Министерство образования и науки РФ
Нижегородский государственный университет имени Н.И. Лобачевского (Национальный исследовательский университет)
Институт Информационных Технологий Математики и Механики

Отчет по лабораторной работе № 4

Арифметические операции с полиномами

Выполнил:
студент группы 0823-3
Краснов Александр Александрович
Проверил: Козинов Е.А.

г. Нижний Новгород

2016г.

Содержание

Введение

В данной лабораторной работе рассмотрен вопрос об арифметических выражениях с полиномами (или многочленами). Рассматриваемый пример организации обработки полиномов может рассматриваться как введение в проблематику аналитических вычислений на ЭВМ.

В процессе разработки использовалась такая динамическая структура данных, как линейный односвязный список, для хранения мономов в качестве звеньев. Напомню, что линейный односвязный список — абстрактный тип данных, представляющий собой список элементов, связанных посредством указателя. При этом последний элемент списка содержит указатель на NULL.

При реализации данного проекта применяется язык C++ (с применением классов).

Постановка задачи

Полиномы как формальный объект хорошо изучены в математике. Математическая модель – алгебра полиномов.

Под многочленом понимается выражение из нескольких термов, соединённых знаками сложения или вычитания.

Терм включает коэффициент и моном, содержащий одну или несколько переменных, каждая из которых может иметь степень.

В число возможных операций можно включить сложение, вычитание, умножение полиномов, а также умножение полинома на константу.

Одна из наиболее частых операций – приведение подобных мономов.

Для ускорения поиска подобных элементов целесообразно ввести правило упорядочения мономов. Возможный вариант – сортировать мономы по индексам – “сверткам” степеней полинома.

В ходе вычислений количество мономов в полиноме будет меняться. Следовательно, полином стоит представлять динамической структурой, например, линейным списком.

Таким образом, необходимо разработать программу, выполняющую арифметические операции с полиномами трех переменных: сложение, вычитание, умножение на константу, умножение двух полиномов. Считается, что полином составлен из мономов от трех переменных со степенью от 0 до 19. Коэффициенты полинома - вещественные числа. Работоспособность программы необходимо проверить с помощью Google Test-ов. Кроме того, необходимо разработать пользовательское консольное приложение и графический интерфейс.

Руководство пользователя

Данная программа предлагает следующий набор возможностей:

1) Ввод двух полиномов со степенями от 0 до 19 включительно
2) Проверка на корректность ввода степеней
3) Реализация операций: умножение полинома на константу, сложение, вычитание, умножение полиномов
4) Вывод результата вычислений.

Руководство программиста

Особенности реализации:

1) В качестве структуры хранения полинома используется линейный односвязный список. Мономы представляются как звенья списка, а полиномы как сам список.
2) Степень полинома хранится в "свернутом" виде: N = 20^2 ’x’ + 20 ‘y’ + ‘z’, где ‘x’, ‘y’, ‘z’ – степени при x, y, z соответственно. Мономы хранятся упорядоченно по данному числу.
3) Полином не содержит мономов с нулевыми коэффициентами. Если в результате ввода или выполнения операций образовались мономы с нулевыми коэффициентами, они сразу удаляются.
4) Если пользователь вводит полином со степенями больше 19, выводится сообщение об ошибке.
5) Следует учесть, что пользователь может вводить полином, не упорядочив в нем мономы.
6) Умножение полинома на константу осуществляется как умножение коэффициентов каждого монома на вводимую константу.
7) Сложение полиномов осуществляется алгоритмом слияния упорядоченных массивов.
8) Вычитание выполняется аналогично сложению.
9) При умножении отслеживается приведение подобных слагаемых.
10) Если при умножении полиномов полученные степени переменных больше 19, выводится сообщение об ошибке.
11) Пользователь может вводить полином, как в консольном приложении, так и в графическом.

Описание алгоритмов:

1) *Добавление монома*

   Проходим циклом по списку, проверяя, есть ли в нем мономы с такими же индексами. Если нет, то добавляем звено в конец, иначе добавляем моном к существующему моному с одинаковым индексом путем сложения коэффициентов. Если коэффициент больше коэффициента первого элемента, вставляем новое звено в начало. Если коэффициент больше коэффициента текущего звена (не начального), выходим из цикла. Если коэффициент нулевой, то никаких действий не выполняем. Если сложение коэффициентов мономов с одинаковыми индексами дало 0, то удаляем текущий моном из списка. Если полином нулевой, то просто вставляем в него новое звено. Завершив цикл, проверяем, если выход был досрочный, то вставляем новое звено в список после последнего из звеньев, коэффициент которого был меньше добавляемого, иначе – вставляем в конец. Тем самым, мы получаем отсортированный полином по убыванию индексов.

