|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **МИРЭА** | | |

Лабораторная работа №2

по дисциплине

**«Структуры и алгоритмы ОД»**

Отчет выполнен студентом

1 курса

Группы: ИВБО-06-17

Бикеевым Адилем

*2 вариант*

Москва 2019

**Содержание**

1. Задание 1
   1. Вариант задания

Разработать многомодульную программу, которая демонстрирует выполнение всех операций, определенных вариантом, над линейным однонаправленным динамическим списком.

Требования к разработке.

1. Разработать структуру узла списка, структура информационной части узла определена вариантом. Определение структуры узла списка выполните через спецификацию typedef в отдельном заголовочном файле.
2. Разработайте функции для выполнения операции над линейным списком:

* создание списка из n узлов
* вывод списка
* поиск узла с заданным значением (операция должна возвращать указатель на заданный узел).

1. Оформите операции, предложенные вариантов в виде функций и включите в отдельный файл с расширением cpp. Подключите к этому файлу заголовочный файл с определением структуры узла.
2. Разработайте программу, управляемую текстовым меню, согласно заданию и включите в меню демонстрацию выполнения всех операций задания и варианта.

|  |  |
| --- | --- |
| Тип информационной части узла списка | Дополнительные Операции |
| Номер телефона (из 7 цифр), время разговора (целое число), номер телефона вызываемого абонента. | Добавить новый узел в список, упорядочивая узлы по первым четырем цифрам телефона.  Удалить последний узел с заданным значением телефона.  Подсчитать суммарное время разговора с заданного телефона. |

Примечание. В определении информационной части узла варианта, подчеркнутое поле считать полем ключа.

* 1. Абстрактный тип данных (далее АТД) для варианта задания, включая список общих функций из задания.

struct Telephone

{

    //Номер отправителя

    int phoneFrom;

    //Номер получателя

    int phoneTo;

    //Время разговора в минутах

    int timeTell;

};

|  |  |
| --- | --- |
| Объявление функции | Назначение |
| Node\* createList(Node\* head, int n); | Создает список из n узлов |
| void showList(Node\* list); | Выводит на экран список list из n узлов |
| Node\* findNode(Node\* current, int phoneFrom); | Находит узел с заданынм номером отправтеля |
| Node\* getLastNode(Node\* current); | Возвращает указатель на последний элемент списка |
| void addNode(Node\* lastNode, Node\* nodeNew); | Добавляет в конец списка новый узел |
| int getNum(int phone); | Возвращает первые 4 цифры телефона |
| void swapDataNode(Node\* node1, Node\* node2); | Меняет значения элементов списка |
| void sortList(Node\* head); | Сортирует список по первым четырем цифрам телефона |
| Node\* getLastNodeForDelete(Node\* head, int phoneFrom); | Возвращает ссылку на последний узел, содержащий соответствующий номер отправителя |
| void deletChoosenNode(Node\* head); | Удаляет выбранный узел |
| int sumTimeTell(Node\* head, int phoneFrom); | Возвращает суммарную длиельность разговора по указанному номеру отправителя |

* 1. Реализация АТД
* Определение типа информационной части узла(typedef)

typedef struct Node

{

    //Номер отправителя

    int phoneFrom;

    //Номер получателя

    int phoneTo;

    //Время разговора в минутах

    int timeTell;

    //Ссылка на следующий элемент списка

    struct Node\* next;

} Telephone;

* Алгоритмы задач варианта, которые не включены в АТД на псевдокоде и использующие операции из АТД.

