COMP110: Principles of Computing

Transition to C++ III

Learning outcomes

In this session you will learn how to...

- Split your program into multiple files, and understand the difference between **source files** and **header files**
- Understand the C++ build pipeline, and the roles of the **preprocessor**, **compiler** and **linker**
- Use arrays, and the difference between creating them on the **stack** versus on the **heap**
- Define C++ functions, and how passing **by reference** differs from passing **by value**

FALMOUTH
UNIVERSITY

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

In Python:

```
class MyClass:
  def __init__(self):
    self.__field = 7

  def doMethod(self, x):
    print x * self. ↩
        __field
```

Class declarations

In Python:

```
class MyClass:
  def __init__(self):
    self.__field = 7

  def doMethod(self, x):
    print x * self. ↩
        __field
```

In C++:

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

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

Fields

- In Python, fields are declared by assigning values to them in the `__init__` method
- In C++, fields (and their types) are declared in the class declaration, just like variables
- Unlike variables, the declaration can't include an initial value — initial values are set in the constructor

Methods

- Methods are defined like functions

Constructors and destructors

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

Modular program design

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

  `ClassName::methodName`
- 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;
}
```

FALMOUTH
UNIVERSITY

Live coding: Generating Images