2) *Умножение полинома на константу*

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

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

4) *Вычитание полиномов*
Аналогично сложению полиномов.

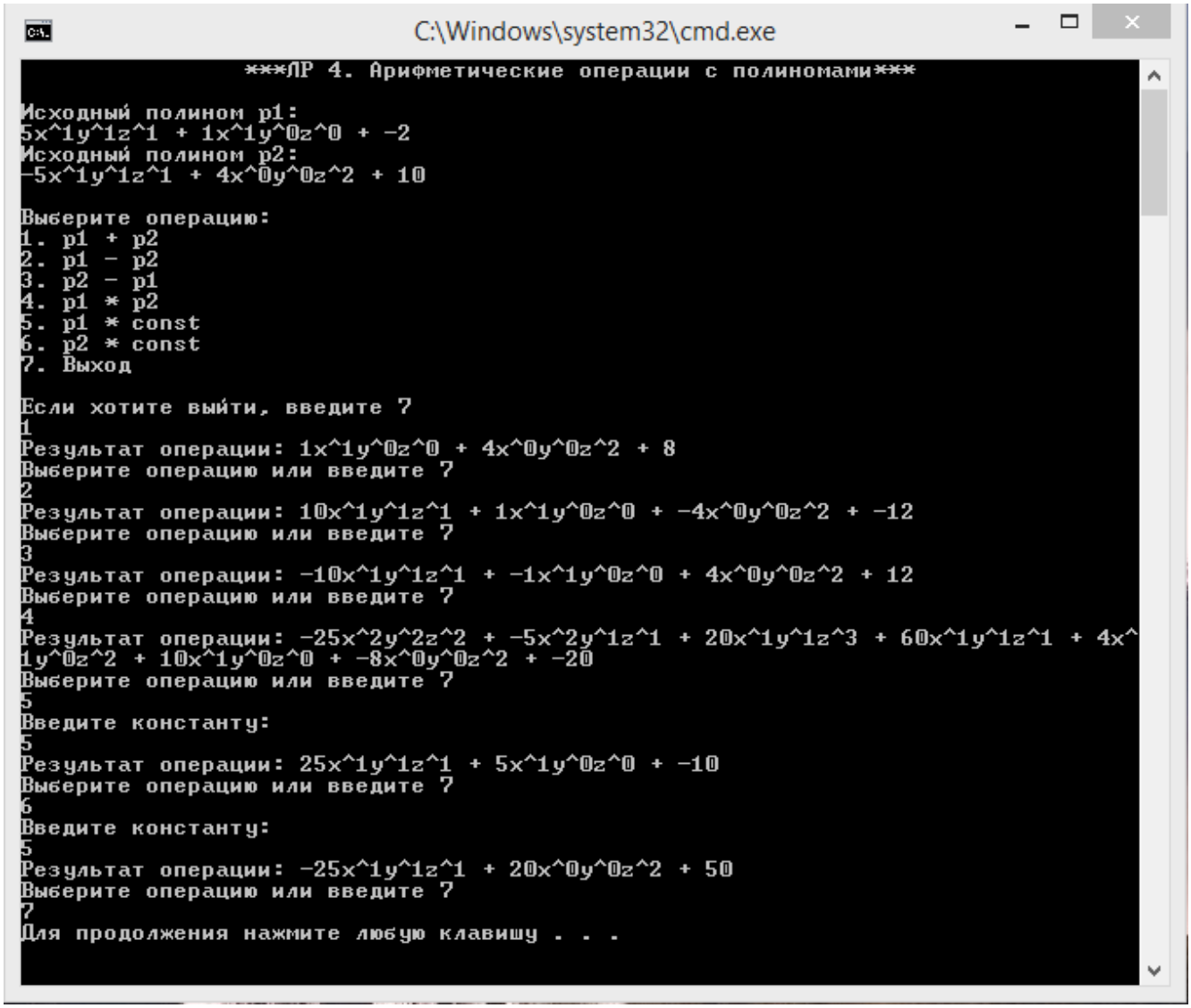
5) *Умножение полиномов*

Создаем новый полином. Используя два цикла, проходим по двум исходным полиномам как по двумерному массиву. Добавляем к новому полиному моном, коэффициент которого равен произведению мономов двух исходных полиномов, а индекс равен сумме их индексов. Если результирующий моном получился со степенями больше 19, то выдаем сообщение об ошибке.