|  |  |
| --- | --- |
| Объявление функции | Алгоритм на псевдокоде |
| Node\* findNode(Node\* current, int phoneFrom); | Ставим указатель на голову лин. Списка  ПОКА current.phoneFrom != i.phoneFrom:  ЕСЛИ current.next != null\_ptr:  current = current.next  КЦ ЦИКЛА:  ЕСЛИ current.phoneFrom != i.phoneFrom:  ВЕРНУТЬ current  ИНАЧЕ  ВЕРНУТЬ null\_ptr |
| Node\* getLastNode(Node\* current); | Ставим указатель на голову лин. Списка  ПОКА current != null\_ptr И current.nexxt != null\_ptr  current = current.next  КЦ ЦИКЛА:  ВЕРНУТЬ current |
| void addNode(Node\* lastNode, Node\* nodeNew); | Ставим указатель на голову лин. Списка  ПОКА current != null\_ptr И current.nexxt != null\_ptr  current = current.next  КЦ ЦИКЛА:  current.next = nodeNew |
| void swapDataNode(Node\* node1, Node\* node2); | Создаем переменную для хранении копии узла (copy\_node)  Свапаем данные узлов посредством copy\_node |
| void deletChoosenNode(Node\* head); | Ставим указатель на голову лин. Списка  ПОКА current != head  current = current.next  КЦ ЦИКЛА:  УДАЛИТЬ current из списка |
| int sumTimeTell(Node\* head, int phoneFrom); | Ставим указатель на голову лин. Списка  Для хранения сумм. Времени используем переменную sum  ПОКА current != null\_ptr И current.nexxt != null\_ptr  sum += current.timeTell  КЦ ЦИКЛА:  ВЕРНУТЬ sum |

* Список модулей реализации АТД (или описать где расположена реализация АТД)

Реализация модуля расположена в корневой директории проекта в файле «Operation.cpp»

* 1. Таблица Тестов

|  |  |  |  |
| --- | --- | --- | --- |
| № теста | Эталон результата | Результат Программы | Результат анализа теста |
| 1 | 1 пункт) Создаем новый список из 3 узлов  Осталость добавить: 3  Введите номер получателя: 1  Введите номер отправителя: 1  Введите время разговора: 1  Осталость добавить: 2  Введите номер получателя: 2  Введите номер отправителя: 2  Введите время разговора: 2  Осталость добавить: 1  Введите номер получателя: 3  Введите номер отправителя: 3  Введите время разговора: 3  2 пункт) Вывод списка  Номер отправителя: 1 Номер получателя: 1 Время разговора: 1  Номер отправителя: 2 Номер получателя: 2 Время разговора: 2  Номер отправителя: 3 Номер получателя: 3 Время разговора: 3  3 пункт) Найти первый узел с указанным номером отправителя  Введите номер телефона отправителя: 2  Номер отправителя: 2 Номер получателя: 2 Время разговора: 2  4 пункт) Добавить новый узел  Осталость добавить: 1  Введите номер получателя: 4  Введите номер отправителя: 4  Введите время разговора: 4  6 пункт) Удалить последний узел с заданным значением телефона отправителя  Введите номер телефона отправителя: 4  Операция выполнена успешно  2 пункт) Вывод списка  Номер отправителя: 1 Номер получателя: 1 Время разговора: 1  Номер отправителя: 2 Номер получателя: 2 Время разговора: 2  Номер отправителя: 3 Номер получателя: 3 Время разговора: 3  0 пункт) Выход | 1 пункт) Создаем новый список из 3 узлов  Осталость добавить: 3  Введите номер получателя: 1  Введите номер отправителя: 1  Введите время разговора: 1  Осталость добавить: 2  Введите номер получателя: 2  Введите номер отправителя: 2  Введите время разговора: 2  Осталость добавить: 1  Введите номер получателя: 3  Введите номер отправителя: 3  Введите время разговора: 3  2 пункт) Вывод списка  Номер отправителя: 1 Номер получателя: 1 Время разговора: 1  Номер отправителя: 2 Номер получателя: 2 Время разговора: 2  Номер отправителя: 3 Номер получателя: 3 Время разговора: 3  3 пункт) Найти первый узел с указанным номером отправителя  Введите номер телефона отправителя: 2  Номер отправителя: 2 Номер получателя: 2 Время разговора: 2  4 пункт) Добавить новый узел  Осталость добавить: 1  Введите номер получателя: 4  Введите номер отправителя: 4  Введите время разговора: 4  6 пункт) Удалить последний узел с заданным значением телефона отправителя  Введите номер телефона отправителя: 4  Операция выполнена успешно  2 пункт) Вывод списка  Номер отправителя: 4 Номер получателя: 1 Время разговора: 1  Номер отправителя: 4 Номер получателя: 2 Время разговора: 2  Номер отправителя: 4 Номер получателя: 3 Время разговора: 3  0 пункт) Выход | **+** |
| 2 | 1 пункт)  Список из скольких элементов вы хотите создать? : 0  2 пункт) Вывод списк  Список пуст | 1 пункт)  Список из скольких элементов вы хотите создать? : 0  2 пункт) Вывод списк  Список пуст | **+** |
| 3 | 2 пункт) Вывод списк  Список пуст  3 пункт)  Список пуст  5 пункт)  Список пуст  6 пункт)  Список пуст  7 пункт)  Список пуст  0 пункт) Выход | 2 пункт) Вывод списк  Список пуст  3 пункт)  Список пуст  5 пункт)  Список пуст  6 пункт)  Список пуст  7 пункт)  Список пуст  0 пункт) Выход | **+** |

