02393 Programming in C++

Module 6: Classes and Objects

Alceste Scalas `<alcsc@dtu.dk>`

5 October 2021

Course plan

Module no.	Date	Topic	Book chapter*
0 and 1	31.08	Welcome & C++ Overview	1
2	07.09	Basic C++ and Data Types	1, 2.2 – 2.5
3	14.09	*LAB DAY*	*C++ Practice*
4	21.09	Data Types	2
		Libraries and Interfaces	3
5	28.09		
6	05.10	Classes and Objects	4.1, 4.2 and 9.1, 9.2
7	12.10	Templates	4.1, 11.1
		Autumn break	
8	26.10	Inheritance	14.3, 14.4, 14.5
9	02.11	**Guest lecture** & *LAB DAY*	*Previous exams*
10	09.11	Recursive Programming	5
11	16.11	Linked Lists	10.5
12	23.11	Trees	13
13	30.11	Conclusion & *LAB DAY*	*Exam preparation*
	05.12	**Exam**	

* Recall that the book uses some ad-hoc libraries (e.g., for strings and vectors). We will use standard libraries

Outline

Recap

Introduction to Object-Oriented Programming in C++

Abstract Data Types

Survey

Lab

A recap of the previous lectures

- **The structure of a C++ program**
  - `#include` and `#define` directives, the `main` function, user-defined functions
- **Simple input/output**
  - `cin`, `cout`
- **Variables, values, and types**
  - `string`, `int`, `double`, `float`, arrays (statically and dynamically allocated), pointers, `enum`, `struct`, `vector`, `ifstream`, `ofstream`
- **Expressions**
  - Some numeric and boolean operators and math functions, conditional expressions
- **Statements**
  - `if`, `while`, `for`, `switch`

The "++" in C++

- So far we have seen **C** programming with few elements of **C++**
  - `string`, `cin`/`cout`, `int &i`, . . .
- **C++** extends C with two key features:
  - **Object-Oriented Programming (OOP)** (today)
  - **templates** for generic programming (next lecture)

Example: safe bank account

Live coding

OOP in C++ in a nutshell

- A `class` is similar to a `struct`, but its members can be both **variables** and **methods**
  - a method is bit like a function

OOP in C++ in a nutshell

- A `class` is similar to a `struct`, but its members can be both **variables** and **methods**
  - a method is bit like a function
- An **object** is an instance of a class

OOP in C++ in a nutshell

- A `class` is similar to a `struct`, but its members can be both **variables** and **methods**
  - a method is bit like a function
- An **object** is an instance of a class
- Class members can be `public` or `private`
  - users of a class can only access `public` members (**data encapsulation**)

OOP in C++ in a nutshell

- A `class` is similar to a `struct`, but its members can be both **variables** and **methods**
  - a method is bit like a function
- An **object** is an instance of a class
- Class members can be `public` or `private`
  - users of a class can only access `public` members (**data encapsulation**)
- Classes can have some **special methods**:
  - **constructor**: called when an object is created
    - either statically, or dynamically using `new`
  - **destructor**: called when an object is destroyed
    - either statically by exiting a scope, or dynamically using `delete`
  - **assignment**: one can customise the behaviour of operator `=`
    - e.g., when the class internally uses dynamic allocation

Abstract Data Types

Use C++ encapsulation to write code that **abstracts from implementation details**

- Specify allowed operations on an ADT, by making them `public`
- Hide everything else, by making it `private`
- Instances of an ADT can only be **constructed** and **used** via `public` operations

Programs that use a well-designed ADT **do not need to be changed** when the ADT's (`private`) implementation details are changed

Live programming examples

Let's implement our own `vector` class

More is available on the **Module 6 materials on DTU Learn**:

- Implementing our own `matrix` class
- Implementing our own `dictionary`/`map` class
- Implementing the `bag` (from previous exercises) in OO style

Your feedback is important!

Please take this **brief anonymous mid-term survey**

(if asked to log in, use your DTU email)

`https://forms.office.com/r/M3P4gYp0XD`

Lab

Today's lab begins now. Tasks:

- make sure C++ works on your computer, request help if it doesn't
- begin working on **Assignment 6**
  - **suggestion**: have a look at the live coding files before starting. . .
- ask questions if something is unclear (including previous assignments)