Помимо этих основных операций в классах Monom и Polinom перегружены операторы присваивания и сравнения, а также реализованы конструкторы копирования. Помимо этого перегружены операторы ввода и вывода для класса Polinom.

Пример вычисления выражения

В качестве примера рассмотрим выполнение сложения, вычитания, умножения на константу (допустим, 5), умножения двух полиномов x+5xyz-2 и 4z^2–5xyz+10:

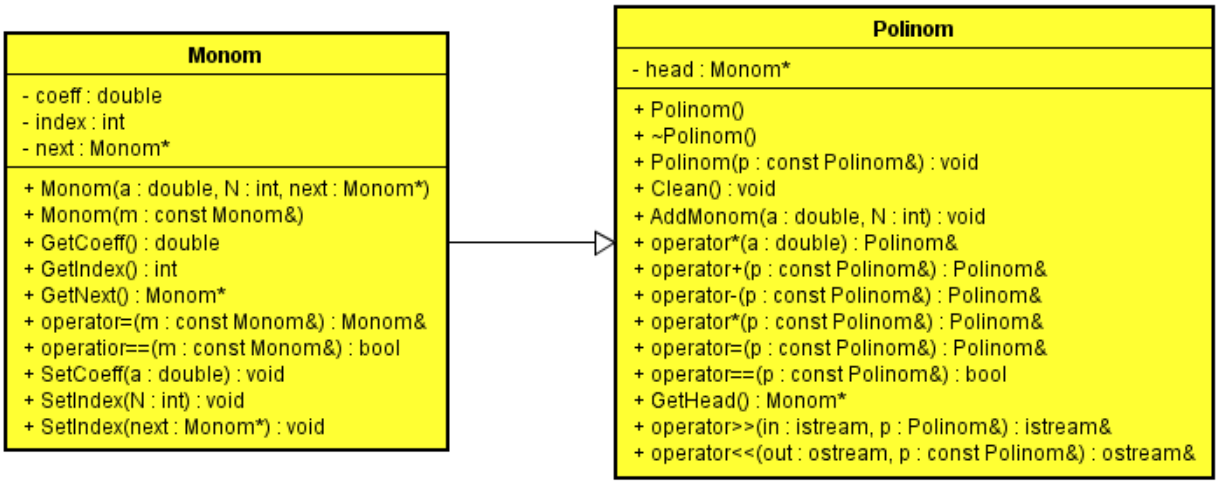


На данном примере продемонстрирована работа консольного и графического приложений программы. Стоит заметить, что при сложении получившийся нулевой моном не выводится.

Структура программы

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) MyForm.h, MyForm.cpp – реализация графического приложения
7) test_main.cpp, test_monom.cpp, test_polinom.cpp – реализация тестов

Схема наследования



Заключение

В результате выполнения данной лабораторной работы было создано программное приложение, позволяющее производить различные стандартные арифметические операции над полиномами.
Данные классы можно совершенствовать, добавляя туда необходимые разработчику функции. К примеру, можно добавить функцию вычисления полинома при заданных значениях переменных или функцию дифференцирования или интегрирования по какой-либо переменной.
Цель данной работы заключалась так же в создании полноценных классов, которыми могут пользоваться другие люди, работе с динамической памятью, перегрузке функций.
Была написана тестовая оболочка (тестовое приложение), в которой можно легко проверить работу данных классов.
Таким образом, задачи, поставленные в начале, были успешно выполнены, а результаты проверены.

Литература

1) Гергель В.П. Рабочие материалы к учебному курсу «Методы программирования», ННГУ, 2002. – 100 с.
2) Б. Страуструп Язык программирования C++. Специальное издание. Пер. с англ. – М.: ООО «Бином-Пресс», 2008 г. – 1104 с.
3) Р.Лафоре. Объектно-ориентированное программирование в языке СИ++.-М: «ПИТЕР», 2004 г.-922 с.

