Министерство образования науки РФ

Нижегородский государственный университет имени Н.И. Лобачевского

( Национальный исследовательский университет )

Институт Информационных Технологий Математики и Механики

Отчёт по лабораторной работе №4

Тема: « Калькулятор полиномов »



Выполнил:

студент группы 8103

Латанов Владислав Александрович

Проверил:

Козинов Е.А.

Нижний Новгород 2016 год.

# Содержание.

1. [Введение……………………………………………………………………………..3](#_Введение.)
2. [Постановка учебно-практической задачи………………………………………....4](#_Постановка_учебно-практической_зада)
3. [Руководство пользователя………………………………………………………….5](#_Руководство_пользователя.)
4. [Руководство программиста………………………………………………………....8](#_Руководство_программиста.)
5. [Заключение………………………………………………………………………….14](#_Заключение.)
6. [Список литературы………………………………………………………………....15](#_Список_литературы.)
7. [Приложение………………………………………………………………………....16](#_Приложение.)

# Введение.

Сложные математические модели всегда интересовали профессиональных математиков. В начале двадцатого века центральной проблемой алгебры была проблема алгебры полиномов, а точнее оптимизация операций над ними и вычисление их значений. К этому времени алгебра полиномов уже имела значительный «багаж» теоретических знаний. Первооткрывателями в этой сфере были: Карл Фридрих Гаусс и его основная теорема алгебры; Эйлер, Декарт, все они внесли непомерный вклад в развитие алгебры полиномов и алгебры в целом. На сегодняшний день полиномы как формальный объект хорошо изучены в математике. Фактически математическая модель формируют *алгебру полиномов*. Сегодня вычисление и привидение полиномов к каноническому виду очень важно для корректной работы многих алгоритмов, определяющих работу станков, военной и гражданской техники. Поэтому для корректного вычисления и применения операций над полиномами необходимы компьютерные технологии. Это значительно увеличит скорость обработки полиномов, что сохраняет наиважнейший ресурс процесса – время. Плюс ко всему не будем забывать, что компьютер точнее в вычислениях, чем человек. И так, важность такой математической структуры как полиномы, неоспорима. Теперь необходимо переложить всю сложность операций над полиномами на машину, а именно на компьютер. Для этого необходимо выбрать структуру хранения для данной алгебраической структуры. Воспользуемся уже частично известной нам структурой – линейным списком.

# Постановка учебно-практической задачи.

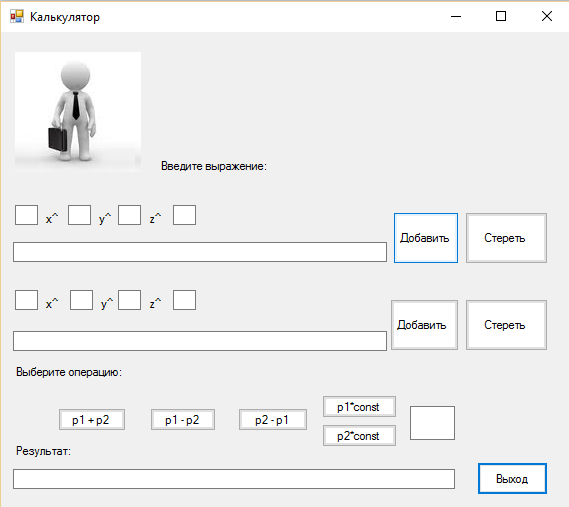
В данной лабораторной работе, необходимо реализовать класс Monom и Polinom.

