МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 5

по дисциплине

«Структуры и алгоритмы обработки данных»

Тема: «Однонаправленный список»

Выполнил студент группы ИКБО-18-22 Ракитин В.А.

Принял преподаватель Филатов А.С.

Лабораторная работа выполнена «__»_____202__ г. *(подпись студента)*

«Зачтено» «__»_____202__ г. *(подпись руководителя)*

Москва 2023

1. Цель работы

Получение навыков в разработке алгоритмов. Получение знаний и практических навыков управления динамическим однонаправленным списком.

2. Постановка задачи

1. Разработать функцию для создания исходного списка, его вывода и добавления узла.
2. Информационная часть узла списка определена вариантом.
3. Разработать функции дополнительного задания варианта.
4. В основной программе выполнить тестирование каждой функции, описанной в задании.
5. Составить отчет по выполненному заданию. В отчет включить ответы на вопросы к практической работе.

Вариант №4. Условие задания:

Упражнение	Дан линейный однонаправленный список L1 1) Разработать функцию, которая переформирует список L1, переписав в начало списка его часть, начиная с заданной позиции (номер позиции передается в функцию). 2) Разработать функцию вставки узла в упорядоченный по не возрастанию список. Сформировать такой список L2. 3) Разработать функцию, которая удаляет из L2 все повторяющиеся значения, оставляя одно из них.

3. Решение

Односвязный список - это динамическая структура данных, состоящая из узлов. Каждый узел будет иметь какое-то значение и указатель на следующий

узел.

Односвязный список, в отличие от динамического массива, может хранить элементы в разных областях памяти. В них начало списка — это головной элемент, звенья — это узлы, а конец списка определяется посредством NULL — специального узла. Причём, чтобы структура списка была полезной, на каждый узел «вешают» определённое значение.

У односвязного списка есть преимущество — вставка и удаление узлов осуществляется довольно легко **в любом месте списка**. Однако список нельзя индексировать в качестве массива, а структура списка ограничивает доступ к узлам по индексу. Если нужно попасть на какой-нибудь узел односвязного списка, нужно пройти весь путь последовательно, начиная от головного элемента, заканчивая нужным узлом.

Для решения первой задачи была написана функция reform, которая получает на вход целое число – индекс элемента односвязного списка. Эта функция переписывает в начало списка часть списка, начинающаяся с полученного на входе функции индекса.

```
void reform(int index) {
    // найти элемент, предшествующий заданной позиции
    Node* prevNode = head;
    for (int i = 1; i < index && prevNode != nullptr; i++) {
        prevNode = prevNode->next;
    }
    if (prevNode == nullptr) {
        return; // позиция больше длины списка
    }
    // переместить часть списка в начало
    Node* newHead = prevNode->next;
    if (newHead == nullptr) {
        return; // нет элементов после заданной позиции
    }
    prevNode->next = nullptr; // обрезать список после prevNode
    Node* lastNode = newHead;
    while (lastNode->next != nullptr) {
        lastNode = lastNode->next;
    }
```

```
    lastNode->next = head; // соединить перемещенную часть со старым началом
    head = newHead; // новое начало списка
}
```

Для решения второй задачи была написана функция push_bask, которая получает на входе число, которое будет стоять в нашем односвязном списке.

```
void push_back(int data) {
    Node* newNode = new Node;
    newNode->data = data;
    newNode->next = nullptr;
    if (head == nullptr){
        head = newNode;
        }
    else {
        Node* current = head;
        while (current->next != nullptr){
            current = current->next;
        }
        current->next = newNode;
    }
}
```

Для решения третьей задачи была написана функция deleteRepeatingElements, которая удаляет из списка одинаковые элементы, оставляя только одно.

```
void deleteRepeatingElements() {
    unordered_set<int> uniqueValues; // Создание хеш-таблицы
    Node* current = head;
    Node* prev = nullptr;

    while (current != nullptr) {
        if (uniqueValues.find(current->data) != uniqueValues.end()) {
            prev->next = current->next;
            delete current;
            current = prev->next;
        }
        else {
            uniqueValues.insert(current->data);
            prev = current;
            current = current->next;
        }
    }

}
```

При запуске программы пользователь видит пользовательское меню, где пользователю надо выбрать, какую задачу нужно решить.