Приложения

monom.h

```
////////////////////////////////////////////////////////////////////////
// monom.h                                                            //
// Арифметические операции с полиномами                               //
// Автор - Краснов А.А., Нижний Новгород, 2016                        //
////////////////////////////////////////////////////////////////////////

#pragma once
#include <iostream>

class Monom
{
	double coeff;
	int index;
	Monom *next;
public:
    Monom(double = 0.0,int N = 0, Monom *_next = NULL);
	Monom(const Monom &m);
	void SetCoeff(double a);
    double GetCoeff();
	void SetIndex(int N);
    int GetIndex();
	void SetNext(Monom *_next);
	Monom* GetNext();
	Monom& operator=(const Monom &m);
	bool operator==(const Monom &m) const;
};
```

monom.cpp

```
////////////////////////////////////////////////////////////////////////
// monom.cpp                                                          //
// Арифметические операции с полиномами                               //
// Автор - Краснов А.А., Нижний Новгород, 2016                        //
////////////////////////////////////////////////////////////////////////

#include "monom.h"

Monom::Monom(double a,int N,Monom *_next)
{
	SetIndex(N);
	SetCoeff(a);
	SetNext(_next);
}

Monom::Monom(const Monom &m)
{
	SetIndex(m.index);
	SetCoeff(m.coeff);
	SetNext(m.next);
}

void Monom::SetCoeff(double a)
{
	coeff=a;
}
```

```
double Monom::GetCoeff()
{
	return coeff;
}

void Monom::SetIndex(int N)
{
	if ((N>=0) && (N<8000)) index=N;
	else if (N<0) throw "Степень < 0 ";
	else if (N>=8000) throw "Степень >= 20 ";
}

int Monom::GetIndex()
{
	return index;
}
void Monom::SetNext(Monom *_next)
{
	next=_next;
}

Monom* Monom::GetNext()
{
	return next;
}

Monom& Monom::operator=(const Monom &m)
{
	SetCoeff(m.coeff);
	SetIndex(m.index);
	return *this;
}


bool Monom::operator==(const Monom &m) const
{
	if ((coeff==m.coeff)&&(index==m.index)) return true;
	else return false;
}
```

polinom.h

```
//////////////////////////////////////////////////////////////////////
// polinom.h                                                        //
// Арифметические операции с полиномами                             //
// Автор - Краснов А.А., Нижний Новгород, 2016                      //
//////////////////////////////////////////////////////////////////////

#pragma once
#include "monom.h"
#include <locale>

using namespace std;

class Polinom : public Monom
{
	Monom *head;
public:
	Polinom();
	~Polinom();
	Polinom(const Polinom &p);
	void Clean();
```

```
	void AddMonom(double a, int N);
	Polinom& operator*(double a) const;
	Polinom& operator+(const Polinom &p) const;
	Polinom& operator-(const Polinom &p) const;
	Polinom& operator*(const Polinom &p) const;
	Polinom& operator=(const Polinom &p);
	bool operator==(const Polinom &p) const;
	Monom* GetHead();
	friend istream& operator>>(istream &in, Polinom &p)
	{
		double k;
		int s1,s2,s3;
		int f;
		while (1)
		{
			setlocale(LC_ALL, "Russian");
			cout << "Введите коэффициент: ";
			in>>k;
			cout << "Введите степени x,y,z: "<<endl;
			in>>s1;
			in>>s2;
			in>>s3;
			if (s1<0||s2<0||s3<0) throw "Степень < 0 ";
			if (s1>20||s2>20||s3>20) throw "Степень >= 20 ";
			int N=s1*400+s2*20+s3;
			p.AddMonom(k,N);
			cout << endl << "Продолжить ввод мономов? 1-да, 0-нет"<<endl;
			in>>f;
			cout<<endl;
			if (f==0) break;
			else if (f==1) continue;
		}
		return in;
	}
	friend ostream& operator<<(ostream &out, const Polinom &p)
	{
		Polinom q(p);
		q.Sort();
		Monom *t = q.head;
		if (t==NULL)
		{
			out<<t->GetCoeff();
			return out;
		}
		if (t->GetNext()==NULL)
		{
			if (t->GetCoeff()==0)
			{
				out<<t->GetCoeff();
				return out;
			}
			else if (t->GetIndex()==0) out<<t->GetCoeff();
			else		out<<t->GetCoeff()<<"x^"<<t->GetIndex()/400<<"y^"<<(t-
>GetIndex()/20)%20<<"z^"<<t->GetIndex()%20<<endl;
			return out;
		}
		while (t->GetNext()!=NULL)
		{
			if (t->GetCoeff()==0.0) t=t->GetNext();
			else if (t->GetIndex()==0)
			{
				out<<t->GetCoeff();
				t=t->GetNext();
			}
			else
```

```
                    {
                            out<<t->GetCoeff()<<"x^"<<t->GetIndex()/400<<"y^"<<(t-
>GetIndex()/20)%20<<"z^"<<t->GetIndex()%20;
                            t=t->GetNext();
                    }
                    if (t->GetNext()!=NULL) out<<" + ";
             }
             if (t->GetCoeff()==0) return out;
             else if (t->GetIndex()==0) out<<" + "<<t->GetCoeff();
             else    out<<"    +    "<<t->GetCoeff()<<"x^"<<t->GetIndex()/400<<"y^"<<(t-
>GetIndex()/20)%20<<"z^"<<t->GetIndex()%20<<endl;
             return out;
      }
   };
```