* 1. Код основной программы с комментариями

#include <iostream>

#include <windows.h>

#include "Operation.h"

#include "Telephone.h"

using namespace std;

int main()

{

    SetConsoleCP(1251);

    SetConsoleOutputCP(1251);

    short answer2 = 10;

    Telephone\* list = nullptr;

    while (answer2 != 0) {

        system("cls");

        cout << "Лабораторная работа №1 Бикеева А. И. Вариант 2\n\n";

        cout << "Задание 1\nРазработать многомодульную программу, \nкоторая демонстрирует выполнение всех операций, \nопределенных вариантом, над линейным однонаправленным \nдинамическим списком.\n";

        cout << "\nМеню\n";

        cout << "1) Создать новый список из n узлов\n";

        cout << "2) Вывести список\n";

        cout << "3) Найти первый узел с указанным номером отправителя\n";

        cout << "4) Добавит новый узел в список\n";

        cout << "5) Упорядочить список по 1-ым четырем цифрам телефона\n";

        cout << "6) Удалить последний узел с заданным значением телефона отправителя\n";

        cout << "7) Подчитать суммарное время разговора с заданным номером отправителя\n";

        cout << "0) Выход\n";

        cout << "\nВаш выбор: ";

        while ((!(cin >> answer2)) || cin.get() != '\n') {     //делаем проверку на ввод букв

            cout << "ERROR\n";

            cin.clear(); // сбрасывает все биты потока, тоесть поток становится "хорошим"

            cin.sync();//Удалим данные из буффера

            cout << "Повторите ввод : ";

        }

        system("cls");

        cout << "Лабораторная работа №1 Бикеева А. И. Вариант 2\n\n";

        switch (answer2)

        {

        case 0: {

            system("pause");

            break;

        }

        case 1:

        {

            if (list)

            {

                free(list);

            }

            int n = 0;

            cout << "Список из скольких элементов вы хотите создать? : ";

            cin >> n;

            cout << "\n\n";

            list = (Telephone\*)malloc(sizeof(Telephone));

            list = createList(list, n);

            cout << "\n\nСписок сформирован";

            system("pause");

            break;

        }

        case 2:

        {

            if (list)

            {

                cout << "Вывод списка\n\n";

                showList(list);

            }

            else {

                cout << "Список пуст";

            }

            system("pause");

            break;

        }

        case 3:

        {

            if (list)

            {

                int phoneFrom = 0;

                cout << "Введите номер телефона отправителя: "; cin >> phoneFrom;

                Telephone\* finded = findNode(list, phoneFrom);

                if (finded)

                {

                    cout << "\n\n";

                    cout << "Номер отправителя: " << finded->phoneFrom << "\tНомер получателя: " << finded->phoneTo << "\tВремя разговора: " << finded->timeTell << endl;

                }

                else {

                    cout << "Узле с указанным телефоном отправителя - не найден"<<endl;

                }

            }

            else {

                cout << "Список пуст\n\n";

            }

            system("pause");

            break;

        }

        case 4:

        {

            if (list)

            {

                addNode(getLastNode(list), nullptr);

                cout << "\n\nНовый узел добавлен";

            }

            else {

                cout << "Список пуст\n\n";

