

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное образовательное учреждение высшего  
профессионального образования  
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО  
ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

ОТЧЕТ  
ЗАЩИЩЕН С ОЦЕНКОЙ  
РУКОВОДИТЕЛЬ

старший преподаватель  
должность, уч. степень звание

\_\_\_\_\_  
подпись, дата

С.А.Рогачёв  
инициалы, фамилия

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ №2

**«ЛИНЕЙНЫЕ И ЦИКЛИЧЕСКИЕ СПИСКИ»**

по дисциплине: СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. Z7431

22.01.2018

\_\_\_\_\_  
подпись, дата

М.Д.Семочкин

\_\_\_\_\_  
инициалы, фамилия

Санкт-Петербург  
2018

## 1. Цель работы

Целью работы является изучение структур данных «линейный список» и «циклический список», а также получение практических навыков их реализации.

## 2. Задание на лабораторную работу

Реализовать структуры данных «линейный список» и «циклический список» в соответствии с заданным вариантом. Дополнительно программа должна осуществлять следующие операции:

- 1) Добавление/удаление элемента в список (с клавиатуры);
- 2) Вывод исходного и результирующего списков на экран;
- 3) Если списки являются многочленами, в выводе должна быть отражена степень каждого элемента.

Согласно варианту 8,

Дана последовательность неповторяющихся целых чисел  $a_1, a_2, \dots, a_n$ , где  $n > 4$ . Получить последовательность, содержащую элементы исходной последовательности с удаленными двумя наименьшими и двумя наибольшими элементами.

$K = A - (\max(a), \min(a), \max(a - \max), \min(a - \min))$

Вид списка: Линейный двусвязный

## 3. Листинг программы

```
#include <stdlib.h>
#include <iostream>
#include <limits>
#include <cstring>
#include <ctype.h>
using namespace std;

struct List {           // связный список для хранения записей
    int data;
    List* next;
    List* previous;
};
```

```

//вывод всех элементов списка на экран
void ShowList(List *begin) {
    List * p = begin;
    while (p!=NULL) {
        cout << (p -> data) << " ";
        p = p -> next;
    }
    cout << endl;
}

// Добавление элемента в конец списка
void AddElem(List **begin, List ** cur, int elem) {

    // создаем новый элемент
    List * p = new List;
    p -> data = elem;

    // если список является пустым
    if (*begin == NULL) {
        p -> next = NULL;
        p -> previous = NULL;
        *begin = p;
        // Теперь в списке единственный элемент

    // если он не является пустым
    } else {
        p -> next = (*cur) -> next;
        p -> previous = *cur;
        (*cur) -> next = p;
    }

    *cur = p;
}

// Удаление элемента из списка
void DelElem(List **begin, List* ptr, List ** cur) {
    List * p;
    if (ptr == *begin) { // удаляем первый элемент
        *begin = (*begin) -> next;
        (*begin) -> previous = NULL;
    } else {
        // устанавливаем вспомогательный указатель на элемент,
        // предшествующий удаляемому
        p = ptr -> previous;
        // удаление элемента
        p -> next = ptr -> next;
        if (ptr -> next) {
            (ptr -> next) -> previous = p;
        }
    }
    // если это был последний элемент, записать в cur новый
    // последний элемент
    if (ptr -> next == NULL) {
        *cur = p;
    }
    delete ptr;
}

```

```

// поиск элемента в списке по значению
List * FindElem(List *begin, int value) {
    List * p = begin;
    while (p != NULL) {
        if (p -> data == value) {
            break;
        }
        p = p -> next;
    }
    return p;
}

// поиск элемента в списке по номеру
List * FindElemByIndex(List *begin, int index) {
    List * p = begin;
    int counter = 0;
    while (p != NULL) {
        if (counter == index) {
            break;
        }
        counter++;
        p = p -> next;
    }
    return p;
}

// создание результирующего списка
List * CreateResultList(List *begin) {

    // если исходный список еще не создан, вернуть NULL
    if (!begin) {
        return NULL;
    }

    // для результирующего списка
    List* headResult = NULL;
    List* curResult = NULL;

    List * p;
    int counter;

    // Найти индексы максимального и минимального элемента
    p = begin;
    counter = 0;
    int maxElem1Index = 0;
    int maxElem1Value = p -> data;
    int minElem1Index = 0;
    int minElem1Value = p -> data;
    while (p != NULL) {
        if ((p -> data) > maxElem1Value) {
            maxElem1Value = p -> data;
            maxElem1Index = counter;
        }
        if ((p -> data) < minElem1Value) {
            minElem1Value = p -> data;
            minElem1Index = counter;
        }
        p = p -> next;
        counter++;
    }
}