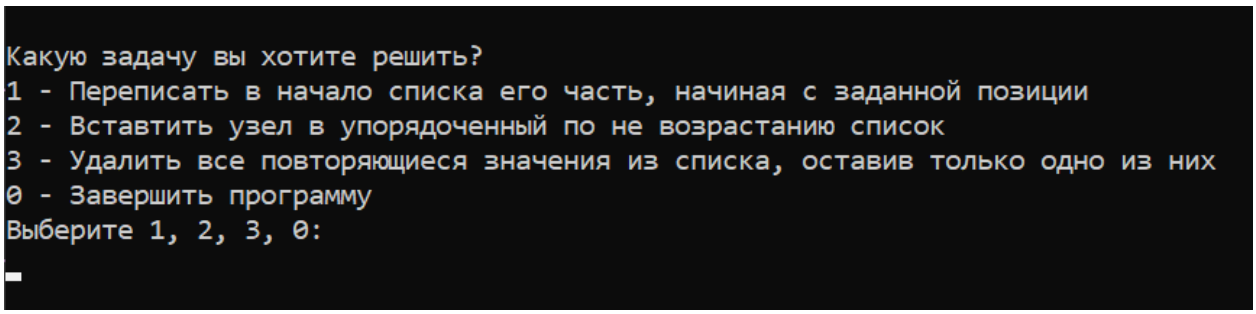

Рисунок 1. Интерфейс программы

После выбора задачи программа предложит пользователю ввести количество элементов списка. Затем программа предложит выбор между ручным и автоматическим вводом.

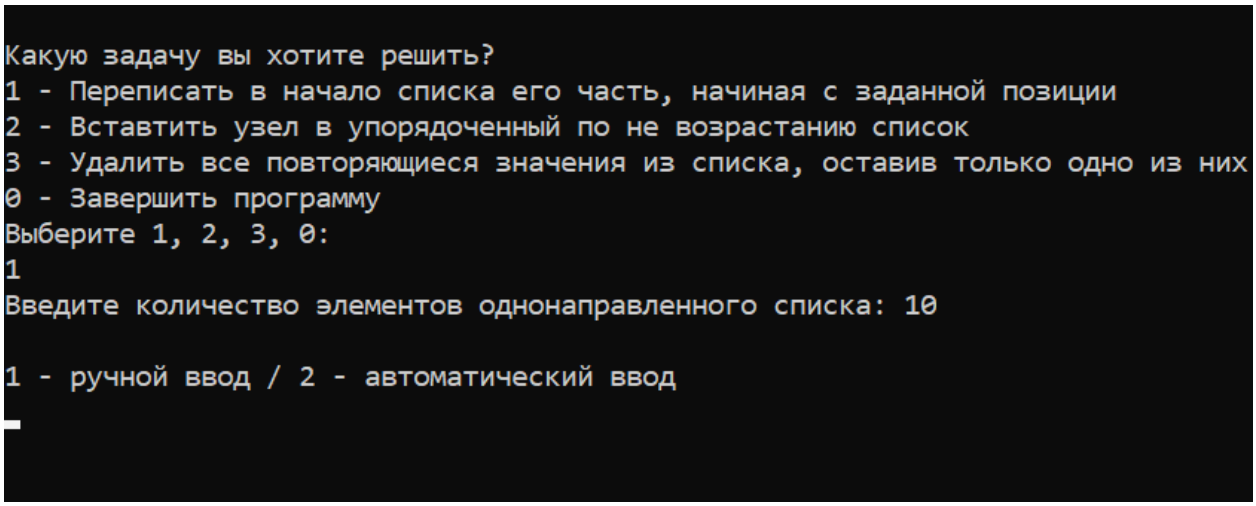

Рисунок 2. Интерфейс программы

4. Тестирование

Протестируем программой выполнение первого упражнения. Ручным вводом введём 8 чисел: 1, 2, 3, 4, 5, 6, 7, 8. Далее введём цифру 4, т.е. программа должна, начиная с 4-го элемента (в нашем случае 4-й элемент равен 5), переставить часть списка в начало до 4-го элемента. Ответ должен получиться 5, 6, 7, 8, 1, 2, 3, 4. На рисунке 3 программа дала верный результат.