            }

            system("pause");

            break;

        }

        case 5:

        {

            if (list)

            {

                sortList(list);

                cout << "\n\nСписок упорядочен" << endl;

            }

            else {

                cout << "Список пуст\n\n";

            }

            system("pause");

            break;

        }

        case 6:

        {

            if (list)

            {

                deletChoosenNode(list);

                cout << "\n\nОперация выполнена успешно" << endl;

            }

            else {

                cout << "Список пуст\n\n";

            }

            system("pause");

            break;

        }

        case 7:

        {

            if (list)

            {

                int phoneFrom = 0;

                cout << "Введите номер телефона отправителя: "; cin >> phoneFrom;

                cout << "\n\nВремя разговора с данным телефоном = "<< sumTimeTell(list, phoneFrom)<< endl;

            }

            else {

                cout << "Список пуст\n\n";

            }

            system("pause");

            break;

        }

        default:

        {

            system("pause");

        }

        break;

        }

    }

    if (list)

    {

        free(list);

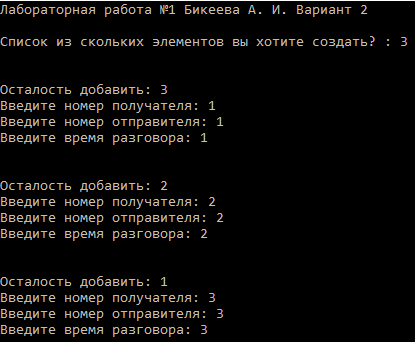
    }

    return 0;