```

```

// и заодно проверить что список длиннее 4-х элементов
if (counter <= 4) {
    return NULL;
}

// Найти индексы вторых максимального и минимального элемента
p = begin;
counter = 0;
int maxElem2Index = 0;
int maxElem2Value = INT_MIN;
int minElem2Index = 0;
int minElem2Value = INT_MAX;
while (p != NULL) {
    // Если это максимальный или минимальный элемент, пропустить
    if (counter == maxElem1Index || counter == minElem1Index) {
        p = p -> next;
        counter++;
        continue;
    }
    if ((p -> data) > maxElem2Value) {
        maxElem2Value = p -> data;
        maxElem2Index = counter;
    }
    if ((p -> data) < minElem2Value) {
        minElem2Value = p -> data;
        minElem2Index = counter;
    }
    p = p -> next;
    counter++;
}

cout << "Два максимальных элемента: " << maxElem1Value <<
    " и " << maxElem2Value << endl;
cout << "Два минимальных элемента: " << minElem1Value <<
    " и " << minElem2Value << endl;

// Скопировать первый список во второй
//
// Не копировать элементы с индексами двух максимальных
// и двух минимальных
//
// Если минимальный элемент равен максимальному -
// все элементы в списке равны, просто не копировать
// первые 4 элемента
p = begin;
counter = 0;
if (maxElem1Index == minElem1Index) {
    while (counter < 4) {
        // первые 4 пропускаем
        counter++;
        p = p -> next;
    }
    while (p != NULL) {
        // остальные добавляем
        AddElem(&headResult, &curResult, p -> data);
        counter++;
        p = p -> next;
    }
} else {
    // добавляем все, пропускаем минимальные и максимальные
    while (p != NULL) {
        if (!(counter == minElem1Index ||

```

```

        counter == minElem2Index ||
        counter == maxElem1Index ||
        counter == maxElem2Index
    )) {
        AddElem(&headResult, &curResult, p -> data);
    }
    counter++;
    p = p -> next;
}

return headResult;

}

// Очистка памяти
void Free(List **begin) {
    if (*begin == 0) return;
    List *p = *begin;
    List *t;
    while (p) {
        t = p;
        p = p -> next;
        delete t;
    }
    *begin = NULL;
}

int main() {

    // Первый список
    List* head = NULL;
    List* cur = NULL;
    // Второй список
    List* head2 = NULL;

    setlocale(LC_ALL, "russian");

    int input;
    int n = -1;

    // Меню пользователя
    while (n != 0) {
        cout << endl <<
            "===== ДОСТУПНЫЕ КОМАНДЫ =====" << endl <<
            "1 - Добавить элемент в список" << endl <<
            "2 - Вывести исходный список" << endl <<
            "3 - Удалить элемент из списка по номеру" << endl <<
            "4 - Удалить элемент из списка по значению" << endl <<
            "5 - Получить результирующий список" << endl <<
            "6 - Вывести результирующий список" << endl <<
            "0 - Выход" << endl <<
            "===== " << endl <<
            "Выберите действие: ";
        cin >> n;
        cout << endl;
        while (cin.fail() || (n < 0) || (n > 6)) {
            cout << "Ошибка ввода, выберите действие: ";
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cin >> n;
        }
    }
}