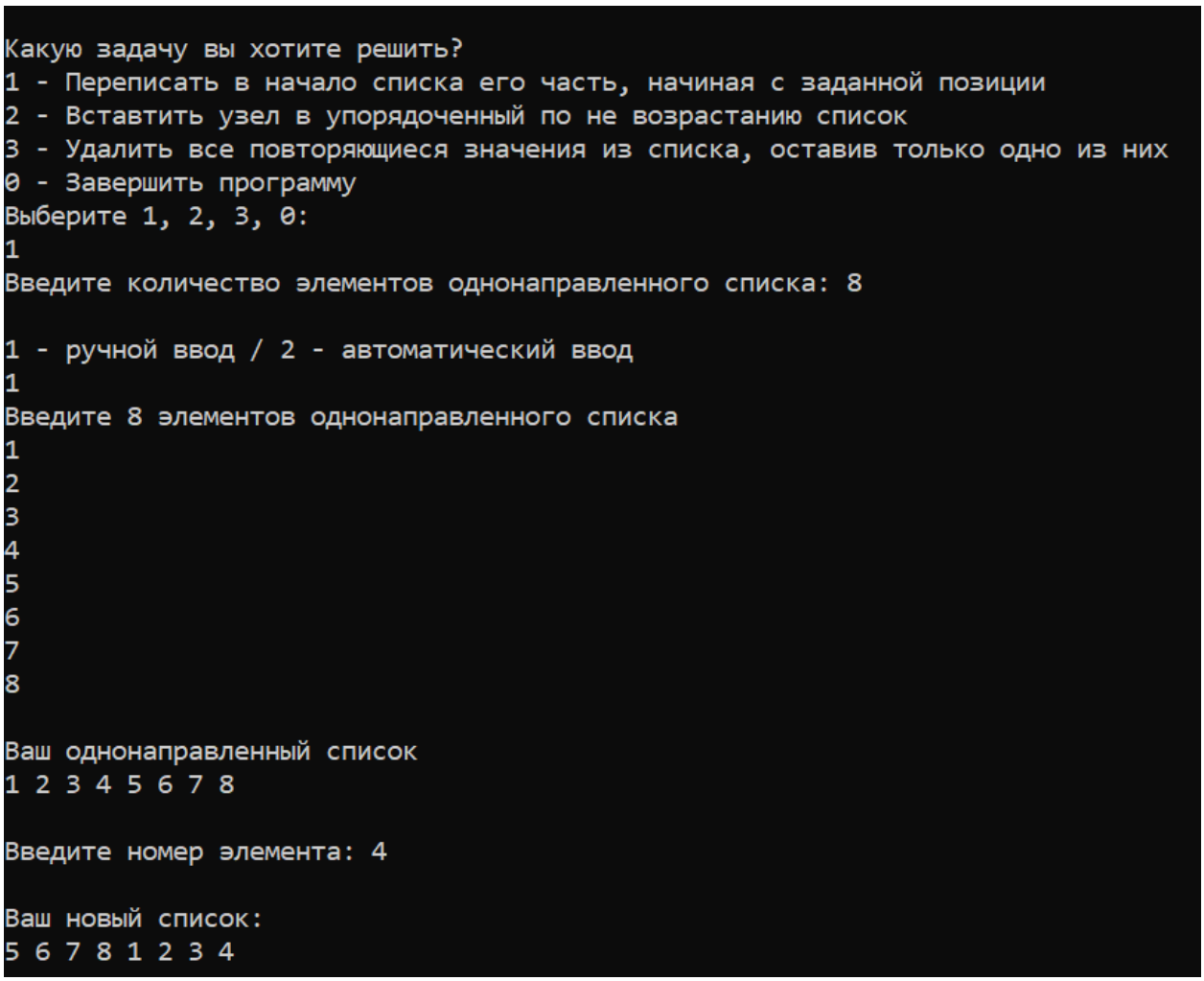

Рисунок 3. Решение программой первого упражнения

Протестируем программой выполнение второго упражнения. Введём число 2, т.е. программа будет решать вторую задачу. Автоматическим вводом программа введёт 8 значений. В нашем случае получились числа 41, 67, 34, 0, 69, 24, 78, 58. Далее программа выведет наш отсортированный не по возрастанию список и предложит ввести число, которое будет стоять в списке. Введём число 37. Наш новый список должен выглядеть так: 78, 69, 67, 58, 41, 37, 34, 24, 0. На рисунке 4 программа вывела верный результат.