* Разработанная мной программа, должна обеспечивать:

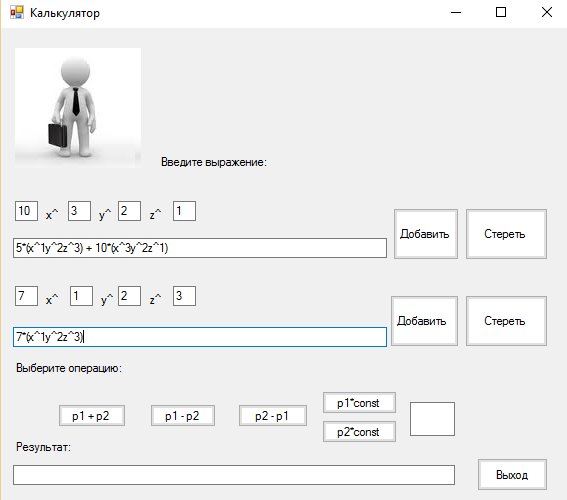
1. Ввод двух полиномов со степенями от 0 до 19;
2. Реализовать операции: сложение 2-х полиномов, вычитание, умножение на константу обоих полиномов;
3. Вывод результата;
4. Реализовать 20 тестов, для корректной работы программы;
5. Реализовать визуальный интерфейс Windows Form;

# Руководство пользователя.

При запуске программы, должно появиться визуальное приложение, вида:



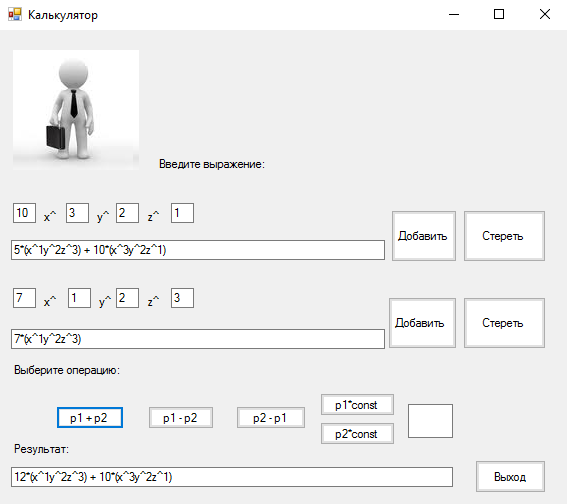
Затем мы должны ввести в поля первый и второй полином и нажать кнопку «Добавить»:



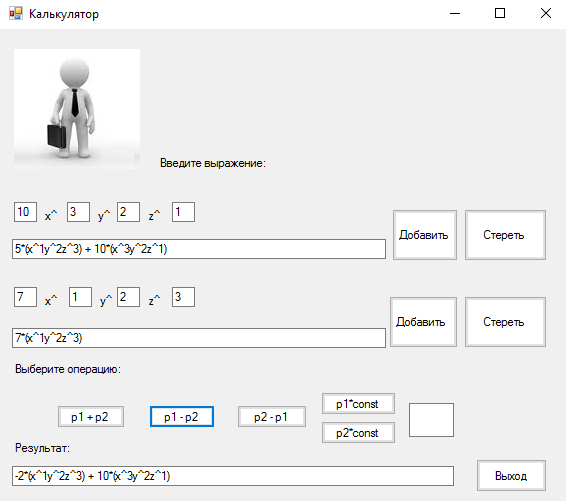
Далее, используя соответствующие кнопки сложения, вычитания, умножение на число, мы можем рассчитать результат операций, и вывести его в поле «Результат».

И так, проверим работоспособность каждой кнопки:

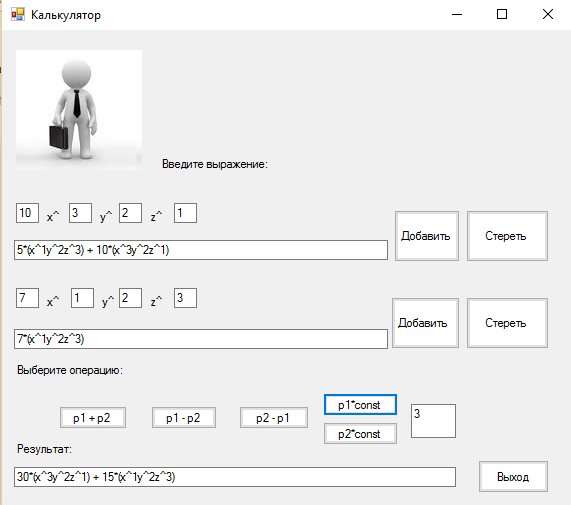
1. Сумма:



