Computer Science II: Data Structures: Programming Assignment 2

Alexander Starr – 00567613

July 15, 2013

Problem Description

The objective of this assignment is to represent polynomials in the C++ language. To do this, we must implement linked lists, with each node representing a term in the polynomial.

Linked lists are a data structure used to represent lists in a computer system. They are different from arrays in that they use pointers to indicate the locations of elements in the list. Each node has two portions – a data field and a pointer field. The data field contains the element, and the pointer field indicates the memory location of the next node. This enables them to have no memory waste, and they do not require allocation of a large memory block. However, it means that to find an element in the list, you must scan through each element, as the memory locations of the elements are not strictly defined.

The second portion of the assignment is to add two polynomials together and display the result. The result should be reduced, so that any terms of the same power are combined and not displayed separately. Also, while not necessary, it is preferable to display the polynomials in descending order.

Finally, the input can be in any preferred way. The program can read from a file, take user input, or only compute hardcoded polynomials.

Source Code and Test Results

Please see `main.cpp` for the source code of my solution.
The program accepts user input for the polynomial addition. It will prompt for a coefficient, then an exponent, and then ask if the user would like to add another term. This continues until the user does not want to add any more terms, and then it repeats for the second polynomial.
The following are examples of input and output of the program:

```
First Polynomial
Enter a coefficient: 0.75
Enter an exponent: 2
Add another term? [y/n] n

Second Polynomial
Enter a coefficient: 0.1
Enter an exponent: 2
Add another term? [y/n] y
Enter a coefficient: 0.9
Enter an exponent: 1
Add another term? [y/n] n

  0.75x^2
+ 0.1x^2 + 0.9x^1
----------------------------------------
  0.85x^2 + 0.9x^1



First Polynomial
Enter a coefficient: 1
Enter an exponent: 0
Add another term? [y/n] y
Enter a coefficient: 2
Enter an exponent: 1
Add another term? [y/n] y
Enter a coefficient: 3
Enter an exponent: 2
Add another term? [y/n] n

Second Polynomial
Enter a coefficient: 5
Enter an exponent: 5
Add another term? [y/n] y
Enter a coefficient: 4
Enter an exponent: 3
Add another term? [y/n] y
Enter a coefficient: 2
Enter an exponent: 2
Add another term? [y/n] n
```

Source Code and Test Results continued on next page...

```
  3x^2 + 2x^1 + 1x^0
+ 5x^5 + 4x^3 + 2x^2
---------------------------------------
  5x^5 + 4x^3 + 5x^2 + 2x^1 + 1x^0



First Polynomial
Enter a coefficient: 100
Enter an exponent: 100
Add another term? [y/n] y
Enter a coefficient: 1
Enter an exponent: 2
Add another term? [y/n] n

Second Polynomial
Enter a coefficient: 0.5
Enter an exponent: 100
Add another term? [y/n] y
Enter a coefficient: 0.5
Enter an exponent: 2
Add another term? [y/n] y
Enter a coefficient: 0.5
Enter an exponent: 1
Add another term? [y/n] n

  100x^100 + 1x^2
+ 0.5x^100 + 0.5x^2 + 0.5x^1
---------------------------------------
  100.5x^100 + 1.5x^2 + 0.5x^1
```

Note that the program can handle float coefficients, but only integer exponents. Also note that the polynomials are displayed in descending order by power.

Comments

The algorithm implemented to order the polynomials by their power was bubble sort. This is a very inefficient algorithm at large values of n, because it has a time complexity $O(n^2)$. However, it is very simple to implement, and the inefficiency should be negligible at the values of n expected in the program.