НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ

КАФЕДРА МАТЕМАТИЧЕСКОГО И КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ

**ЛАБОРАТОРНАЯ РАБОТА № 1**

по дисциплине: «Технологии программирования»

«Разработка классов»

Выполнил студент гр. А-16-20

Симаков Александр Максимович

Принял преподаватель

А.В. Князев

Москва 2022

* **Задание для ЛР**

1) Составить на языке С++ описания классов для указанных объектов.

2) В среде Visual Studio разработать консольную программу, иллюстрирующую использование объектов заданных классов.

3) Программа должна иметь меню с пунктами для ввода исходных данных, вывода результатов, выполнения требуемых операций над объектами.

4) Отчёт по лабораторной работе должен содержать:

-Титульный лист

-Задание на работу (общее и индивидуальное)

-Описание работы программы

-Алгоритмы выполнения основных операций на псевдокоде

-Тесты

-Распечатки экранов при работе программы

-Листинг программы

**Личное задание**

Реализовать многочлен на основе двунаправленного списка (сложение, умножение)

* **Описание работы программы**

В самом начале пользователя встречает меню, реализованное с помощью *switch-case* (в зависимости от выбора пункта программа будет обрабатывать разные запросы, также были учтены возможные ошибки). Ранее был создан список, который будет выводиться в качестве результата работы некоторых пунктов программы. Далее просто **список**. Присутствует функция, которая добавляет в список новый член. При этом она параллельно проверяет, есть ли в нём подобные. В случае наличия происходит сумма коэффициентов, иначе создаётся элемент в конце списка.

Если в качестве действия было выбрано сложение многочленов, просто добавляем всё, подобные приводятся с помощью ранее описанной функции.

Если в качестве действия было выбрано умножение многочленов, то сначала происходит перемножение по схеме *«каждый с каждым»,* далее результат добавляется в список.

* **Алгоритмы выполнения основных операций на**

**псевдокоде**

Основные функции:

* addBringEnd

Смысл: добавление элемента в конец списка или приведение подобного слагаемого, добавление с приведением.

-Если отсутствует первый элемент (список пуст)

-Выделяю память и создаю элемент с входными данными

-Присваиваю первый и последний элемент списка данному элементу

-Следующий предыдущий элемент списка присваиваю NULL

-Присваиваю данному элементу все входные данные

-Иначе

-Проверяю, есть ли в списке, к которому добавляю элемент, элемент с такими же коэффициентами при

-Если есть, то просто складываю коэффициенты, входных данных, и с найденным подобным элементом

-Если нет, то создаю новый элемент по стандартной схеме, и добавляю его в конец списка

* sum

Смысл: сложение многочленов

-Пока указатель на текущий одночлен 1 многочлена не равен NULL

-Добавляю данный одночлен в итоговый ответ

-Перехожу к следующему элементу 1 многочлена

-Пока указатель на текущий одночлен 2 многочлена не равен NULL

-Добавляю данный одночлен в итоговый ответ

-Перехожу к следующему элементу 2 многочлена

* product

Смысл: умножение многочленов

-Пока указатель на текущий одночлен 1 многочлена не равен NULL

-Пока указатель на текущий одночлен 2 многочлена не равен NULL

-Перехожу к следующему элементу 2 многочлена

-Перехожу к следующему элементу 1 многочлена

-Добавляю в итоговый список элемент с перемноженными коэффициентами и сложенными степенями

(При добавлении элемента приведение производится автоматически, смотреть функцию addBringEnd)

Вспомогательные функции:

* print

Смысл: вывод многочленов на экран

-Если элемент в списке один, вывожу его составляющие

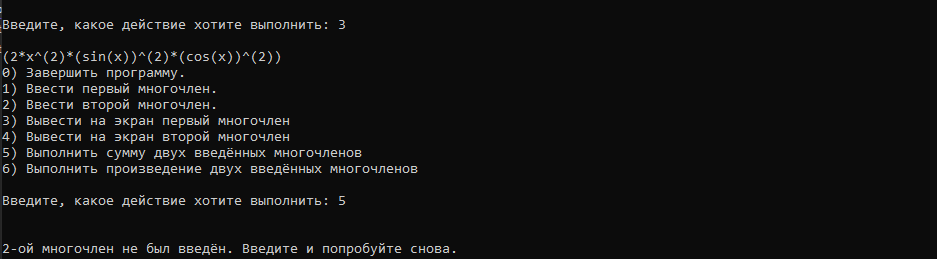
-Если первый элемент равен NULL, списка не существует, вывожу такое сообщение