polinom.cpp

```
//////////////////////////////////////////////////////////////////////
// polinom.cpp                                                      //
// Арифметические операции с полиномами                             //
// Автор - Краснов А.А., Нижний Новгород, 2016                      //
//////////////////////////////////////////////////////////////////////

#include "polinom.h"

using namespace std;

Polinom::Polinom()
{
       head=NULL;
}

Polinom::~Polinom()
{
       Clean();
}

Polinom:: Polinom(const Polinom &p)
{
       head = new Monom;
       head->SetNext(NULL);
       Monom *cur = new Monom;
       cur = p.head;
       while (cur!=NULL)
       {
              AddMonom(cur->GetCoeff(),cur->GetIndex());
              cur = cur->GetNext();
       }
}

Monom* Polinom::GetHead()
{
       return head;
}

void Polinom::Clean()
{
       Monom *t=new Monom;
       while(head!=NULL)
       {
              t=head->GetNext();
              delete head;
```

```
		head=t;
	}
}

void Polinom::AddMonom(double a, int N)
{
	if (a==0) return;
	if (head == NULL) head = new Monom(a,N,NULL);
	else
	{
		if (head->GetIndex() < N)
		{
			Monom *tmp = new Monom(a,N,NULL);
			tmp->SetNext(head);
			head = tmp;
		}
		else
		{
			Monom *cur, *last;
			for (cur=head;cur!=NULL;cur=cur->GetNext())
			{
				if (cur->GetIndex() < N) break;
				if (cur->GetIndex() == N)
				{
					if ((a+cur->GetCoeff())!=0) cur->SetCoeff(a+cur-
>GetCoeff());
					else
					{
						Monom *tmp = head;
						if (cur==head) head = cur->GetNext();
						else
						{
							while (tmp->GetNext()!=cur) tmp=tmp-
>GetNext();
							tmp->SetNext(cur->GetNext());
						}
						delete cur;
					}
					return;
				}
				last = cur;
			}
			if (cur != NULL)
			{
				Monom *p = new Monom(a,N,cur);
				last->SetNext(p);
			}
			else
			{
				Monom *p = new Monom(a,N,NULL);
				last->SetNext(p);
			}
		}
	}
}

Polinom& Polinom:: operator*(double a) const
{
	Polinom *res = new Polinom;
	Polinom q(*this);
	Monom *t=q.GetHead();
	if (a==0.0) return *res;
	while (t!=NULL)
	{
		(*res).AddMonom(a*(t->GetCoeff()),t->GetIndex());
```

```
			t=t->GetNext();
		}
		return *res;
}

Polinom& Polinom:: operator+(const Polinom &p) const
{
		Polinom *res = new Polinom(*this);
		Monom *t=p.head;
		while (t!=NULL)
		{
			(*res).AddMonom(t->GetCoeff(),t->GetIndex());
			t=t->GetNext();
		}
		return *res;
}

Polinom& Polinom:: operator-(const Polinom &p) const
{
		Polinom *res = new Polinom(*this);
		Monom *t=p.head;
		while (t!=NULL)
		{
			(*res).AddMonom(-t->GetCoeff(),t->GetIndex());
			t=t->GetNext();
		}
		return *res;
}

Polinom& Polinom::operator*(const Polinom &p) const
{
		Polinom *res = new Polinom;
		Polinom q(*this);
		for (Monom *t1=p.head;t1!=NULL;t1=t1->GetNext())
			for (Monom *t2=q.GetHead();t2!=NULL;t2=t2->GetNext())
			{
				int s1(0),s2(0),s3(0);
				s1=(t1->GetIndex())%20 + (t2->GetIndex())%20;
				s2=((t1->GetIndex())/20)%20 + ((t2->GetIndex())/20)%20;
				s3=(t1->GetIndex())/400 + (t2->GetIndex())/400;
				if (s1>19||s2>19||s3>19)
					throw ("Одна из степеней итогового полинома >= 20");
				else (*res).AddMonom(t1->GetCoeff() * t2->GetCoeff(), t1->GetIndex()
+ t2->GetIndex());
			}
			return *res;
}

Polinom& Polinom:: operator=(const Polinom &p)
{
		this->Clean();
		head=p.head;
		Monom *l1=p.head;
		Monom *l2=(*this).GetHead();
		l2=l1;
		while (l1!=NULL)
		{
			l2=l1;
			l1=l1->GetNext();
			l2=l2->GetNext();
		}
		return *this;
}

bool Polinom:: operator==(const Polinom &p) const
```

```
{
	Polinom *q = new Polinom(p);
	if (head==NULL)
	{
		if (q->head==NULL) return true;
		else return false;
	}
	Monom *t1=head;
	Monom *t2=q->head;
	int flag(1);
	while (t1!=NULL)
	{
		if ((t1->GetCoeff()!=t2->GetCoeff())||(t1->GetIndex()!=t2->GetIndex()))
		{
			flag=0;
			break;
		}
		t1=t1->GetNext();
		t2=t2->GetNext();
	}
	if (flag==1) return true;
	else return false;
   }
```