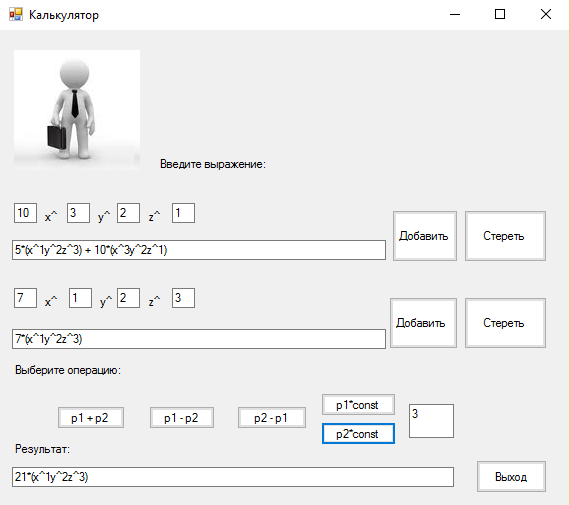
1. Вычитание:



1. Умножение первого полинома на константу:



1. Умножение второго полинома на константу:



Для того, чтобы ввести новый полином, необходимо нажать кнопку «Стереть». Чтобы завершить работу приложения, необходимо нажать кнопку «Завершить». Ну вот и все. Все достаточно очевидно и просто и готово к использованию.

# Руководство программиста.

При выполнении работы можно использовать следующее понимание полинома:

Полином состоит из мономов. Каждый моном характеризуется коэффициентом KOF и степенями переменных A, B, C: KOF\*x^Ay^Bz^C. Величину степеней переменных можно ограничить значением 19. При манипуляции полиномами, эффективной структурой хранения являются списки. При этом каждое звено списка хранит моном с отличным от нуля коэффициентом.

Структура хранения полиномов должна обеспечивать эффективное выполнение операций. Если мономы в списке упорядочить по степеням переменных, то можно предложить эффективный алгоритм сложения полиномов, основанный на идее алгоритма слияния двух упорядоченных массивов.

В результате операций (например, сложения) может быть получен полином, у которого нет отличных от нуля коэффициентов. Структура хранения такого нулевого полинома не должна отличаться от структуры хранения прочих полиномов.

Для эффективного использования оперативной памяти реализуем операцию сложения таким образом, чтобы результат операции был получен на месте одного из слагаемых. При этом неизбежны вставки и удаления звеньев в списке.

Таким образом, с учетом вышеизложенного, рекомендуется следующая структура хранения полиномов. Полиномы упорядочиваются по убыванию степеней мономов. Для определения старшинства мономов вводится следующее правило. Во-первых, фиксируется старшинство переменных. Будем считать, что x - самая старшая переменная, затем следует y, затем z. Для каждого монома определим его "свернутую степень" (индекс). Для монома x^Ay^Bz^C. индекс равен A\*102+B\*101+C\*100 (по условию задачи A, B и C не выше 19). Старшим считается моном с большей свернутой степенью. Например, x^3y старше xy^7z^6, так как 310 больше 176.

