Practical Task 3.3

(Credit Task)

Submission deadline: 10:00am Monday, April 20
Discussion deadline: 10:00am Saturday, May 2

General Instructions

In mathematics, a ***polynomial*** is an expression that can be built from constants and symbols called indeterminates (or variables) by means of addition, multiplication and exponentiation to a non-negative integer power. A polynomial in a single variable x can always be written in the general form

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$$

where $a_0, \ldots, a_n$ are ***constants*** (***coefficients***) and x is the ***variable***. The word "indeterminate" means that x represents no particular value, although any value may be substituted for it. The mapping that associates the result of this substitution to the substituted value is a function, called a ***polynomial function***. This can be expressed more concisely by using summation notation:

$$\sum_{k=0}^{n} a_k x^k$$

That is, a polynomial can either be zero or can be written as the sum of a finite number of non-zero terms. Each term consists of the product of a number (called the ***coefficient*** of the term) and a finite number of variables, raised to non-negative integer powers. An example of a polynomial of a single indeterminate x is $3x^2 - 4x + 11$.

In this practical task, you need to implement a class called `MyPolynomial`, which models polynomials of degree n. The class must contain an instance variable named `_coeffs`, which stores the coefficients of the n -degree polynomial in a double array of size n +1, where c_0 is kept at index 0.

The class must provide the following public methods to a user:

- **MyPolynomial(double[] coeffs)**

  Constructor. Initializes a new instance of the `MyPolynomial` class that takes as input an array of numbers of type double. This is to initialize the internal array `_coeffs` as well as define the degree n of the polynomial; n equals the length of the given array minus one. The first element of the array corresponds to a_0.

- **int GetDegree()**

  Returns the degree of this polynomial.

- **String ToString()**

  Returns a string that represents the polynomial as a string formatted to the general form "$a_n x\text{^}n + a_{n-1} x\text{^}(n-1) + \cdots + a_2 x\text{^}2 + a_1 x + a_0$", e.g. "$6x\text{^}4 + 8x\text{^}3 + 6x + 1$". Note that if a certain coefficient is zero, then the corresponding term turns out to be zero as well; therefore, the whole term is omitted in the string representation. `ToString()` is the major formatting method in the .NET Framework. It converts an object to its string representation so that it is suitable for display.

- **double Evaluate(double x)**

  Evaluates the polynomial for the given x by substituting the x into the polynomial expression.

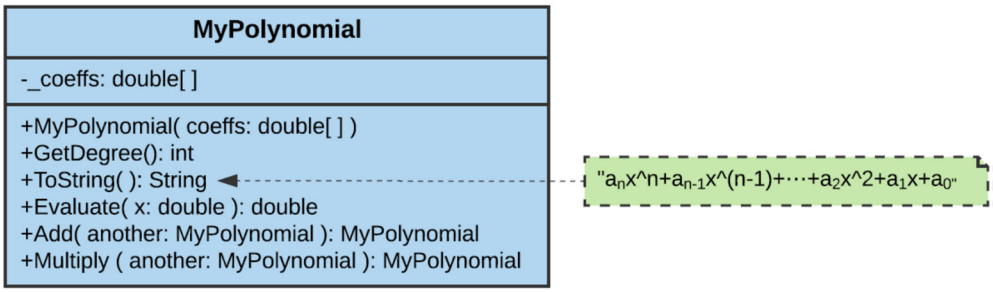
- **MyPolynomial Add(MyPolynomial another)**

  Adds the polynomial `another`, an instance of the `MyPolynomial` class given as input, to this polynomial. Returns this instance that represents the altered polynomial expression.

- **MyPolynomial Multiply (MyPolynomial another)**

  Multiplies the polynomial `another`, an instance of the `MyPolynomial` class given as input, to this polynomial. Returns this instance that represents the altered polynomial expression.

The following UML class diagram should help you to understand the design of the class.



Create a new C# Console Application project and write your code for the `MyPolynomial` class. Also write a test driver, a class called `TestMyPolynomial`, to test all the public methods defined in the `MyPolynomial` class.

Do you need to keep the degree of the polynomial as an instance variable in the `MyPolynomial` class in C#? What is your argument for the answer? How about C/C++?

Further Notes

- Study the way to program and use classes and objects in C# by reading Sections 2.3-2.6 of SIT232 Workbook available in CloudDeakin → Resources.
- Study the concept of a data collection and how to use it in your programs by reading Sections 3.2 of SIT232 Workbook available in CloudDeakin → Resources.
- The following link will give you more insights about arrays and the way to use them:
  - https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/
- Wikipedia will help you to grasp what a polynomial is. Other links will help you with understanding how addition and multiplication work for simple polynomials like one we use for this practical task.
  - https://en.wikipedia.org/wiki/Polynomial
  - https://www.purplemath.com/modules/polyadd.htm
  - https://www.purplemath.com/modules/polymult.htm
- If you seriously struggle with the remaining concepts used in this practical task, we recommend you to start reading the entire Sections 1 and 2 of SIT232 Workbook.
- In this unit, we will use Microsoft Visual Studio 2017 to develop C# programs. Find the instructions to install the community version of Microsoft Visual Studio 2017 available on the SIT232 unit web-page in CloudDeakin in Resources → Software → Visual Studio Community 2017. You however are free to use another IDE, e.g. Visual Studio Code, if you prefer that.

Marking Process and Discussion

To get your task completed, you must finish the following steps strictly on time.

- Make sure that your program implements the required functionality. It must compile and have no runtime errors. Programs causing compilation or runtime errors will not be accepted as a solution. You need to test your program thoroughly before submission. Think about potential errors where your program might fail.

- Submit the expected code files as an answer to the task via OnTrack submission system. Cloud students must record a short video explaining their work and solution to the task. Upload the video to one of accessible resources, and refer to it for the purpose of marking. You must provide a working link to the video to your marking tutor in OnTrack.
- On-campus students must meet with their marking tutor to demonstrate and discuss their solution in one of the dedicated practical sessions. Be on time with respect to the specified discussion deadline.
- Answer all additional questions that your tutor may ask you. Questions are likely to cover lecture notes, so attending (or watching) lectures should help you with this **compulsory** interview part. Please, come prepared so that the class time is used efficiently and fairly for all students in it. You should start your interview as soon as possible as if your answers are wrong, you may have to pass another interview, still before the deadline. Use available attempts properly.

Note that we will not accept your solution after the submission deadline and will not discuss it after the discussion deadline. If you fail the deadline, you also fail the task and this may impact your performance and your final grade in the unit. Unless extended for all students, the deadlines are strict to guarantee smooth and on-time work throughout the unit.

Remember that this is your responsibility to keep track of your progress in the unit that includes checking which tasks have been marked as completed in the OnTrack system by your marking tutor, and which are still to be finalised. When marking you at the end of the unit, we will solely rely on the records of the OnTrack system and feedback provided by your tutor about your overall progress and quality of your solutions.