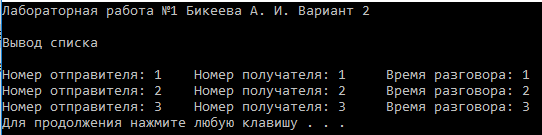
}

* 1. Скриншоты выполнения тестов

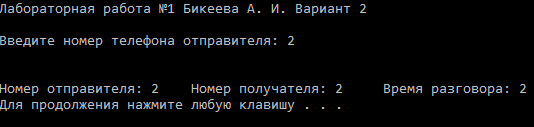
**1 пункт**



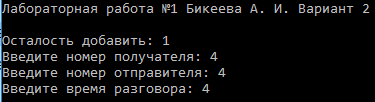
**2 пункт**



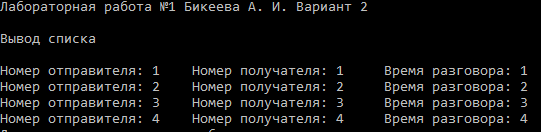
**3 пункт**



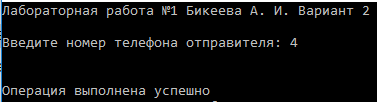
**4 пункт**



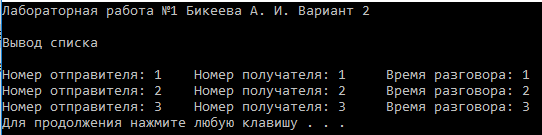
**2 пункт**



**6 пункт**



**2 пункт**



**0 пункт**



1. Задание 2

**Разработка тестов**

|  |  |  |  |
| --- | --- | --- | --- |
| №  теста | Эталон результата | Результат программы | Результат анализа  теста (+ или -) |
|
| 1 | 1 пункт)  Добавляем 1 многочлен  Введите степень многочлена: 3  Введите коэффициенты при Х^0: 1  Введите коэффициенты при Х^1: 1  Введите коэффициенты при Х^2: 1  1 пункт)  Добавляем 2 многочлен  Введите степень многочлена: 3  Введите коэффициенты при Х^0: 2  Введите коэффициенты при Х^1: 2  Введите коэффициенты при Х^2: 2  2 пункт)  Выводим многочлены  1) +1Х^0 +1Х^1 +1Х^2  2) +2Х^0 +2Х^1 +2Х^2  3 пункт)  1) +1Х^0 +1Х^1 +1Х^2  2) +2Х^0 +2Х^1 +2Х^2  Введите номера многочленов которые хотите сложить  (в качесвте 1 номера укажите многочлен к которому желаете прибавить 2-ой многочлен): 1  2  2 пункт)  Выводим многочлены  1) +3Х^0 +3Х^1 +3Х^2  2) +2Х^0 +2Х^1 +2Х^2  4 пункт)  1) +3Х^0 +3Х^1 +3Х^2  2) +2Х^0 +2Х^1 +2Х^2  Введите номера многочленов для приравнивания их друг к другу(1 номер - многочлен, чьи коэфф. сравняются с коэф. 2 номер): 1  2  2 пункт)  Выводим многочлены  1) +2Х^0 +2Х^1 +2Х^2  2) +2Х^0 +2Х^1 +2Х^2  5 пункт)  Выводим многочлены  1) +2Х^0 +2Х^1 +2Х^2  2) +2Х^0 +2Х^1 +2Х^2  Введите номер многочлена: 2  Введите новую степень многочлена  4  2 пункт)  Выводим многочлены  1) +2Х^0 +2Х^1 +2Х^2  2) +2Х^0 +2Х^1 +2Х^2 +0Х^3  6 пунтк)  Выводим многочлены  1) +2Х^0 +2Х^1 +2Х^2  2) +2Х^0 +2Х^1 +2Х^2 +0Х^3  Введите номер многочлена: 2  Введите значение Х: 1  Значение многочлена при заданном х = 6  0 пункт) Для продолжения нажмите любую клавишу . . . | 1 пункт)  Добавляем 1 многочлен  Введите степень многочлена: 3  Введите коэффициенты при Х^0: 1  Введите коэффициенты при Х^1: 1  Введите коэффициенты при Х^2: 1  1 пункт)  Добавляем 2 многочлен  Введите степень многочлена: 3  Введите коэффициенты при Х^0: 2  Введите коэффициенты при Х^1: 2  Введите коэффициенты при Х^2: 2  2 пункт)  Выводим многочлены  1) +1Х^0 +1Х^1 +1Х^2  2) +2Х^0 +2Х^1 +2Х^2  3 пункт)  1) +1Х^0 +1Х^1 +1Х^2  2) +2Х^0 +2Х^1 +2Х^2  Введите номера многочленов которые хотите сложить  (в качесвте 1 номера укажите многочлен к которому желаете прибавить 2-ой многочлен): 1  2  2 пункт)  Выводим многочлены  1) +3Х^0 +3Х^1 +3Х^2  2) +2Х^0 +2Х^1 +2Х^2  4 пункт)  1) +3Х^0 +3Х^1 +3Х^2  2) +2Х^0 +2Х^1 +2Х^2  Введите номера многочленов для приравнивания их друг к другу(1 номер - многочлен, чьи коэфф. сравняются с коэф. 2 номер): 1  2  2 пункт)  Выводим многочлены  1) +2Х^0 +2Х^1 +2Х^2  2) +2Х^0 +2Х^1 +2Х^2  5 пункт)  Выводим многочлены  1) +2Х^0 +2Х^1 +2Х^2  2) +2Х^0 +2Х^1 +2Х^2  Введите номер многочлена: 2  Введите новую степень многочлена  4  2 пункт)  Выводим многочлены  1) +2Х^0 +2Х^1 +2Х^2  2) +2Х^0 +2Х^1 +2Х^2 +0Х^3  6 пунтк)  Выводим многочлены  1) +2Х^0 +2Х^1 +2Х^2  2) +2Х^0 +2Х^1 +2Х^2 +0Х^3  Введите номер многочлена: 2  Введите значение Х: 1  Значение многочлена при заданном х = 6  0 пункт) Для продолжения нажмите любую клавишу . . . | **+** |

**Заголовочный файл 5.3.1**

#pragma once

#include <iostream>

#include <vector>

#include "math.h"

using namespace std;

class Polinom

{

private:

double polinom[20];//коэффициенты многочлена

short N;//степень многочлена

void setKoef();

public:

Polinom();

~Polinom();

friend ostream& operator<<(ostream& os, const Polinom& dt);//выводит многочлен

friend void operator+( Polinom& arr1, Polinom& arr2);//прибавляет к 1 многчлену 2-ой многочлен