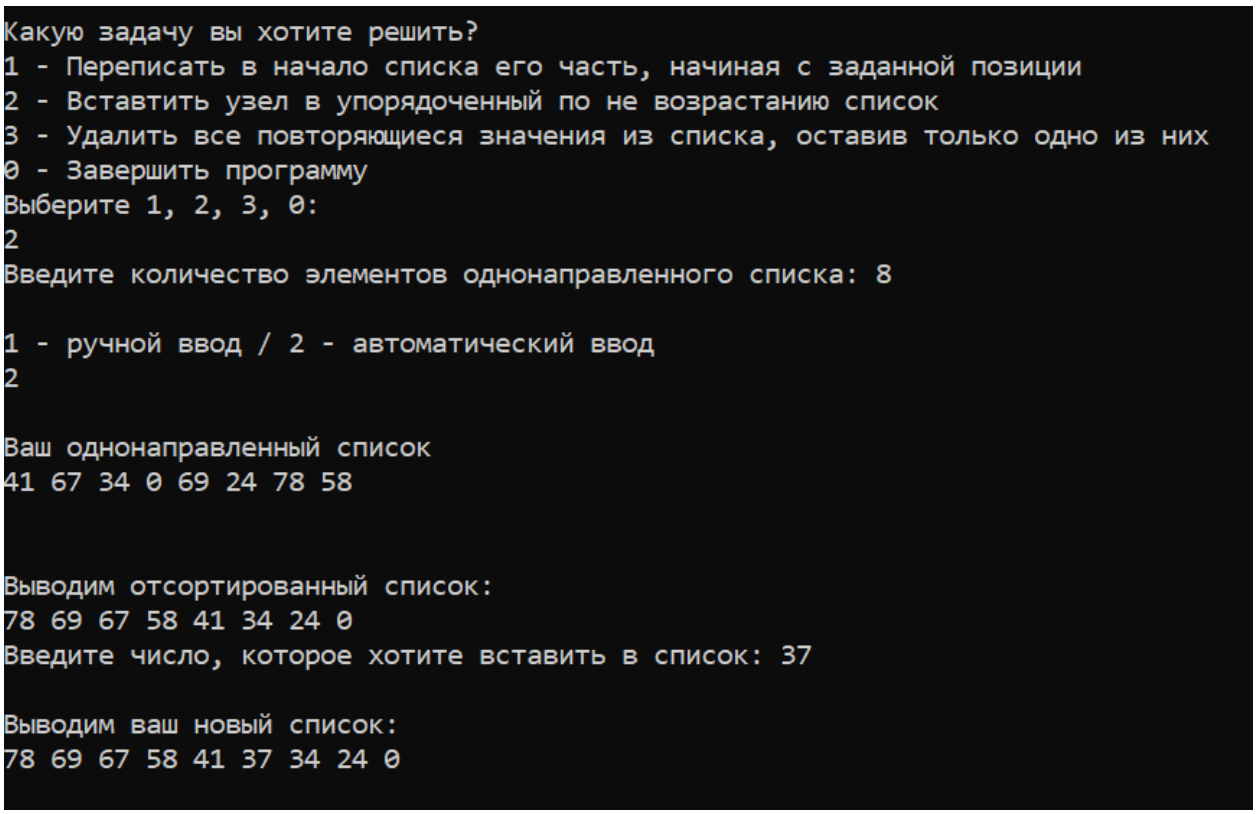

Рисунок 4. Решение программой второго упражнения

Протестируем программой выполнение третьего упражнения. Введём цифру 3, чтобы программа решала 3-ю задачу. Ручным вводом введём 10 значений: 9, 1, 1, 3, 6, 3, 1, 9, 5, 3. Далее программа выведет наш отсортированный не по возрастанию список, а затем выведет список, в котором не будет повторяющихся элементов. В нашем случае программа должна вывести 9, 1, 3, 6, 5. На рисунке 5 программа вывела верный результат.

```
Какую задачу вы хотите решить?
1 - Переписать в начало списка его часть, начиная с заданной позиции
2 - Вставтить узел в упорядоченный по не возрастанию список
3 - Удалить все повторяющиеся значения из списка, оставив только одно из них
0 - Завершить программу
Выберите 1, 2, 3, 0:
3
Введите количество элементов однонаправленного списка: 10

1 - ручной ввод / 2 - автоматический ввод
1
Введите 10 элементов однонаправленного списка
9
1
1
3
6
3
1
9
5
3

Ваш однонаправленный список
9 1 1 3 6 3 1 9 5 3

Выводим ваш новый список:
9 1 3 6 5
```

Рисунок 5. Решение программой третьего упражнения

5. Вывод

В результате выполнения работы я:

1. Получил знания и навыки реализации и применения односвязного списка
2. Получил знания по применению указателя ->