```

```

switch (n) {
    case 1: {
        cout << "Введите элемент: ";
        cin >> input;
        while (cin.fail()) {
            cout << "Вы должны ввести целое число, повторите ввод: ";
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cin >> input;
        }
        AddElem(&head, &cur, input);
        cout << "Запись добавлена!" << endl;
        break;
    }
    case 2: {
        if (head) {
            cout << "Исходный список:" << endl;
            ShowList(head);
        } else {
            cout << "Список пуст!" << endl;
        }
        break;
    }
    case 3: {
        if (!head) {
            cout << "Список пуст!" << endl;
            break;
        }
        cout << "Введите номер элемента: ";
        cin >> input;
        List* ptr = FindElemByIndex(head, input);
        if (ptr == NULL) {
            cout << "Запись не найдена!" << endl;
        } else {
            DelElem(&head, ptr, &cur);
            cout << "Запись удалена!" << endl;
        }
        break;
    }
    case 4: {
        if (!head) {
            cout << "Список пуст!" << endl;
            break;
        }
        cout << "Введите значение элемента: ";
        cin >> input;
        List* ptr = FindElem(head, input);
        if (ptr == NULL) {
            cout << "Запись не найдена!" << endl;
        } else {
            DelElem(&head, ptr, &cur);
            cout << "Запись удалена!" << endl;
        }
        break;
    }
    case 5: {
        head2 = CreateResultList(head);
        if (head2) {
            cout << "Результирующий список создан!" << endl;
        } else {
            cout << "Не получилось создать результирующий список:"
                << endl << "Исходный список должен быть длинее"
                << " 4-х элементов" << endl;
        }
        break;
    }
    case 6: {

```

```

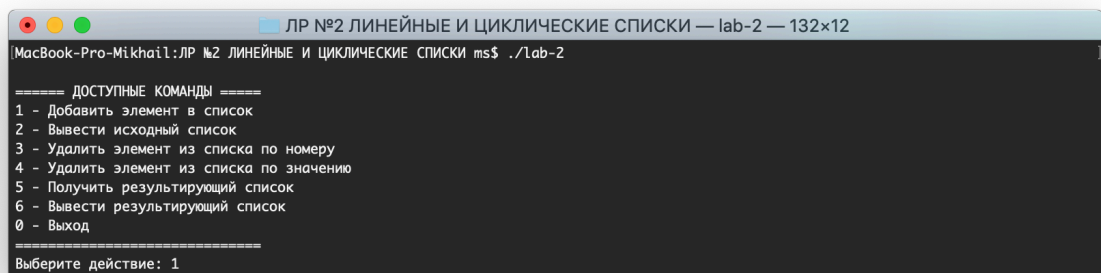
        if (head2) {
            cout << "Результирующий список:" << endl;
            ShowList(head2);
        } else {
            cout << "Результирующий список пуст!" << endl;
        }
        break;
    }
}
Free(&head);
Free(&head2);

return 0;
}

```

## 4. Пример работы программы

1) Программа имеет меню пользователя. Пользователь должен ввести нужную цифру, чтобы выполнить действие, или 0 чтобы выйти из программы.



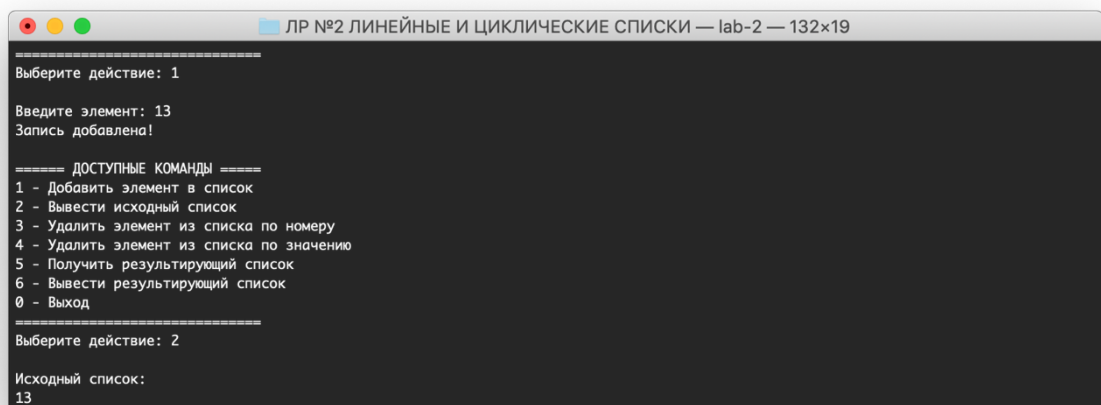
```

MacBook-Pro-Mikhail:ЛР №2 ЛИНЕЙНЫЕ И ЦИКЛИЧЕСКИЕ СПИСКИ ms$ ./lab-2

===== ДОСТУПНЫЕ КОМАНДЫ =====
1 - Добавить элемент в список
2 - Вывести исходный список
3 - Удалить элемент из списка по номеру
4 - Удалить элемент из списка по значению
5 - Получить результирующий список
6 - Вывести результирующий список
0 - Выход
=====
Выберите действие: 1