-Иначе

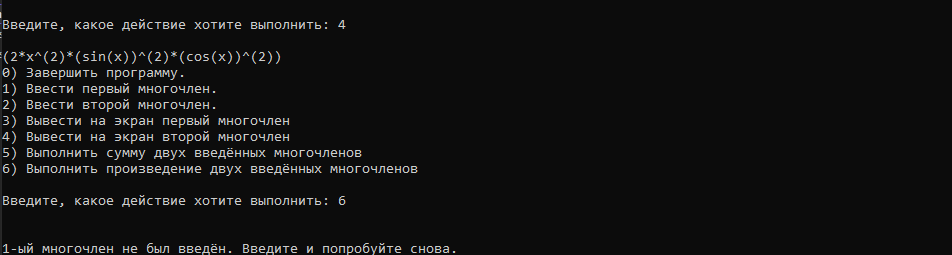
-Пока не выхожу на границы списка, вывожу данный элемент на экран

* **Тесты с подтверждениями (скриншоты программы)**

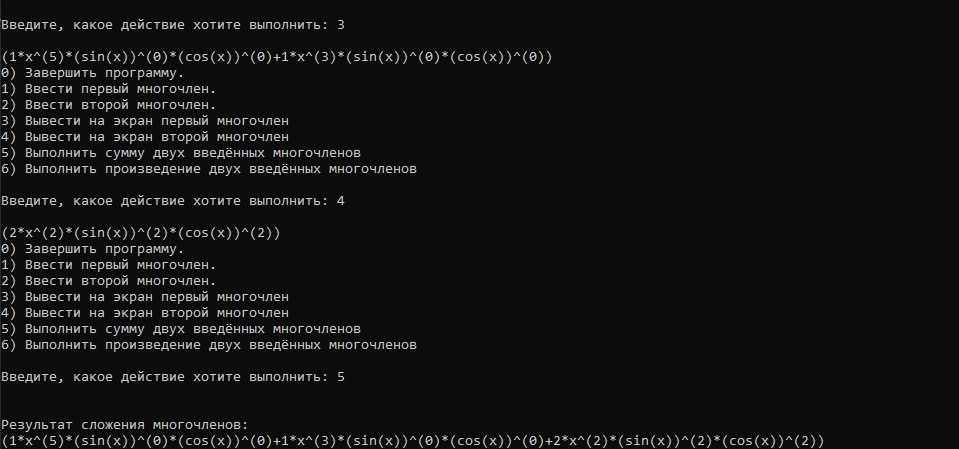
|  |  |  |  |
| --- | --- | --- | --- |
| **Номер теста** | **Входные данные** | **Ожидаемый результат** | **Смысл теста** |
| 1 | 1-ый многочлен:  2-ой многочлен:  нет  Сумма | Сообщение об отсутствии 2-ого многочлена. | Проверка поведения программы при выборе суммы многочленов при отсутствии 2-ого многочлена |
| 2 | 1-ый многочлен:  2-ой многочлен:  Произведение | Сообщение об отсутствии 1-ого многочлена. | Проверка поведения программы при выборе произведения многочленов при отсутствии 1-ого многочлена |
| 3 | 1-ый многочлен:  2-ой многочлен:  Сумма | + | Проверка корректности работы программы при простой сумме. |
| 4 | 1-ый многочлен:  2-ой многочлен:  Произведение |  | Проверка корректности работы программы при простом произведении. |
| 5 | 1-ый многочлен:  2-ой многочлен:  Сумма |  | Проверка корректности работы программы при сложной сумме. |
| 6 | 1-ый многочлен:  2-ой многочлен:  Произведение |  | Проверка корректности работы программы при сложном произведении. |



