CSC3221 Practical Classes

Welcome to the practical classes for CSC3221. These classes are built to complement and drive the lectures. The practicals are very important for a programming module. Things which can seem clear during the lectures may just appear not to work at all when in front of a machine. Conversely concepts which seem intractable when described are obvious when seen.

Module Assessment

There are 3 pieces of assessment for this module:

1. Project 1 which is submitted to NESS. (20)
2. Project 2 which is submitted to NESS. (30)
3. An exam in January. (50)

Project 1 (Deadline: 4pm, Friday 8th November, 2019)

Aims

Learn to write full classes in C++ using stack and heap memory with operator overloading and memory management.

Specification

A polynomial (much loved by A Level Maths teachers) has the form $a_n x^n + a_{n-1} x^{n-1} + \dots \_ a_1 x + a_0$

The degree of an polynomial is the highest power of x. For example: $3x^2 + 2x + 4$ has degree 2 and $x^4 - 1$ has degree four (a constant polynomial has degree 0).

Implement a class `Cubic` to store co-efficients of an instance of a polynomial of degree 3. Store your components on the stack. Implement appropriate constructors for your class. Implement member functions and overload operators as appropriate to:

- return the co-efficient of a given power of x.
- compute the value of the cubic given a value for x.
- overload addition and subtraction operators to allow cubics to be manipulated.
- Allow multiplication of a cubic by an `int` or `double`
- Implement == != += and -= operators.
- Overload the input and output operators (>> and <<) appropriately.

Now generalise your solution by writing a class `Polynomial` which can cope with arbitrary polynomials rather than just quadratics. This will require data to be stored on the heap rather than the stack using a C++ array to store co-efficients. Note - no credit will be given for solutions using the STL vector class or variants based on other STL data structures.

Add appropriate constructors, a copy constructor, a destructor and an assignment operator. Implement all the functionality that was required for the Cubic class. In addition, overload the * and *= operators so that polynomials can be multiplied together e.g. $(x+1)*(x-1) = x^2 - 1$.

Test both implementations in a file Test.cpp. Testing may be formal unit tests or informal – the choice is yours. Simply convince a marker that the program works correctly.

Deliverables

A zip file of the relevant project in the projects directory of Visual Studio containing the following classes:

Cubic.h
Cubtic.cpp
Polynomial.h
Polynomial.cpp
Test.cpp

Mark Scheme

Required functionality `Cubic`:	5
Required functionality `Polynomial`:	12
Testing:	3
Total:	20