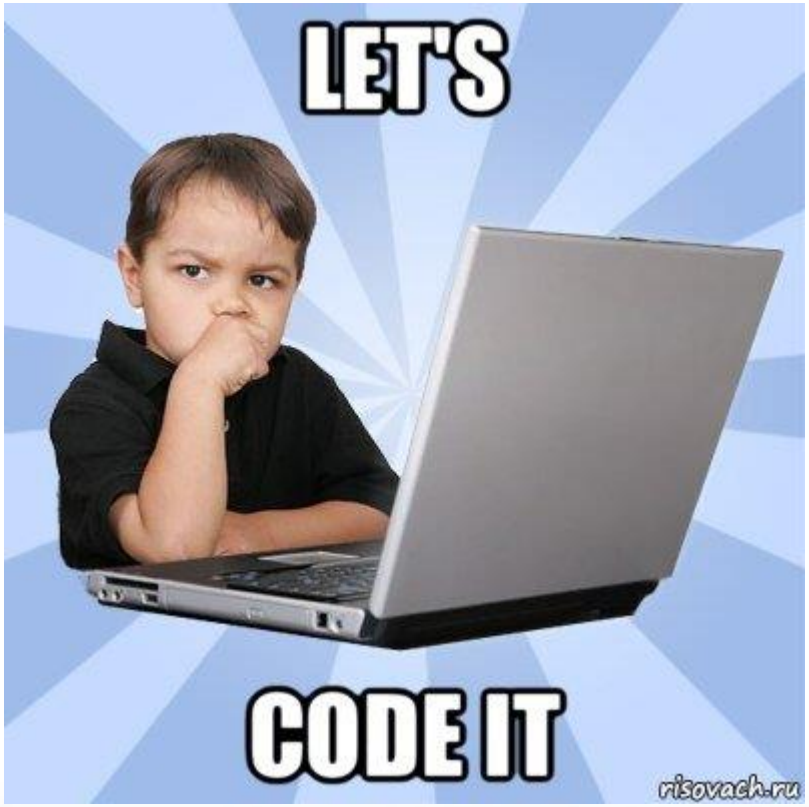
…

6. Close the file when you're done reading from it

```
inFile.close();
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Demo



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Writing array of structures

```
// Open file
ofstream outFile("file.bin");

// Check if file opened
if (!outFile.is_open()) {
    cerr << "Error: Cannot open file\n";
    return 1;
}

// writing data to file
outFile.write(reinterpret_cast<char*>(people),
              sizeof(people));
// close file for clear buffer
outFile.close();
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Reading array of structures

```
// Open file (also it can be at one line)
ifstream inFile;
inFile.open("file.bin", ios::binary);

// Check if file opened
if (!inFile.is_open()) {
    cout << "Unable to open file" << endl;
    return 1;
}

// Declare array
Person people [SIZE];
// Reading data from file
inFile.read(reinterpret_cast<char*>(people),
            sizeof(Person) * SIZE);

// Close file
inFile.close();
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Kotlin: Data Classes & Files

- In Kotlin, the java.io.Serializable interface is used to mark a class as serializable, just like in Java
- To make a class serializable, you simply need to implement the java.io.Serializable interface:

```
data class Person(
    var name: String,
    var age: Int,
    var height: Double
) : Serializable
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Kotlin: Data Classes & Files

```
val person = Person("Alice", 19, 1.75)
val file = File("person.ser")

ObjectOutputStream(FileOutputStream(file)).use { out ->
    out.writeObject(person)
}

ObjectInputStream(FileInputStream(file)).use { inp ->
    val decodedPerson = inp.readObject() as Person
    println(decodedPerson)
}
```

Kotlin: Data Classes & Files

```
val person = Person("Alice", 19, 1.75)
val file = File("person.ser")

ObjectOutputStream(FileOutputStream(file)).use { out ->
    out.writeObject(person)
}

ObjectInputStream(FileInputStream(file)).use { inp ->
    val decodedPerson = inp.readObject() as Person
    println(decodedPerson)
}
```

But there is also a more modern approach…

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Kotlin: Data Classes & Files

- To serialize data class objects to a file in Kotlin, you can use the built-in Kotlin serialization library, which allows you to serialize and deserialize Kotlin objects to and from various data formats, including JSON, CBOR, and ProtoBuf.
- Let’s look on example of how to serialize data class objects to a JSON file.

Serializing data class objects

- In Kotlin, you can serialize a data class using one of the serialization frameworks available, such as Kotlinx serialization or Jackson.
  - First, add the following dependencies to your build.gradle file:

```
plugins {
    kotlin("jvm") version "1.9.23"
    kotlin("plugin.serialization") version "1.9.23"
}
```

```
dependencies {
    testImplementation(kotlin("test"))
    implementation(
 "org.jetbrains.kotlinx:kotlinx-serialization-json:1.6.0")
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Serializing data class objects

1. Define a data class that you want to serialize

```
@Serializable
data class Person(
    var name: String,
    var age: Int,
    var height: Double
)
```

Note about annotation your data class with @Serializable

Serializing data class objects

2. Use the Json.encodeToString function to serialize an instance(s) of your data class to JSON