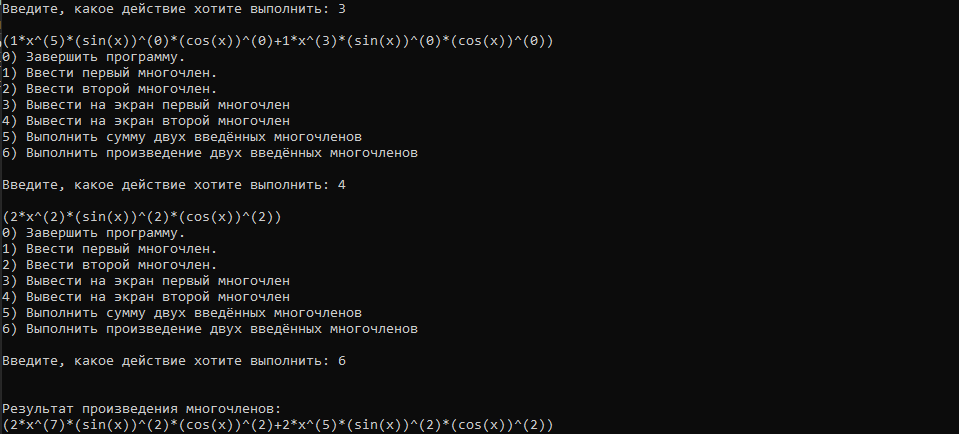




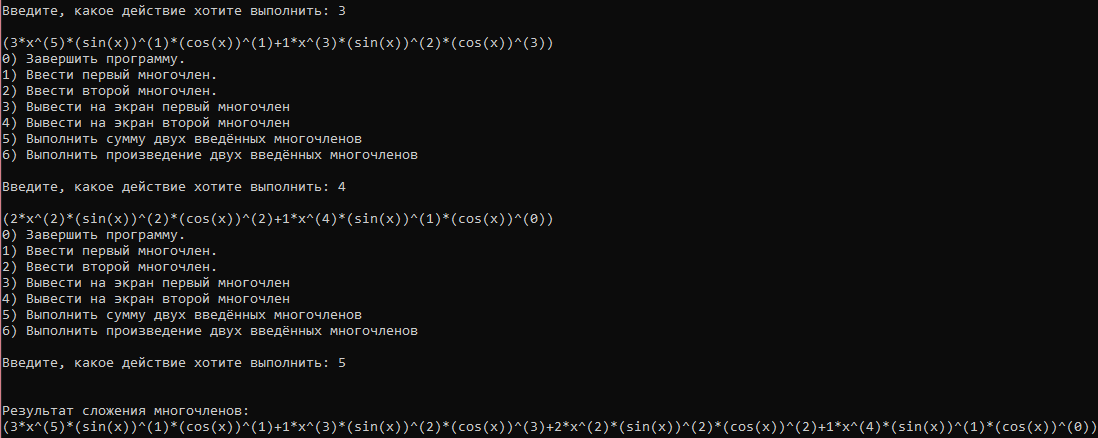




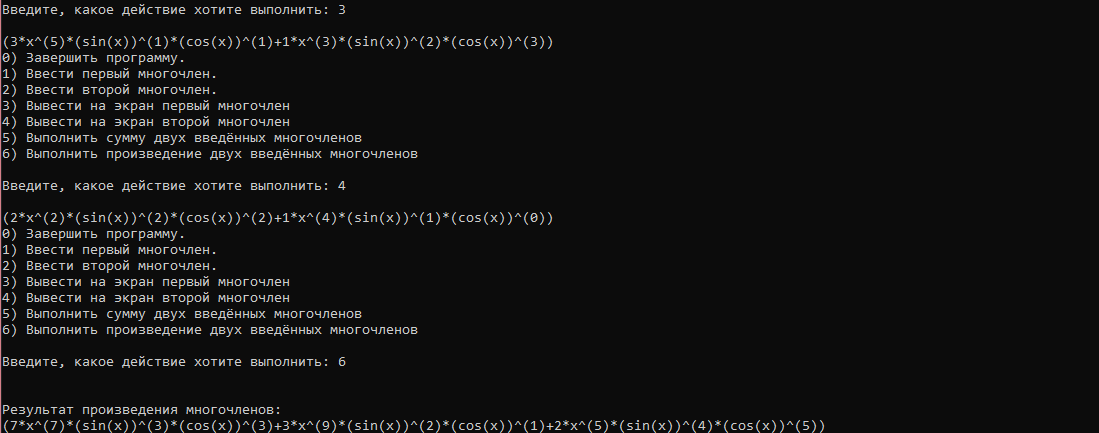












* **Листинг программы**

**lab1.cpp**

#include "list.h"

int main()