Polinom& operator=(Polinom& arr1);//присваивает многочлен многочлену

friend istream& operator>>(istream& is, Polinom& dt);//изменяем степень многочлена

void calculate(Polinom& arr1);//вычисляет значение многочлена при заданном Х

};

**Файл с реализацией некоторых методов 5.3.2**

#include "Polinom.h"

void Polinom::setKoef()

{

cout << "Введите степень многочлена: ";

cin >> N;//возможно здесь будет ошибка

if (N > 20)N = 20;

for (int i = 0; i < 20; i++)

{

if (i < N)

{

cout << "Введите коэффициенты при Х^" << i << ": "; cin >> polinom[i];

}

else {//пустую часть массива заполняем 0

polinom[i] = 0;

}

}

}

Polinom::Polinom()

{

setKoef();

}

Polinom::~Polinom()

{

}

Polinom & Polinom::operator=(Polinom & arr1)

{

for (int i = 0; i < 20; i++)//пробегаемся по коэффициентам многочленов

{

polinom[i] = arr1.polinom[i];

}

if (N < arr1.N) { N = arr1.N; }//если степень 2 многочлена больше чем у 1-ого, то при сложении степень 1-ого увеличится

return \*this;//возвращаем в правую часть операции присваивания

}

void Polinom::calculate(Polinom & arr1)

{

double y=0;//значение многочлена

double x;

cout << "Введите значение Х: "; cin >> x;

for (int i = 0; i < 20; i++)

{

y += arr1.polinom[i]\*pow(x, i);

}

cout << "\n\nЗначение многочлена при заданном х = " << y << endl;

}

ostream & operator<<(ostream & os, const Polinom & dt)

{

for (int i = 0; i < dt.N; i++)//пробегаем по всем коэффициентам многочлена

{

if (dt.polinom[i] >= 0)

{

os << "+" << dt.polinom[i] << "Х^" << i << " ";

}

else

{

os << dt.polinom[i] << "Х^" << i << " ";

}

}

os << endl;

return os;

}

void operator+(Polinom & arr1, Polinom & arr2)

{

for (int i = 0; i < 20; i++)//пробегаемся по коэффициентам многочленов

{

arr1.polinom[i] += arr2.polinom[i];

}

if (arr1.N < arr2.N) { arr1.N = arr2.N; }//если степень 2 многочлена больше чем у 1-ого, то при сложении степень 1-ого увеличится

}

istream & operator>>(istream & is, Polinom & dt)

{

cout << "Введите новую степень многочлена\n";

is >> dt.N;

return is;

}

**5.3.3 Основная программа**

#include <iostream>

#include "Polinom.h"

#include <Windows.h>

#include <vector>

using namespace std;

bool verification(int num1, int num2, short k);//проверка на корректность веденных номеров многочленнов

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

short answer2 = 10;

vector<Polinom>arr;//вектор многочленов

short k=0;//кол-во многочленов

while (answer2 != 0) {

system("cls");

printf("Лабораторная работа №2 Бикеева А. И. Вариант 1\n\n");

printf("Задание 1\nОпределить степень многочленаВычислить значение многочлена при заданном х\n");

printf("\nМеню\n");

printf("1) Создать многочлен N-ой степени\n");

printf("2) Вывести все многочлены\n");

printf("3) Сложить многочлены\n");

printf("4) Приравнять коэффициенты указанных многочленах\n");

printf("5) Определить новую степень выбранного многочлена\n");

printf("6) Вычислить значение выбранного многочлена при заданном х\n");

printf("0) Выход\n");

printf("\nВаш выбор: ");

while ((!(cin >> answer2)) || cin.get() != '\n') { //делаем проверку на ввод букв

cout << "ERROR\n";

cin.clear(); // сбрасывает все биты потока, тоесть поток становится "хорошим"

cin.sync();//Удалим данные из буффера

cout << "Повторите ввод : ";

}

system("cls");

printf("Лабораторная работа №2 Бикеева А. И. Вариант 1\n\n");

switch (answer2)

{

case 0: {system("pause");

break;