```
val p1 = Person("John", 21, 1.8)
val p2 = Person("Piter", 20, 1.7)
val p3 = Person("Mary", 19, 1.65)

val a = arrayOf(p1, p2, p3)

val file = File("person.json")
val jsonString = Json.encodeToString(a)
file.printWriter().use {
    it.println(jsonString)
}
```

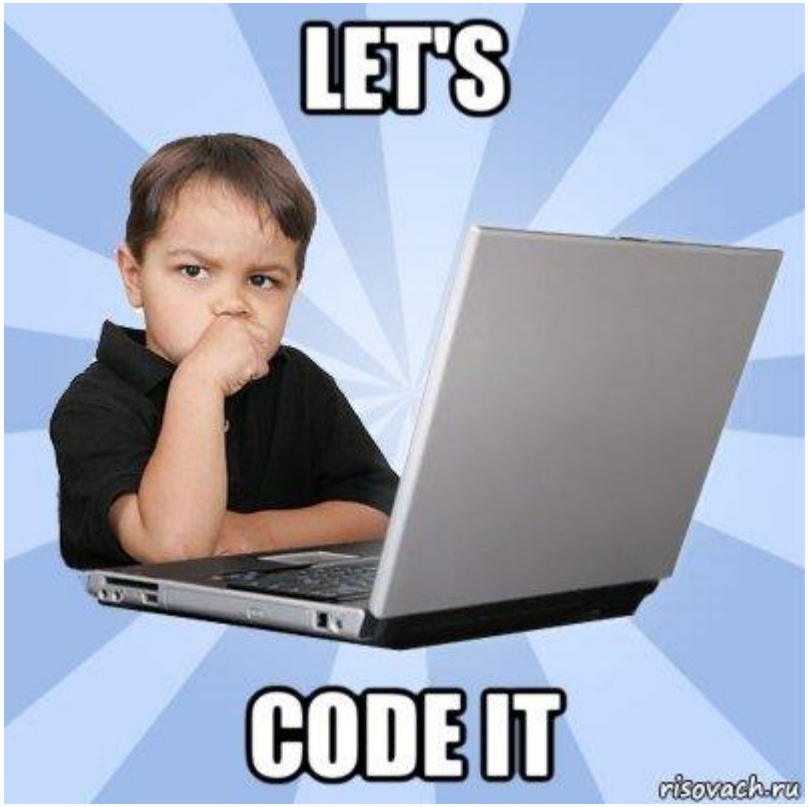
Deserializing data class objects

3. Use the Json.decodeFromString function to deserialize an instance(s) of your data class from JSON

```
val file = File("person.json")
val people = Json.decodeFromString<Array<Person>>(
        file.readText()
)
people.forEach { println(it) }
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Demo

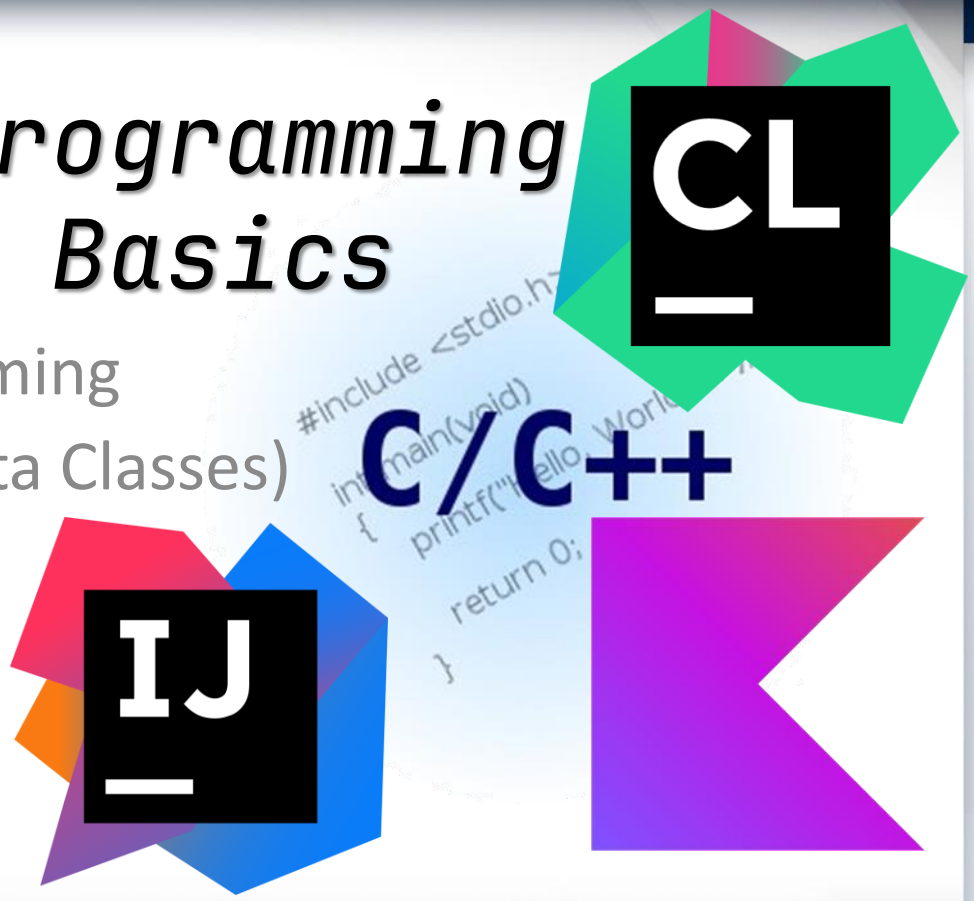


НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Algorithms & Programming
Programming Basics

C/C++/Kotlin programming

(p.6 – Files + Structures / Data Classes)



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