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

**Отчет по лабораторной работе**

|  |  |  |
| --- | --- | --- |
| **Структуры и алгоритмы обработки данных** | | |
| *( наименование дисциплины )* | | |
| Направление подготовки | | 09.03.01 Информатика и вычислительная техника |
|  | |  |
| Тема | Линейные списки | |
| Институт | информационных технологий (ИТ) | |

Выполнил студент группы ИВБО-06-17 Бикеев А. И.

Принял преподаватель Скворцова Л.А.

Москва 2019

**Содержание**

[**1.** **Задание 1** 3](#_Toc26127749)

[**1.1.** **Вариант задания** 3](#_Toc26127750)

[**1.2.** **Абстрактный тип данных (далее АТД) для варианта задания, включая список общих функций из задания.** 3](#_Toc26127751)

[**1.3.** **Реализация АТД** 4](#_Toc26127752)

[ **Определение типа информационной части узла (typedef)** 4](#_Toc26127753)

[ **Структура узла** 5](#_Toc26127754)

[ **Список заголовков методов с учетом реализованных типов** 5](#_Toc26127755)

[ **Список модулей реализации АТД (или описать, где расположена реализация АТД)** 6](#_Toc26127756)

[**1.4.** **Таблица тестов** 6](#_Toc26127757)

[**1.5.** **Код основной программы с комментариями** 10](#_Toc26127758)

[**2.** **Задание** 15](#_Toc26127759)

[**2.1.** **Структура класса «узел»** 15](#_Toc26127760)

[**2.2.** **Структура класса «список»** 15](#_Toc26127761)

[**2.3.** **Код основной программы с комментариями** 19](#_Toc26127762)

[**2.4.** **Скриншоты выполнения тестов** 23](#_Toc26127763)

[**Вывод** 25](#_Toc26127764)

# **Задание 1**

# **Вариант задания**

Разработать структуру узла списка, структура информационной части узла: Номер телефона (из 7 цифр), время разговора (целое число), номер телефона вызываемого абонента..

Определение структуры узла списка выполните через спецификацию typedef в отдельном заголовочном файле.

Разработайте функции для выполнения операции над линейным списком:

* 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

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.

Разработайте программу, управляемую текстовым меню, согласно заданию и включите в меню демонстрацию выполнения всех операций задания и варианта.

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

***Абстрактные данные:***

Telephone – структура узла данных номера счета в банке, у него должны быть следующие поля:

* Номер отправителя (Целочисленное);
* Номер получателя (Целочисленное);
* Тип операции (Целочисленное);
* Время разговора в минутах (Целочисленное);
* Указатель на следующий элемент (Node).

***Операции:***

Создает список из n узлов

Node\* createList(Node\* head, int n);

Выводит на экран список list из n узлов

void showList(Node\* list);

Находит узел с заданынм номером отправтеля

Node\* findNode(Node\* current, int phoneFrom);

Возвращает указатель на последний элемент списка

Node\* getLastNode(Node\* current);

Добавляет в конец списка новый узел

void addNode(Node\* lastNode, Node\* nodeNew);

Возвращает первые 4 цифры телефона

int getNum(int phone);

Меняет значения элементов списка

void swapDataNode(Node\* node1, Node\* node2);

Сортирует список по первым четырем цифрам телефона

void sortList(Node\* head);

Возвращает ссылку на последний узел, содержащий соответствующий номер отправителя

Node\* getLastNodeForDelete(Node\* head, int phoneFrom);

Удаляет выбранный узел

void deletChoosenNode(Node\* head);

Возвращает суммарную длиельность разговора по указанному номеру отправителя

int sumTimeTell(Node\* head, int phoneFrom);

# **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;

# **Структура узла**

struct list

{

list() { head = NULL; tail = NULL; };

Telephone \* head; //первый элемент списка

Telephone \* tail; //последнй элемент списка

};

# **Список заголовков методов с учетом реализованных типов**

Node\* createList(Node\* head, int n);

void showList(Node\* list);

Node\* findNode(Node\* current, int phoneFrom);

Node\* getLastNode(Node\* current);

void addNode(Node\* lastNode, Node\* nodeNew);

int getNum(int phone);

void swapDataNode(Node\* node1, Node\* node2);

void sortList(Node\* head);

Node\* getLastNodeForDelete(Node\* head, int phoneFrom);

void deletChoosenNode(Node\* head);

int sumTimeTell(Node\* head, int phoneFrom);

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

Программа состоит из пяти модулей: Lab1\_1.cpp, где содержится основная функция main(), Telephone.h, которая содержит описание информационной части структуры, Operation.cpp лежат объявления вспомогательных функций и Operation.cpp, где лежат их реализация.