Например имеется 2 полинома: Р(z,y,z)= 5\*(x^1y^2z^3) + 10\*(x^3y^2z^1)

и Q(x,y,z)= 7\*(x^1y^2z^3), результатом будет:

1. Сложение: 12\*(x^1y^2z^3) + 10\*(x^3y^2z^1);
2. Вычитание из первого полинома второй: -2\*(x^1y^2z^3) + 10\*(x^3y^2z^1);
3. Вычитание из второго полинома первый: -10\*(x^3y^2z^1) + 2\*(x^1y^2z^3);
4. Умножение на константу(3) первого полинома: 30\*(x^3y^2z^1) + 15\*(x^1y^2z^3);
5. Умножение на константу(3) второго полинома: 21\*(x^1y^2z^3);

**Описание алгоритмов.**

**Класс Monom.h имеет следующие методы:**

1. Monom(int \_Coefficient=0,int Svertka=0, Monom \*\_Next=NULL);

Это инициализирующий конструктор. Предназначен для того, чтобы инициализировать поля класса;

1. Monom(const Monom &Tmp);

Конструктор копирования. Его работа заключается в том, чтобы создать реальную копию объекта со своей личной выделенной динамической памятью;

1. Monom& operator=( Monom &Tmp);

Присваивание монома(присваивает коэффициент и степени одного монома к другому моному);

1. void SetCoefficient(int a);

Установка коэффициента монома;

1. void SetSvertka(int Tmp);

Установка свертки (степеней монома);

1. void SetNext(Monom \*Tmp);

Установка указателя на моном;

1. int GetCoefficient();

Получение коэффициента монома;

1. int GetSvertka();

Получение свертки(степени) монома;

1. Monom\* GetNext();

Получение указателя на моном;

**Класс Polinom.h имеет следующие методы:**

1. Polinom();

Это инициализирующий конструктор. Предназначен для того, чтобы инициализировать поля класса;

1. ~Polinom();

Деструктор. Выделяется определенный объем памяти, а деструктор класса эту память освобождает;

1. Polinom(const Polinom &p);

Конструктор копирования. Его работа заключается в том, чтобы создать реальную копию объекта со своей личной выделенной динамической памятью;

1. void Add(int a, int N);

Функция добавления монома в полином: функция просматривает, есть ли среди мономов в полиноме подобный к вводимому моному(с одинаковыми коэффициентами). Если есть, то эти 2 монома суммируются. Если таких мономов нет, то вводимый моном устанавливается в голову полинома;

1. Polinom& operator\*(int a);

Умножение на константу. Проходим по списку, перемножая коэффициент у каждого монома на входное значение константы. Если константа равна нулю, то возвращаем нулевой моном;

1. Polinom& operator+(Polinom &p);

Сложение полиномов. Проходим по списку мономов первого полинома. Добавляем каждый моном второго полинома с помощью уже реализованной функции добавления монома;

1. Polinom& operator-(Polinom &p);

Вычитание полиномов. Проходим по списку мономов первого полинома. Добавляем каждый моном второго полинома с помощью уже реализованной функции добавления монома;

1. Polinom& operator=(Polinom &p);

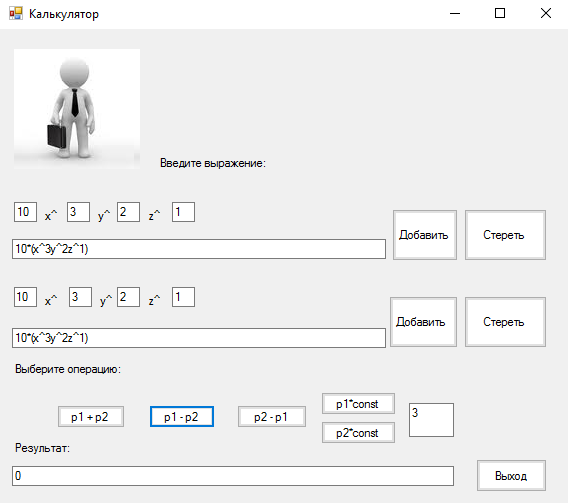
Функция присваивания. Поочередно присваивает мономы одного полинома с мономом другого полинома.