{

setlocale(LC\_ALL, "Russian");

int num = 0, k = 0, x = 0, sinx = 0, cosx = 0, m = -1, v;

list lA, lB, lRes;

bool exA = false, exB = false;

while (m != 0)

{

printf("\n0) Завершить программу.\n");

printf("1) Ввести первый многочлен.\n");

printf("2) Ввести второй многочлен.\n");

printf("3) Вывести на экран первый многочлен\n");

printf("4) Вывести на экран второй многочлен\n");

printf("5) Выполнить сумму двух введённых многочленов\n");

printf("6) Выполнить произведение двух введённых многочленов\n");

printf("\nВведите, какое действие хотите выполнить: ");

scanf("%d", &m);

printf("\n");

switch (m)

{

case 0:

{

printf("Программа завершена.\n");

break;

}

case 1:

{

v = 1;

if (lA.first != NULL)

lA.~list();

printf("Введите количество слагаемых в вашем 1-ом многочлене: ");

scanf("%d", &num);

for (int i = 1; i <= num; i++)

{

printf("Введите коэффицент k, который стоит возле %d-го члена 1-го многочлена: ", v);

scanf("%d", &k);

printf("Введите степень, которую имеет x в %d-ом члене 1-го многочлена: ", v);

scanf("%d", &x);

printf("Введите степень, которую имеет sin(x) в %d-ом члене 1-го многочлена: ", v);

scanf("%d", &sinx);

printf("Введите степень, которую имеет сos(x) в %d-ом члене 1-го многочлена: ", v);

scanf("%d", &cosx);

v++;

lA.addBringEnd(k, x, sinx, cosx);

}

exA = true;

break;

}

case 2:

{

v = 1;

if (lB.first != NULL)

lB.~list();

printf("Введите количество слагаемых в вашем 2-ом многочлене: ");

scanf("%d", &num);

for (int i = 1; i <= num; i++)

{

printf("Введите коэффицент k, который стоит возле %d-го члена 2-го многочлена: ", v);

scanf("%d", &k);

printf("Введите степень, которую имеет x в %d-ом члене 2-го многочлена: ", v);

scanf("%d", &x);

printf("Введите степень, которую имеет sin(x) в %d-ом члене 2-го многочлена: ", v);

scanf("%d", &sinx);

printf("Введите степень, которую имеет сos(x) в %d-ом члене 2-го многочлена: ", v);

scanf("%d", &cosx);

v++;

lB.addBringEnd(k, x, sinx, cosx);

}

exB = true;

break;

}

case 3:

{

if (exA)

lA.print();

else

printf("\n1-ый многочлен не был введён. Введите и попробуйте снова.\n");

break;

}

case 4:

{

if (exB)

lB.print();

else

printf("\n2-ой многочлен не был введён. Введите и попробуйте снова.\n");

break;

}

case 5:

{

if (!exA)

printf("\n1-ый многочлен не был введён. Введите и попробуйте снова.\n");

if (!exB)

printf("\n2-ой многочлен не был введён. Введите и попробуйте снова.\n");

if (exA && exB)

{

if (lRes.first != NULL)

lRes.~list();

lRes.sum(lA, lB);

printf("\nРезультат сложения многочленов: \n");

lRes.print();

}

break;

}

case 6:

{

if (!exA)

printf("\n1-ый многочлен не был введён. Введите и попробуйте снова.\n");

if (!exB)

printf("\n2-ой многочлен не был введён. Введите и попробуйте снова.\n");

if (exA && exB)

{

if (lRes.first != NULL)

lRes.~list();

lRes.product(lA, lB);

printf("\nРезультат произведения многочленов: \n");

lRes.print();