```

2) Команда 1 позволяет добавить элемент в список.



```

MacBook-Pro-Mikhail:ЛР №2 ЛИНЕЙНЫЕ И ЦИКЛИЧЕСКИЕ СПИСКИ ms$ ./lab-2

===== ДОСТУПНЫЕ КОМАНДЫ =====
Выберите действие: 1

Введите элемент: 13
Запись добавлена!

===== ДОСТУПНЫЕ КОМАНДЫ =====
1 - Добавить элемент в список
2 - Вывести исходный список
3 - Удалить элемент из списка по номеру
4 - Удалить элемент из списка по значению
5 - Получить результирующий список
6 - Вывести результирующий список
0 - Выход
=====
Выберите действие: 2

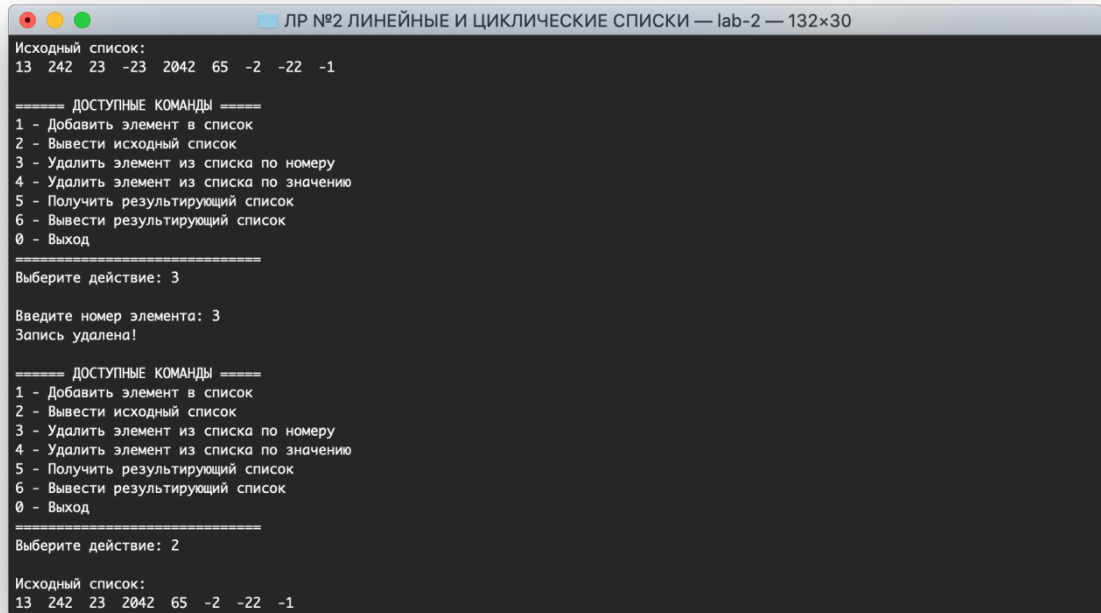
Исходный список:
13