1. Monom\* GetFirst();

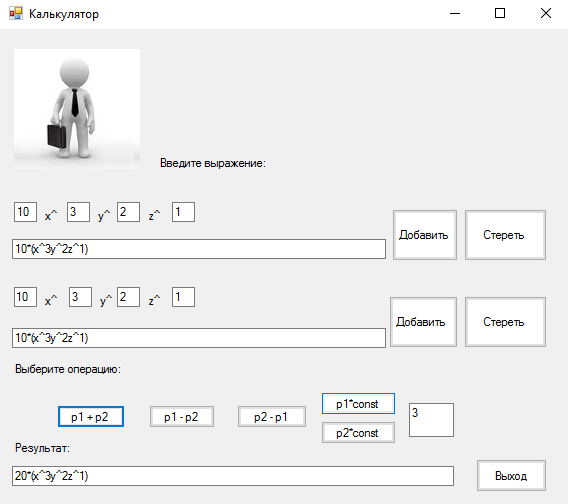
Получение головы списка.

Приведем пару примеров:

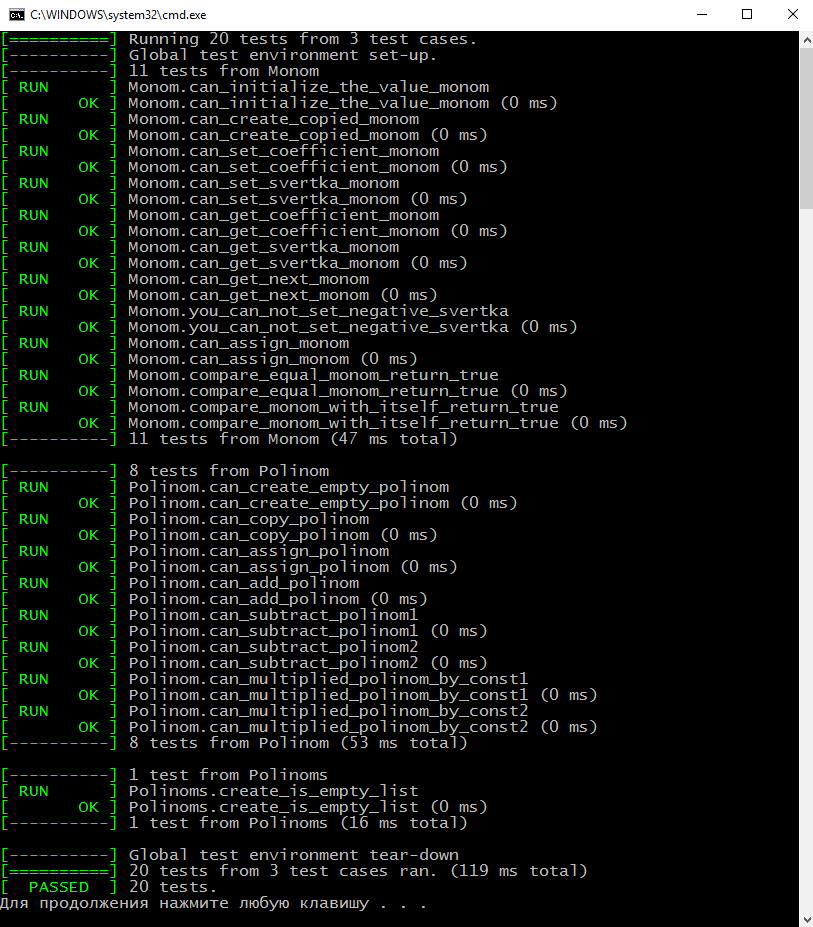
Операция вычитания:



Операция сложения:



Так же я говорил, что в программе присутствует набор тестов, их так же можно увидеть на снимке и убедиться, что они работают корректно.