6. Исходный код программы

```
#include <iostream> // Библиотека для ввода/вывода в консоль
#include <unordered_set>
using namespace std; // Пространство имен std

struct Node {
        int data;
    Node* next;
};

struct L1 {
    Node* head;
```

```
    L1() {
        head = nullptr;
    }

    L1(int data) {
        head = nullptr;
        insert(data);
    }

    void push_back(int data) {      // Добавить элемент в конец
        Node* newNode = new Node;
        newNode->data = data;
        newNode->next = nullptr;

        if (head == nullptr) {
            head = newNode;
        }
        else {
            Node* current = head;
            while (current->next != nullptr) {
                current = current->next;
            }
            current->next = newNode;
        }
    }

    int insert(int index, int data) {  // Добавить элемент по индексу
        if (index == 0) {
            insert(data);
            return 0;
        }

        Node* newNode = new Node;
        newNode->data = data;

        if (head == nullptr) {
            newNode->next = nullptr;
            head = newNode;
        }
        else {
            Node* current = head;
            int i = 0;
            while (current != nullptr) {
                if (i == index - 1) {
                    newNode->next = current->next;
                    current->next = newNode;
                    return 0;
                }
                current = current->next;
                i += 1;
                if (i == index) {
                    push_back(data);
                    return 0;
                }
            }
        }

        return -1;  // Код возврата -1, если вставка не произошла

    }

    void insert(int data) {     // Добавить элемент в начало
        Node* newNode = new Node;
        newNode->data = data;
```

```
        if (head == nullptr) {
            newNode->next = nullptr;
            head = newNode;
        }
        else {
            Node* current = head;
            newNode->next = current;
            head = newNode;
        }
    }

    void reform(int index) {
        // найти элемент, предшествующий заданной позиции
        Node* prevNode = head;
        for (int i = 1; i < index && prevNode != nullptr; i++) {
            prevNode = prevNode->next;
        }
        if (prevNode == nullptr) {
            return; // позиция больше длины списка
        }

        // переместить часть списка в начало
        Node* newHead = prevNode->next;
        if (newHead == nullptr) {
            return; // нет элементов после заданной позиции
        }
        prevNode->next = nullptr; // обрезать список после prevNode
        Node* lastNode = newHead;
        while (lastNode->next != nullptr) {
            lastNode = lastNode->next;
        }
        lastNode->next = head; // соединить перемещенную часть со старым
началом
        head = newHead; // новое начало списка
    }

    void show() const {    // Выводит список на экран
        Node* current = head;

        while (current != nullptr) {
            cout << current->data << " ";
            current = current->next;
        }
        cout << endl;
    }
};
struct L2 : L1 {    // Наследуем от L1 основные поля и методы
public:
    using L1::L1;
    void sortedInsert(int data) {  // Вставка в отсортированный список
        Node* newNode = new Node;
        newNode->data = data;

        if (head == nullptr) {
            newNode->next = nullptr;
            head = newNode;
        }
        else {
            int index = 0;
            Node* current = head;
            while (current != nullptr) {
                if (current->data < data) {
                    insert(index, data);
```

```
                    return;
                }
                current = current->next;
                index += 1;
            }
            push_back(data);
        }
    }
    void deleteRepeatingElements() {
        unordered_set<int> uniqueValues; // Создание хеш-таблицы
        Node* current = head;
        Node* prev = nullptr;

        while (current != nullptr) {
            if (uniqueValues.find(current->data) != uniqueValues.end()) {
                prev->next = current->next;
                delete current;
                current = prev->next;
            }
            else {
                uniqueValues.insert(current->data);
                prev = current;
                current = current->next;
            }
        }
    }

};