# **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 | **+** |

# **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)

    {

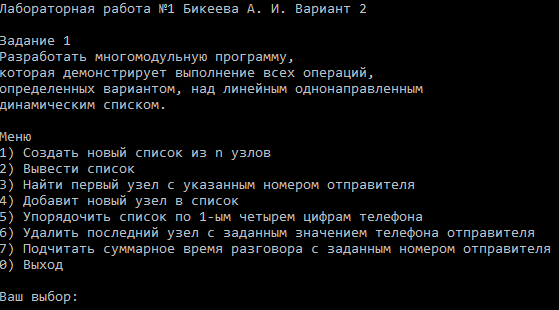
        free(list);

    }

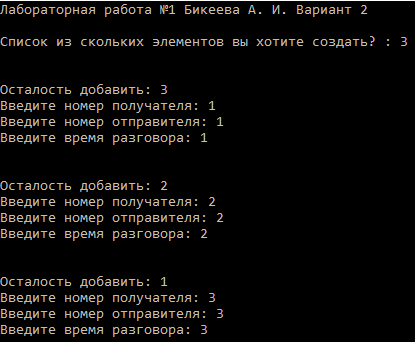
    return 0;

}

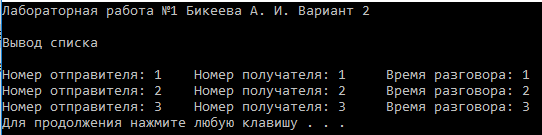
**Скриншоты выполнения тестов**



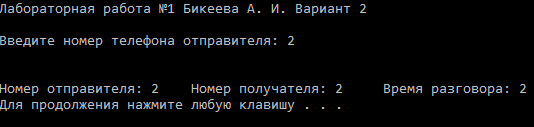
**1 пункт**



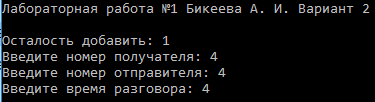
**2 пункт**



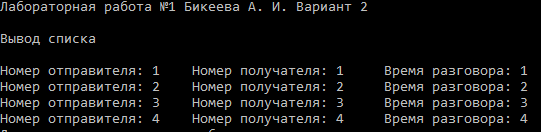
**3 пункт**



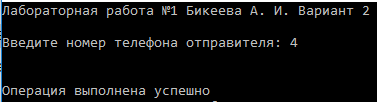
**4 пункт**



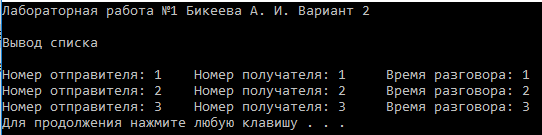
**2 пункт**



**6 пункт**



**2 пункт**



**0 пункт**



# **Задание**

Реализация списковой структуры с использованием ООП

# **Структура класса «узел»**

Класс «Узел» обладает следующими свойствами:

int phoneFrom – номер отправителя

int phoneTo – номер получателя

int timeTell – время разговора в минутах

Дополнительные функции:

# **Структура класса «список»**

Класс «Список» обладает следующими свойствами:

int getPhoneFrom();

int getPhoneTo();

int getTimeTell();

Telephone\* getNext();

void setPhoneFrom(int phoneFrom);

void setPhoneTo(int phoneTo);

void setTimeTell(int timeTell);

void setNext(Telephone\* next);

**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 | **+** |

# **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);

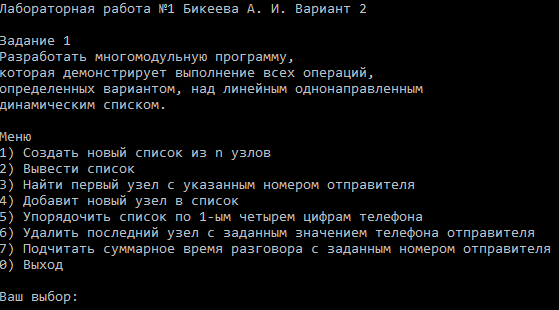
    }

    return 0;

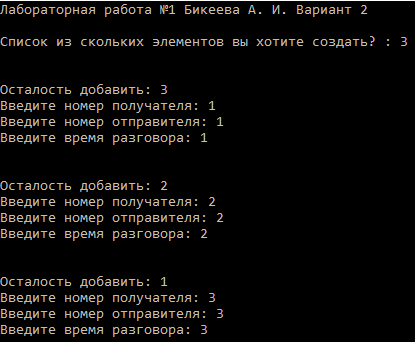
}

# **Скриншоты выполнения тестов**

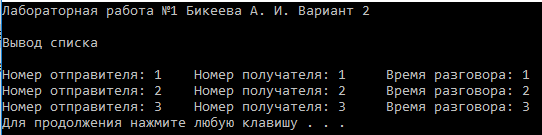
**Запуск программы**



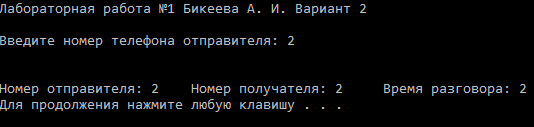
**1 пункт**



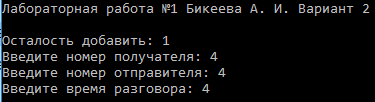
**2 пункт**



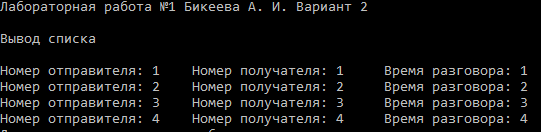
**3 пункт**



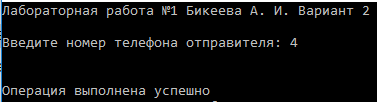
**4 пункт**



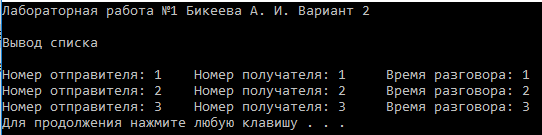
**2 пункт**



**6 пункт**



**2 пункт**



**0 пункт**



# **Вывод**

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