}

break;

}

}

}

return 0;

}

**list.cpp**

#include "list.h"

list::list()

{

next = NULL;

prev = NULL;

first = NULL;

last = NULL;

k = 0;

xDeg = 0;

sinDeg = 0;

cosDeg = 0;

}

list::list(int kH, int xDegH, int sinDegH, int cosDegH)

{

first = NULL;

last = NULL;

next = NULL;

prev = NULL;

k = kH;

xDeg = xDegH;

sinDeg = sinDegH;

cosDeg = cosDegH;

}

list::~list()

{

while (first != NULL)

{

now = first;

first = first->next;

delete now;

}

}

void list::print()

{

list\* help = first;

if (first == last)

printf("(%d\*x^(%d)\*(sin(x))^(%d)\*(cos(x))^(%d))", help->k, help->xDeg, help->sinDeg, help->cosDeg);

else

{

if (first == NULL)

printf("Спиcок не существует, или потерян указатель на его корень");

printf("(");

while (help != NULL)

{

printf("%d\*x^(%d)\*(sin(x))^(%d)\*(cos(x))^(%d)", help->k, help->xDeg, help->sinDeg, help->cosDeg);

if (help != last)

printf("+");

help = help->next;

}

printf(")");

}

return;

}

void list::addBringEnd(int kH, int xDegH, int sinDegH, int cosDegH)

{

if (first == NULL)

{

list\* temp = new list(kH, xDegH, sinDegH, cosDegH);

first = temp;

last = temp;

temp->prev = NULL;

temp->next = NULL;

temp->k = kH;

temp->xDeg = xDegH;

temp->sinDeg = sinDegH;

temp->cosDeg = cosDegH;

}

else

{

bool flag = false;

now = first;

while ((!flag) && (now != NULL))

{

if ((now->xDeg == xDegH) && (now->cosDeg == cosDegH) && (now->sinDeg == sinDegH))

flag = true;

else

now = now->next;

}

if (flag)

now->k += kH;

else

{

list\* temp = new list(kH, xDegH, sinDegH, cosDegH);

last->next = temp;

temp->prev = last;

temp->next = NULL;

last = temp;

temp->k = kH;

temp->xDeg = xDegH;

temp->sinDeg = sinDegH;

temp->cosDeg = cosDegH;

}

}

return;

}

void list :: sum(list &a, list &b)

{

a.now = a.first;

while (a.now != NULL)

{

addBringEnd(a.now->k, a.now->xDeg, a.now->sinDeg, a.now->cosDeg);

a.now = a.now->next;

}

b.now = b.first;

while (b.now != NULL)

{

addBringEnd(b.now->k, b.now->xDeg, b.now->sinDeg, b.now->cosDeg);

b.now = b.now->next;

}

}

void list :: product(list &a, list &b)

{

a.now = a.first;

b.now = b.first;

while (a.now != NULL)

{

b.now = b.first;

while(b.now != NULL)

{

addBringEnd((a.now->k \* b.now->k), (a.now->xDeg + b.now->xDeg), (a.now->sinDeg + b.now->sinDeg), (a.now->cosDeg + b.now->cosDeg));

b.now = b.now->next;

}

a.now = a.now->next;

}

}

**list.h**

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

class list

{

public:

//Для формирования списка)

list\* first; //Корень списка, указатель на первый элемент

list\* last; // последний элемент

list\* next; //Следующий элемент

list\* prev; //Предыдущий элемент

list\* now; //Текущий элемент

// Переменные определяющие наш многочлен

int k; //Коэффицент многочлена

int xDeg; //Степень при x

int sinDeg; //Степень при sin(x)

int cosDeg; //Степень при cos(x)

list();

//Конструктор класса

list(int kH, int xDegH, int sinDegH, int cosDegH);

//деструктор

~list();

//Вывод списка на экран

void print();

//Добавление элемента в конец списка или приведение подобных слагаемых, добавление с приведением

void addBringEnd(int kH, int xDegH, int sinDegH, int cosDegH);

// сложения многочленов

void sum(list& a, list& b);

// перемножение многочленов

void product(list& a, list& b);

};