}

case 1: {

cout << "Добавляем " << k + 1 << " многочлен\n\n";

Polinom obj;

arr.push\_back(obj);

k++;

obj.~Polinom();//удаляем объект, т.к. он уже хранится в векторе

system("pause");

break;

}

case 2: {

if (k)

{

cout << "Выводим многочлены\n\n";

for (int i = 0; i < k; i++)

{

cout << i + 1 << ") ";

cout<< arr[i];//выводи вектор используя перегрузку

}

}

else {

cout << "Для начала задайте хотябы 1 многочлен\n";

}

system("pause");

break;

}

case 3: {

if (k)

{

for (int i = 0; i < k; i++)

{

cout << i + 1 << ") ";

cout << arr[i];//выводи вектор используя перегрузку

}

cout << "Введите номера многочленов которые хотите сложить\n(в качесвте 1 номера укажите многочлен к которому желаете прибавить 2-ой многочлен): ";

int num1, num2;//номера многочленов

cin >> num1; cin >> num2;

num1--; num2--;//т.к. объекты многочленов нумируются с 0, то отличие от отоображения в списке == -1

if (verification(num1, num2, k)) { arr[num1] + arr[num2]; }//перегрузка операции сложения

else cout << "Введены номера не в соответствии списку\n\n";

}

else {

cout << "Для начала задайте хотябы 1 многочлен\n";

}

system("pause");

break;

}

case 4: {

if (k)

{

for (int i = 0; i < k; i++)

{

cout << i + 1 << ") ";

cout << arr[i];//выводи вектор используя перегрузку

}

cout << "Введите номера многочленов для приравнивания их друг к другу(1 номер - многочлен, чьи коэфф. сравняются с коэф. 2 номер): ";

int num1, num2;//номера многочленов

cin >> num1; cin >> num2;

num1--; num2--;//т.к. объекты многочленов нумируются с 0, то отличие от отоображения в списке == -1

if (verification(num1, num2, k))

{

Polinom copy1 = arr[num1], copy2 = arr[num2];

copy1 = copy2;//Используем перегрузку присваивания

arr[num1] = copy1;

copy1.~Polinom();//удаляем ненужные объекты

copy2.~Polinom();

}

else cout << "Введены номера не в соответствии списку\n\n";

}

else {

cout << "Для начала задайте хотябы 1 многочлен\n";

}

system("pause");

break;

}

case 5: {

if (k)

{

cout << "Выводим многочлены\n\n";

for (int i = 0; i < k; i++)

{

cout << i + 1 << ") ";

cout << arr[i];//выводи вектор используя перегрузку

}

cout << "Введите номер многочлена: ";

int num; cin >> num;

if (num > 0 && num <= k)

{

num--;

cin >> arr[num];//перегрузка

}

else {

cout << "Данного номера нет в списке\n\n";

}

}

else {

cout << "Для начала задайте хотябы 1 многочлен\n";

}

system("pause");

break;

}

case 6: {

if (k)

{

cout << "Выводим многочлены\n\n";

for (int i = 0; i < k; i++)

{

cout << i + 1 << ") ";

cout << arr[i];//выводи вектор используя перегрузку

}

cout << "Введите номер многочлена: ";

int num; cin >> num;

if (num > 0 && num <= k)

{

num--;

arr[num].calculate(arr[num]);

}

else {

cout << "Данного номера нет в списке\n\n";

}

}

else {

cout << "Для начала задайте хотябы 1 многочлен\n";

}

system("pause");

break;

}

default:

{

printf("Вы ввели не существующий номер вопроса\n");

system("pause");

}

break;

}

}

system("pause");

}

bool verification(int num1, int num2, short k)

