Bahria University, Islamabad

Department of Software Engineering

Data Structures & Algorithms Lab

(Spring-2024)

Teacher: RAHEELA AMBRIN

Student : Abdul Rafay

Enrollment : 01-131232-004

Lab Journal: 7

Date: 23 / 10 / 24

Comments:

Signature

Code:

All the code files are uploaded on GitHub: https://github.com/CharlieFour/DSA_Lab

You can check out the code on GitHub in Open_Ended folder.

List.h

```
#pragma once
typedef struct Node* Nodeptr;

struct Node{
   int exponent;
   int number;
   Nodeptr next;
   Nodeptr prev;
};

class List
{
   public:
      Nodeptr list;
      int length;
      List();
      void iAE(int x, int y);
      void clear();
};
```

List.cpp

```
#include "list.h"
#include <iostream>
#include <fstream>

List::List(){
    list = nullptr;
    length = 0;
}

void List::iAE(int x, int y){
    if(list == nullptr){
        Nodeptr p = new Node;
        p->number = x;

        p->exponent = y;
        p->prev = list;
        p->next = nullptr;
```

```
    list = p;

    length++;
    return;
  }
  Nodeptr p = list;
  while(p->prev != nullptr){
    p = p->prev;
  }
  Nodeptr q = new Node;
  p->number = x;
  p->exponent = y;
  q->prev = nullptr;
  q->next = p;
  p->prev = q;

  length++;
}

void List::clear(){
  Nodeptr p = list->prev;
  while(p->prev != nullptr){
    Nodeptr q = p->next;
    delete p;
    p = q;
```

Polynomial.h

```
#pragma once

#include <iostream>
#include <fstream>
#include "list.h"

class Polynomial
{
    public:
        Polynomial();
        Polynomial(int d);
        List readPolynomial(std::string fileName);
        List addPolynomial(List p1, List p2);
        List multiplyPolynomial(List p1, List p2);
        void evaluatePolynomial(int x, List p);
};
```

Polynomial.cpp

```cpp
#include "polynomial.h"
#include "list.h"
#include <iostream>
#include <sstream>
#include <string>
#include <cctype>
#include <cmath>

Polynomial::Polynomial(){
}

Polynomial::Polynomial(int d){
}

List Polynomial::readPolynomial(std::string fileName)
{
    List list;
    std::ifstream file(fileName);
    int x = 0, y = 0;  // x is the coefficient, y is the exponent
    std::string line;
    if (getline(file, line)) {
        int i = 0;
        while (i < line.size()) {
            if (isdigit(line[i]) || (line[i] == '-' &&
isdigit(line[i + 1]))) {
                // Read the coefficient (may be multi-digit or
negative)
                std::string coeffStr;
                if (line[i] == '-') {
                    coeffStr += '-';
                    i++;
                }
                while (isdigit(line[i])) {
                    coeffStr += line[i++];
                }
                x = std::stoi(coeffStr);

                // Check if exponent follows
                if (line[i] == 'x') {
                    i++;  // Skip 'x'
                    if (i < line.size() && line[i] == '^') {
                        i++;  // Skip '^'

                        std::string expStr;
                        while (isdigit(line[i])) {
                            expStr += line[i++];
                        }
                        y = std::stoi(expStr);
                    }
```

```
 else {
                        y = 1;  // Implicit exponent 1 for terms
like '2x'
                    }
                } else {
                    y = 0;  // Constant term
                }

                // Add the term to the list
                list.iAE(x, y);
            }
            i++;
        }
    }
    file.close();
    return list;
}

List Polynomial::addPolynomial(List p1, List p2)
{
    Nodeptr current1 = p1.list;
    Nodeptr current2 = p2.list;
    List result;
    while(current1 != nullptr)
    {
        while(current2 != nullptr)
        {
            if(current1->exponent == current2->exponent)
            {
                result.iAE(current1->number + current2->number,
current1->exponent);
            }
            current2 = current2->prev;
        }
        current1 = current1->prev;
    }
    return result;
}

List Polynomial::multiplyPolynomial(List p1, List p2)
{
    Nodeptr current1 = p1.list;
    Nodeptr current2 = p2.list;
    List result;
    while(current1 != nullptr)
    {
        while(current2 != nullptr)
        {
            result.iAE(current1->number * current2->number,
current1->exponent + current2->exponent);
```

```
            current2 = current2->prev;
        }
        current1 = current1->prev;
    }
    return result;
}

void Polynomial::evaluatePolynomial(int x, List p)
{
    Nodeptr current = p.list;
    int result = 0;
    while(current != nullptr)
    {
        result = result + pow((current->number * x), current-
>exponent);
    }
    std::cout << "The value of the polynomial at x = " << x << "
is: " << result << std::endl;
}
```

main.cpp

```
#include <iostream>
#include "../libraries/polynomial.h"
#include "../libraries/list.h"
#include <limits>

using namespace std;

void menu();
int getInput();

main()
{
    List p1, p2;
    Polynomial poly;
    List result;
    p1 = poly.readPolynomial("ptest1");
    p2 = poly.readPolynomial("ptest2");
    menu();
    int input = getInput();

    while(input != 4)
    {
        switch(input)
        {
            case 1:
                result = poly.addPolynomial(p1, p2);
                while(result.list != nullptr)
                {
                    cout << result.list->number << "x^" <<
result.list->exponent << " ";
```

```
            case 2:
                result = poly.multiplyPolynomial(p1, p2);
                cout << result.list->number << "x^" <<
result.list->exponent << " ";
                break;
            case 3:
                int a , b;
                a = getInput();
                poly.evaluatePolynomial(a, p1);
            case 4:
                cout << "Exiting..." << endl;
                break;
            default:
                cout << "Invalid choice. Please try again." <<
endl;
        }
        menu();
        input = getInput();
    }
    system("pause");
    return 0;
}

void menu()
{
    cout << "1. Addition" << endl;
    cout << "3. Multiplication" << endl;
    cout << "3. Evaluate" << endl;
    cout << "4. Exit" << endl;
}

int getInput()
{
    int x;
    cout << "Enter the input: ";
    if(cin>>x)
    {
        return x;
    }
    else
    {
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cerr << "Invalid input. Please enter a number." << endl;
        return getInput();
    }
}
```