sample.cpp

```
////////////////////////////////////////////////////////////////////////
// sample.cpp                                                         //
// Арифметические операции с полиномами                               //
// Автор - Краснов А.А., Нижний Новгород, 2016                        //
////////////////////////////////////////////////////////////////////////

#include "polinom.h"

void main()
{
	setlocale(LC_ALL, "Russian");
	try
	{
		cout << "\t\t***ЛР 4. Арифметические операции с полиномами***\n\n";
		cout << "Правила ввода:"<< endl;
		cout << "1) Вводимый полином состоит из мономов от 3-х переменных" << endl
			<< "2) Степени должны быть в промежутке от 0 до 19 включительно" <<
endl
			<< "3) Допустимые операции: умножение на константу, +, -, *" << endl
		    << "4) При умножении полиномов их степени в сумме не должны быть > 19"
<< endl<<endl;
		int f(0),count(0);
		cout << "Введите полином p1" <<endl;
		Polinom *p1 = new Polinom;
		cin>>(*p1);
		cout << "Введите полином p2" <<endl;
		Polinom *p2 = new Polinom;
		cin>>(*p2);
		system("cls");
		cout << "\t\t***ЛР 4. Арифметические операции с полиномами***\n\n";
		cout << "Исходный полином p1: "<<endl<<*p1<<endl;
		cout << "Исходный полином p2: "<<endl<<*p2<<endl<<endl;
		cout<<"Выберите операцию:"<<endl;
		cout<<"1. p1 + p2"<<endl;
		cout<<"2. p1 - p2"<<endl;
		cout<<"3. p2 - p1"<<endl;
		cout<<"4. p1 * p2"<<endl;
		cout<<"5. p1 * const"<<endl;
```

```
		cout<<"6. p2 * const"<<endl;
		cout<<"7. Выход"<<endl<<endl;
		cout<<"Если хотите выйти, введите 7"<<endl;
		cin >> count;
		while (count!=7)
		{
			switch (count)
			{
			case 1:
				{
					cout << "Результат операции: "<<(*p1)+(*p2)<<endl;
					break;
				}
			case 2:
				{
					cout << "Результат операции: "<<(*p1)-(*p2)<<endl;
					break;
				}
			case 3:
				{
					cout << "Результат операции: "<<(*p2)-(*p1)<<endl;
					break;
				}
			case 4:
				{
					cout << "Результат операции: "<<(*p1)*(*p2)<<endl;
					break;
				}
			case 5:
				{
					int k(0);
					cout << "Введите константу:"<<endl;
					cin>>k;
					cout << "Результат операции: "<<(*p1)*k<<endl;
					break;
				}
			case 6:
				{
					int k(0);
					cout << "Введите константу:"<<endl;
					cin>>k;
					cout << "Результат операции: "<<(*p2)*k<<endl;
					break;
				}
			default:
				{
					cout <<"Неправильный ввод" << endl;
					break;
				}
			}
			cout << "Выберите операцию или введите 7" << endl;
			cin >> count;
		}
	}
	catch (const char* error)
	{
		cout<<error<<endl;
	}
}
```