{

if (num1 >= 0 && num2 >= 0 && num1 < k&&num2 < k) return true;//ввод корректен

return false;//ввод не корректен

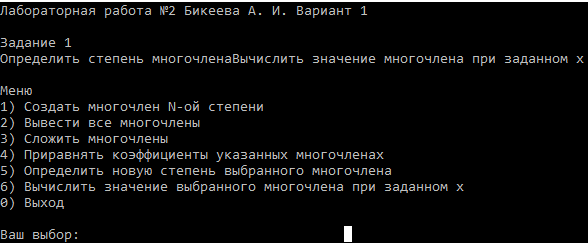
}

**Таблица вариантов использований**

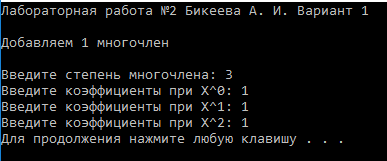
|  |  |
| --- | --- |
| Действие исполнителя | Отклик системы |
| 1. Пользователь инициирует новое задание 3. Пользователь выбирает тип задачи  5.  а)Пользователь ожидает  б)Пользователь вводит с клавиатуры значение, которое попросит его система  7.Пользователь ожидает  9. Пользователь , проанализировав результаты нажимает на одну из клавиш на клавиатуре | 2. Система регистрирует новое задание и предлагает список типов задач  4. Система регистрирует тип задачи. Если :  а)Задача предполагает вывод данных веденных ранее, то участие пользователя не требуется  б)Задача предполагает ввод каких либо данных для ее решения, то система будет ожидать ответа пользователя    6. Система проверяет полноту определения задания и запускает подпрограмму решения задачи  8. Система демонстрирует пользователю результаты и ожидает пользователя  10.Система возращается к пункту 2 |

**Отладка программы**

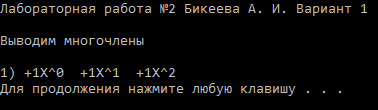
Запуск программы



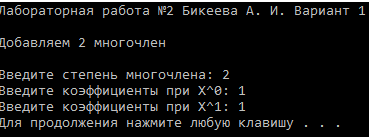
Выбор 1



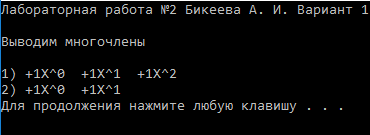
Выбор 2



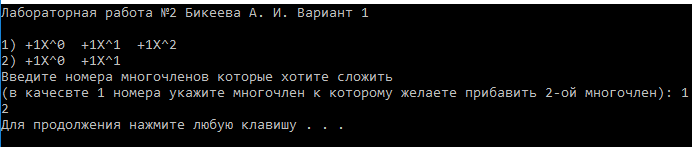
Выбор 1



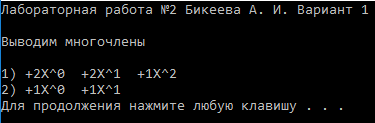
Выбор 2



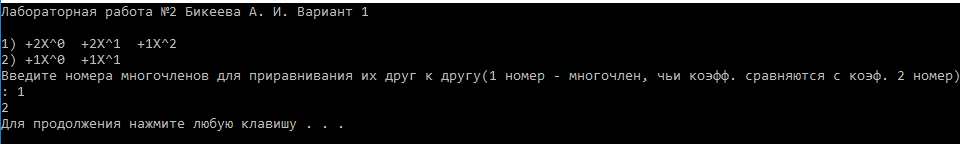
Выбор 3



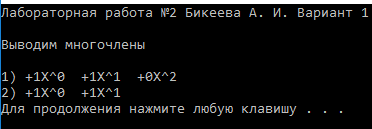
Выбор 2



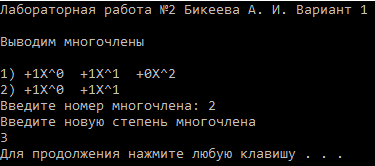
Выбор 4



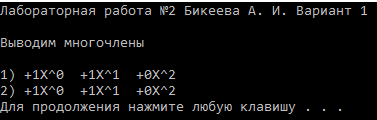
Выбор 2



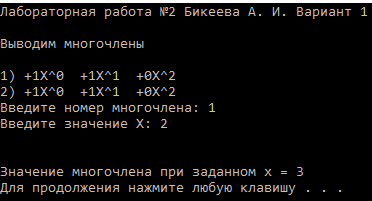
Выбор 5



Выбор 2



Выбор 6



Выбор 0