int main() { // Главная функция программы
    setlocale(LC_ALL, "Russian"); // Установка русского языка для вывода в
консоль
    int pointer = 1;
    while (pointer == 1) {
        int a, N, q, choise, c;
        cout << endl << "Какую задачу вы хотите решить?" << endl <<
            "1 - Переписать в начало списка его часть, начиная с заданной
позиции" << endl <<
            "2 - Вставтить узел в упорядоченный по не возрастанию список" <<
endl <<
            "3 - Удалить все повторяющиеся значения из списка, оставив
только одно из них" << endl <<
            "0 - Завершить программу" << endl <<
            "Выберите 1, 2, 3, 0: " << endl;
        cin >> a;
        L2 l1, l2;
        switch (a) {
        case(1):
            cout << "Введите количество элементов однонаправленного списка:
"; cin >> N;
            cout << endl << "1 - ручной ввод / 2 - автоматический ввод " <<
endl;
            cin >> choise;
            switch (choise)
            {
            case(1):
                cout << "Введите " << N << " элементов однонаправленного
списка" << endl;
                for (int i = 0; i < N; i++) {
                    cin >> c;
                    l1.push_back(c);
                    l2.sortedInsert(c);
                }
                break;
```

```
            case(2):
                for (int i = 0; i < N; i++) {
                    c = rand() % 100;
                    l1.push_back(c);
                    l2.sortedInsert(c);
                }
                break;
            default:
                break;
            }
            cout << endl << "Ваш однонаправленный список" << endl;
            l1.show();
            cout << endl;
            int a1;
            cout << "Введите номер элемента: ";
            cin >> a1;
            l1.reform(a1);
            cout << endl << "Ваш новый список: " << endl;
            l1.show();
            break;
        case(2):
            cout << "Введите количество элементов однонаправленного списка:
"; cin >> N;
            cout << endl << "1 - ручной ввод / 2 - автоматический ввод " <<
endl;
            cin >> choise;
            switch (choise)
            {
            case(1):
                cout << "Введите " << N << " элементов однонаправленного
списка" << endl;
                for (int i = 0; i < N; i++) {
                    cin >> c;
                    l1.push_back(c);
                    l2.sortedInsert(c);
                }
                break;
            case(2):
                for (int i = 0; i < N; i++) {
                    c = rand() % 100;
                    l1.push_back(c);
                    l2.sortedInsert(c);
                }
                break;
            default:
                break;
            }
            cout << endl << "Ваш однонаправленный список" << endl;
            l1.show();
            cout << endl;
            cout << endl << "Выводим отсортированный список:" << endl;
            l2.show();
            int data, index;
            cout << "Введите число, которое хотите вставить в список: "; cin
>> data;
            l2.sortedInsert(data);
            cout << endl << "Выводим ваш новый список:" << endl;
            l2.show();
            break;
        case(3):
            cout << "Введите количество элементов однонаправленного списка:
"; cin >> N;
            cout << endl << "1 - ручной ввод / 2 - автоматический ввод " <<
endl;
```

```
            cin >> choise;
            switch (choise)
            {
            case(1):
                cout << "Введите " << N << " элементов однонаправленного
списка" << endl;
                for (int i = 0; i < N; i++) {
                    cin >> c;
                    l1.push_back(c);
                    l2.sortedInsert(c);
                }
                break;
            case(2):
                for (int i = 0; i < N; i++) {
                    c = rand() % 100;
                    l1.push_back(c);
                    l2.sortedInsert(c);
                }
                break;
            default:
                break;
            }
            cout << endl << "Ваш однонаправленный список" << endl;
            l1.show();
            cout << endl;
            l1.deleteRepeatingElements();
            cout << "Выводим ваш новый список:" << endl;
            l1.show();
            break;
        case(0):
            pointer = 0;
            break;
        default:
            break;
        }
    }
    return 0;
}
```

Таблица 1 – код программы