**Описание структур программного комплекса.**

1. Monom.h – заголовочный файл класса Monom;
2. Monom.cpp – файл с реализацией класса Monom;
3. Polinom.h – заголовочный файл класса Polinom;
4. Polinom.cpp – файл с реализацией класса Polinom;
5. Sample.cpp – консольное приложение;
6. Test\_monom.cpp, Test\_polinom.cpp – реализация тестов;
7. MyForm.h , MyForm.cpp – реализация визульного приложения;

# Схема наследования.

|  |
| --- |
| **class Monom** |
| * Coefficient: int |
| * Svertka: int |
| * Next: Monom\* |
| + Monom(\_Coefficient: int, \_Svertka: int, \_Next: Monom\*); |
| + Monom(Tmp: const Monom&); |
| + SetCoeffircient(a: int): void; |
| + SetIndex(Tmp: int): void; |
| + SetIndex(Tmp: Monom\*)^ void; |
| + GetCoefficient(): int; |
| + GetSvertka(): int; |
| + GetNext(): Monom \*; |
| + operator = (Tmp: const Monom &): Monom&; |

|  |
| --- |
| **class Polinom** |
| * First: Monom\* |
| +Polinom(); |
| + ~Polinom(); |
| +Polinom(p:const Polinom&): void; |
| +Add(a: int , N: int): void; |
| + operator \*(a: int:Polinom&); |
| + operator +(p: const Polinom&): Polinom&; |

|  |
| --- |
| + operator -(p: const Polinom&): Polinom&; |
| + operator =(p: const Polinom&): Polinom&; |
| + GetFirst(): Monom\* |

# Заключение.

Нам удалось освоить и представить в нашем проекте новые для нас принципы построения структур хранения данных в несвязных между собой участках памяти компьютера.

Научились использовать указатели в представлении отношений между данными.

Познакомились с теорией циклических списковых структур хранения данных.

Получили практический навык использования системы динамически-распределяемой памяти.

Получили навык работы с новой версией компилятора winforms и соответственно с основными атрибутами данной среды.

Наши теоретические представления о результатах работы приложения оказались верны и результаты примера совпали нашими расчетами.

# Список литературы.

1. Гергель В.П. «Рабочие материалы к учебному курсу «Методы программирования», ННГУ, 2002. – 100 с.
2. <http://algolist.manual.ru/maths/misc/revpn.php>

# Приложение.

**Monom.h**

#pragma once

#include <iostream>

using namespace std;

class Monom

{

int Coefficient;

int Svertka;

public:

Monom \*Next;

Monom(int \_Coefficient=0,int Svertka=0, Monom \*\_Next=NULL);

Monom(const Monom &Tmp);

Monom& operator=( Monom &Tmp);

void SetCoefficient(int a);

void SetSvertka(int Tmp);

void SetNext(Monom \*Tmp);

int GetCoefficient();

int GetSvertka();

Monom\* GetNext();

};

**Polinom.h**

#pragma once

#include "monom.h"

class Polinom

{

Monom \*first;

public:

Polinom();

~Polinom();

Polinom(const Polinom &p);

void Add(int a, int N);

Polinom& operator\*(int a);

Polinom& operator+(Polinom &p);

Polinom& operator-(Polinom &p);

Polinom& operator=(Polinom &p);

Monom\* GetFirst()

{

return first;

}

};

**Monom.cpp**

#include "monom.h"

Monom::Monom(int \_Coefficient, int \_Svertka, Monom \*\_Next)

{

Coefficient = \_Coefficient;

Svertka = \_Svertka;

Next = \_Next;

}

Monom::Monom(const Monom &Tmp)

{

Coefficient = Tmp.Coefficient;

Svertka = Tmp.Svertka;

Next = Tmp.Next;

}

Monom& Monom::operator=(Monom &Tmp)

{

Coefficient = Tmp.Coefficient;

Svertka = Tmp.Svertka;

return \*this;

}

void Monom::SetCoefficient(int a)

{

Coefficient = a;

}

void Monom::SetSvertka(int Tmp)

{

Svertka = Tmp;

}

void Monom::SetNext(Monom \*Tmp)

{

Next=Tmp;

}

int Monom::GetCoefficient()

{

return Coefficient;

}

int Monom::GetSvertka()

{

return Svertka;

}

Monom\* Monom::GetNext()

{

return Next;

