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

Лабораторная работа №1

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 3](#_Toc22368212)

[1.1. Вариант задания 3](#_Toc22368213)

[1.2. Абстрактный тип данных (далее АТД) для варианта задания, включая список общих функций из задания. 3](#_Toc22368214)

[1.3. Реализация АТД 4](#_Toc22368215)

[1.4. Таблица Тестов 6](#_Toc22368216)

[1.5. Код основной программы с комментариями 8](#_Toc22368217)

[1.6. Скриншоты выполнения тестов 12](#_Toc22368218)

[2. Задание 2 14](#_Toc22368219)

[2.1. Структура узла 14](#_Toc22368220)

[2.2. Структура класса 14](#_Toc22368221)

[2.3. Таблица тестов 15](#_Toc22368222)

[2.4. Код основной программы с комментариями 17](#_Toc22368223)

[2.5. Скриншоты выполнения тестов 21](#_Toc22368224)

[**Заключение** 23](#_Toc22368225)

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);

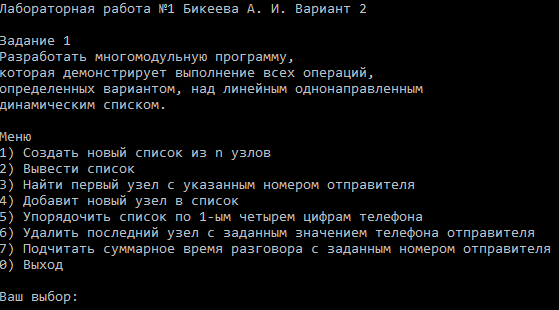
    }

    return 0;

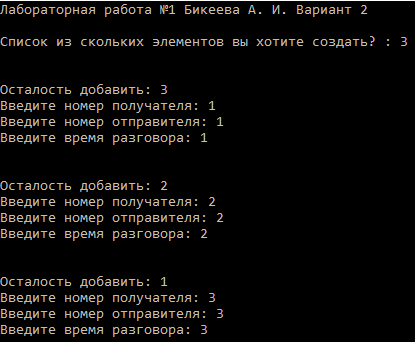
}

* 1. Скриншоты выполнения тестов

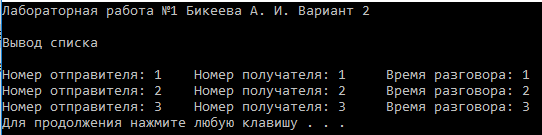
**Запуск программы**



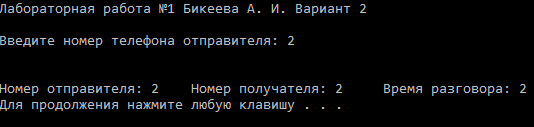
**1 пункт**



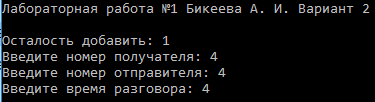
**2 пункт**



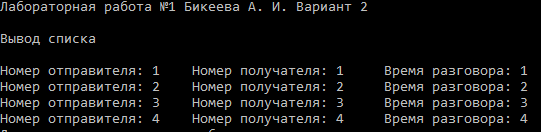
**3 пункт**



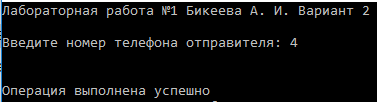
**4 пункт**



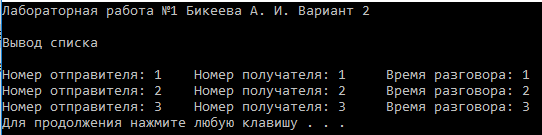
**2 пункт**



**6 пункт**



**2 пункт**



**0 пункт**



1. Задание 2
   1. Структура узла

struct Node

{

    //Номер отправителя

    int phoneFrom;

    //Номер получателя

    int phoneTo;

    //Время разговора в минутах

    int timeTell;

} Telephone;

* 1. Структура класса

class Telephone

{

private:

    Telephone\* next = nullptr;

public:

    Telephone();

    int getPhoneFrom();

    int getPhoneTo();

    int getTimeTell();

    Telephone\* getNext();

    void setPhoneFrom(int phoneFrom);

    void setPhoneTo(int phoneTo);

    void setTimeTell(int timeTell);

    void setNext(Telephone\* next);

};

* 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->getPhoneFrom() << "\tНомер получателя: " << finded->getPhoneTo() << "\tВремя разговора: " << finded->getTimeTell() << 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);

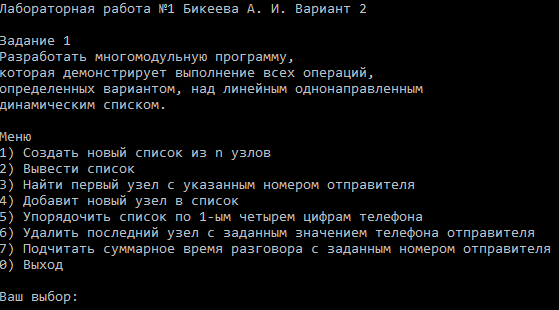
    }

    return 0;

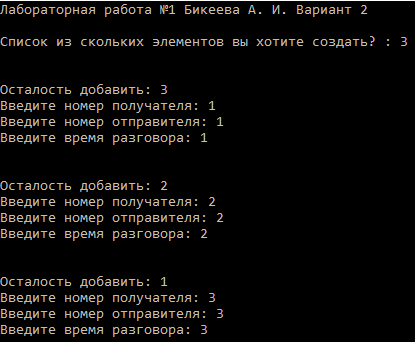
}

* 1. Скриншоты выполнения тестов

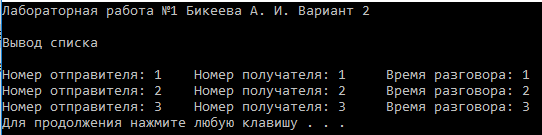
**Запуск программы**



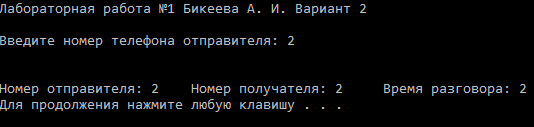
**1 пункт**



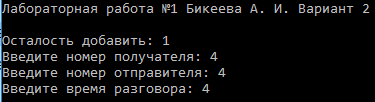
**2 пункт**



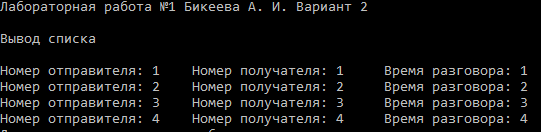
**3 пункт**



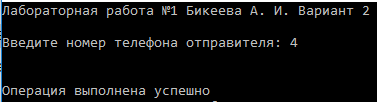
**4 пункт**



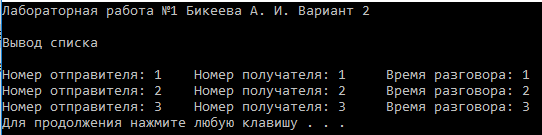
**2 пункт**



**6 пункт**



**2 пункт**



**0 пункт**



**Заключение**

В данной лабораторной работе я получил навыки по обработке линейного динамического списка, изучил как определять структуру узла списка с помощью спецификации **typedef**.