```



3) Команда 2 позволяет вывести исходный список

4) Команда 3 позволяет удалить элемент из списка по его порядковому номеру



```
ЛР №2 ЛИНЕЙНЫЕ И ЦИКЛИЧЕСКИЕ СПИСКИ — lab-2 — 132x30
Исходный список:
13 242 23 -23 2042 65 -2 -22 -1

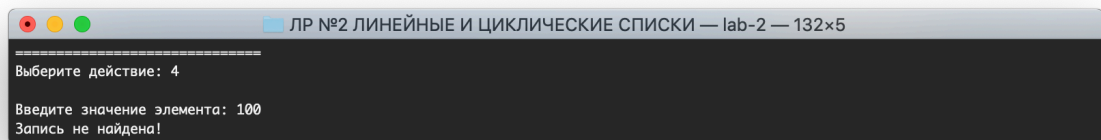
===== ДОСТУПНЫЕ КОМАНДЫ =====
1 - Добавить элемент в список
2 - Вывести исходный список
3 - Удалить элемент из списка по номеру
4 - Удалить элемент из списка по значению
5 - Получить результирующий список
6 - Вывести результирующий список
0 - Выход
=====
Выберите действие: 3

Введите номер элемента: 3
Запись удалена!

===== ДОСТУПНЫЕ КОМАНДЫ =====
1 - Добавить элемент в список
2 - Вывести исходный список
3 - Удалить элемент из списка по номеру
4 - Удалить элемент из списка по значению
5 - Получить результирующий список
6 - Вывести результирующий список
0 - Выход
=====
Выберите действие: 2

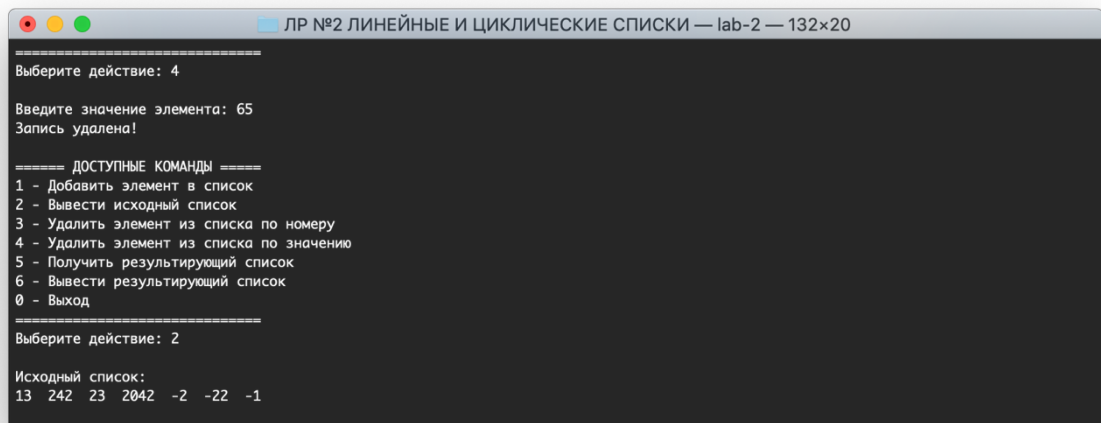
Исходный список:
13 242 23 2042 65 -2 -22 -1
```

5) Команда 4 позволяет удалить элемент из списка по его значению



```
ЛР №2 ЛИНЕЙНЫЕ И ЦИКЛИЧЕСКИЕ СПИСКИ — lab-2 — 132x5
=====
Выберите действие: 4

Введите значение элемента: 100
Запись не найдена!
```



```
ЛР №2 ЛИНЕЙНЫЕ И ЦИКЛИЧЕСКИЕ СПИСКИ — lab-2 — 132x20
=====
Выберите действие: 4

Введите значение элемента: 65
Запись удалена!

===== ДОСТУПНЫЕ КОМАНДЫ =====
1 - Добавить элемент в список
2 - Вывести исходный список
3 - Удалить элемент из списка по номеру
4 - Удалить элемент из списка по значению
5 - Получить результирующий список
6 - Вывести результирующий список
0 - Выход
=====
Выберите действие: 2

Исходный список:
13 242 23 2042 -2 -22 -1
```

6) Команда 5 позволяет создать результирующий список – копию исходного списка с удаленными двумя минимальными и двумя максимальными элементами. Исходный список при этом сохраняется нетронутым.

7) Команда 6 позволяет вывести результирующий список

```
ЛР №2 ЛИНЕЙНЫЕ И ЦИКЛИЧЕСКИЕ СПИСКИ — lab-2 — 132x31
Исходный список:
13 242 23 2042 -2 -22 -1

===== ДОСТУПНЫЕ КОМАНДЫ =====
1 - Добавить элемент в список
2 - Вывести исходный список
3 - Удалить элемент из списка по номеру
4 - Удалить элемент из списка по значению
5 - Получить результирующий список
6 - Вывести результирующий список
0 - Выход
=====
Выберите действие: 5

Два максимальных элемента: 2042 и 242
Два минимальных элемента: -22 и -2
Результирующий список создан!

===== ДОСТУПНЫЕ КОМАНДЫ =====
1 - Добавить элемент в список
2 - Вывести исходный список
3 - Удалить элемент из списка по номеру
4 - Удалить элемент из списка по значению
5 - Получить результирующий список
6 - Вывести результирующий список
0 - Выход
=====
Выберите действие: 6

Результирующий список:
13 23 -1
```

## 5. Вывод

Была создана и протестирована программа с пользовательским интерфейсом для работы с двунаправленными связными списками. Были изучены основы работы со связными списками в C++ и изучена реализация таких операций с ними, как добавление, удаление и поиск элементов. Программа работает без ошибок и позволяет выполнить задачу, данную в задании.