}

**Polinom.cpp**

#include "polinom.h"

Polinom::Polinom()

{

first = NULL;

}

Polinom::~Polinom()

{

}

Polinom::Polinom(const Polinom &p)

{

first = new Monom;

first->SetNext(NULL);

Monom \*cur;

cur = p.first;

while (cur != NULL)

{

Add(cur->GetCoefficient(), cur->GetSvertka());

cur = cur->GetNext();

}

}

void Polinom::Add(int a, int N)

{

bool flag = false;

if (a == 0)

{

return;

}

Monom \*t = first;

while (t != NULL)

{

if (t->GetSvertka() == N)

{

flag = true;

break;

}

t = t->GetNext();

}

if (flag == false)

{

Monom \*p = new Monom(a, N, first);

first = p;

}

if (flag == true)

{

if ((a + t->GetCoefficient()) != 0)

{

t->SetCoefficient(a + t->GetCoefficient());

}

else

{

Monom \*tmp = first;

if (t == first)

{

first = t->Next;

}

else

{

while (tmp->Next != t)

{

tmp = tmp->Next;

}

tmp->Next = t->Next;

}

delete t;

}

}

}

Polinom& Polinom:: operator\*(int a)

{

Polinom \*res = new Polinom;

Polinom q(\*this);

Monom \*t = q.GetFirst();

if (a == 0.0)

{

return \*res;

}

while (t != NULL)

{

(\*res).Add(a\*(t->GetCoefficient()), t->GetSvertka());

t = t->GetNext();

}

return \*res;

}

Polinom& Polinom:: operator+(Polinom &p)

{

Polinom \*res = new Polinom(\*this);

Monom \*t = p.first;

while (t != NULL)

{

(\*res).Add(t->GetCoefficient(), t->GetSvertka());

t = t->GetNext();

}

return \*res;

}

Polinom& Polinom:: operator-(Polinom &p)

{

Polinom \*res = new Polinom(\*this);

Monom \*t = p.first;

while (t != NULL)

{

(\*res).Add(-t->GetCoefficient(), t->GetSvertka());

t = t->GetNext();

}

return \*res;

}

Polinom& Polinom:: operator=(Polinom &p)

{

first = p.first;

Monom \*l1 = p.first;

Monom \*l2 = (\*this).GetFirst();

l2 = l1;

while (l1 != NULL)

{

l2 = l1;

l1 = l1->GetNext();

l2 = l2->GetNext();

}

return \*this;

}

**sample.cpp**

#include "polinom.h"

int main()

{

int flag = 0;

cout << "Input polinom p1:" << endl;

Polinom p1;

cin >> p1;

cout << "Input polinom p2:" << endl;

Polinom p2;

cin >> p2;

system("cls");

cout << "Resiltat p1: " << endl << p1 << endl;

cout << "Resiltat p2: " << endl << p2 << endl << endl;

cout << "Create operation:" << endl;

cout << "1. p1 + p2" << endl;

cout << "2. p1 - p2" << endl;

cout << "3. p2 - p1" << endl;

cout << "4. p1 \* const" << endl;

cout << "5. p2 \* const" << endl;

cout << "0. Exit" << endl << endl;

cin >> flag;

while (flag != 0)

{

switch (flag)

{

case 1:

{

cout << "Resultat: " << p1 + p2 << endl;

break;

}

case 2:

{

cout << "Resultat: " << p1 - p2 << endl;

break;

}

case 3:

{

cout << "Resultat: " << p2 - p1 << endl;

break;

}

case 4:

{

int a = 0;

cout << "Input const:" << endl;

cin >> a;

cout << "Resultat: " << p1\*a << endl;

break;

}

case 5:

{

int a = 0;

cout << "Input const:" << endl;

cin >> a;

cout << "Resultat: " << p1\*a << endl;

break;

}

default:

{

cout << "Error!" << endl;

break;

}

}

cout << "Select the action:" << endl;

cin >> flag;

}

return 0;

}