COMP110: Principles of Computing

Transition to C++ III

Learning outcomes

In this session you will learn how to...

- Define your own **classes** in C++
- Use **pointers**, and allocate objects on the **heap**
- Use **typecasting** to convert values from one type to another
- Use the **CImg** library to write basic GUI applications and image processing algorithms

Object-oriented programming

OOP refresher

- A **class** is a collection of **fields** (data) and **methods** (functions)
- Fields and methods may be **public** (accessible everywhere), **protected** (accessible in the class and classes that inherit from it) or **private** (accessible in the class only)
- Classes may **inherit** fields and methods from other classes
- Subclasses may **override** methods which they inherit — this gives rise to **polymorphism**

Class declarations

```
class MyClass
{
public:
    void doMethod(int x)
    {
        std::cout << x << std::endl;
    }

private:
    int field = 7;
};
```

Fields and methods

- Fields and methods are declared in the class declaration, just like variables and functions
- Class declaration is split into sections by access type (**public**, **protected**, **private**)

Overloading

- Functions and methods can be defined with the **same name** but **different parameters**

```
double getVectorLength(double x, double y)
{
    return sqrt(x * x + y * y);
}

double getVectorLength(Vector v)
{
    return sqrt(v.x * v.x + v.y * v.y);
}
```

FALMOUTH
UNIVERSITY

Constructors and destructors

- The **constructor** is executed when the class is instantiated
- The **destructor** is executed when the instance is freed

```
class MyClass
{
public:
    MyClass()
    {
    }

    ~MyClass()
    {
    }
};
```

Constructors

- The constructor name matches the class name
- Constructors can take parameters
- The constructor can be overloaded, i.e. can have several constructors with different parameters

```
class MyClass
{
public:
    // Parameterless constructor
    MyClass() { }

    // Constructor with parameters
    MyClass(int x, double y) { }
};
```

Destructors

- The destructor name is the class name prefixed with ∼ (tilde)
- Destructors **cannot** take parameters

Modular program design

- Method **declarations** go in the class declaration
- Method **definitions** look like function definitions, with the function name replaced with

  `ClassName::methodName`
- Method definitions can also go inline into the class declaration
  - Best used for short (1 or 2 line) methods
- Good practice: Put class declaration in `ClassName.h`, and method definitions in `ClassName.cpp`

Example: Circle.h

```
#pragma once

class Circle
{
public:
    Circle(double radius);

    double getArea();

private:
    double radius;
};
```

Example: Circle.cpp

```
#include "stdafx.h"
#include "Circle.h"

Circle::Circle(double radius)
    : radius(radius)
{
}

double Circle::getArea()
{
    return M_PI * radius * radius;
}
```

Inheritance

```
class Shape
{
public:
    virtual double getArea();
};

class Circle : Shape
{
public:
    virtual double getArea()
    {
        return M_PI * radius * radius;
    }
};
```

- Methods to be overridden must be marked **virtual**

Pure virtual methods

- **Abstract classes** should never be instantiated — they only exist to serve as a base class
- **Abstract methods** are not defined in the base class, and must be overridden in the subclass
- In C++, abstract methods are called **pure virtual**
- Having at least one pure virtual method **automatically** makes the class abstract

```
class Shape
{
public:
    virtual double getArea() = 0;
};
```

FALMOUTH UNIVERSITY

Instantiation

- To instantiate with a **parameterless constructor**, just declare a variable

```
MyClass myInstance;
```

- To instantiate with a **constructor with parameters**, add the parameters in parentheses

```
MyClass myInstance(27);
```

- This allocates the instance on the **stack**
- The instance is destroyed (and the destructor is called) when the variable goes **out of scope**

FALMOUTH UNIVERSITY

Live coding: Generating Images