Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет прикладной математики и информатики
Кафедра вычислительной математики и программирования

Курсовой проект
По дисциплине
«Практикум на ЭВМ»
2 семестр

Задание VIII
«Линейные списки»

Студент:	Сикорский А. А.
Группа:	М8О-108Б-20
Преподаватель:	Трубченко Н. М.
Подпись:	
Оценка:	
Дата:	20.05.2021

Москва
2021

Содержание

Задание

Составить и отладить на языке Си программу для обработки линейного списка заданной организации с отображением списка на динамические структуры. Навигацию по списку следует реализовать с помощью итераторов. Предусмотреть выполнение одного нестандартного и четырех стандартных. Стандартные действия:

1. **Печать списка**
2. **Вставка нового элемента в список**
3. **Удаление элемента из списка**
4. **Подсчет длины списка**

1 Программа

1.1 Структура данных

Двунаправленный список. Это такая структура данных, написанная на указателях, в которой нам гарантируется вставка и удаление за $O(1)$. Еще к преимуществам можно отнести потенциально больший размер списка по сравнению с массивом, ведь список не требует последовательного выделения памяти, но за это стоит платить невозможностью прямого доступа к элементу, как мы это делаем в массиве по индексу. Двунаправленность значит, что в каждом элементе списка содержится указатель не только на следующий, но и на предыдущий элемент. Таким образом доступна навигация как влево, так и вправо. Сама навигация по списку и его функции работают с помощью итераторов. Так нам не особо важна реализация списка. Используя итераторы, мы можем поменять внутреннее устройство списка и изменить только реализацию итераторов. При этом остальные функции по типу удаления, поиска и сортировки не сломаются. У списка есть следующие функции:

1. **Create** выделяет память под *head* и инициализирует пустой список.
2. **Insert** вставляет элемент перед переданным как аргумент итератором.
3. **Delete** удаляет элемент на месте, куда указывает итератор.
4. **Size** возвращает размер списка.
5. **isEmpty** проверяет список на пустоту.

6. **Write** записывает данные по итератору.
7. **First и Last** возвращают итераторы на первый и терминирующий элементы соответственно.
8. **Read** считывает элемент по итератору.
9. **nPos** возвращает итератор, указываюший на n элемент списка.
10. **Next и Prev** двигают переданный итератор вперед и назад по списку.
11. **Equals** проверяет равенство двух итераторов.
12. **move** двигает итератор на n элементов вправо.
13. **Destroy** очищает список.

1.2 Интерфейс и и выполнение задания

Это лабораторная с интерфейсом. Пользователь может вводить с клавиатуры и выполнять следующие действия во время работы программы:

'+ value position' - вставка

'- position' - удаление

r position - чтение элемента

s - размер списка

p - печать списка

a k - задание (удалить каждый k элемент списка)

c - очистка списка

X - выход из программы с очисткой списка

Если программа завершится, например, при получении EOF, все равно будет вызвана функция *Destroy* и список очистится.

Листинг 1: Задание варианта

```
    case 'a': {
        getchar();
        int k;
        scanf("%d",&k);
        int index = 0;
        iterator last = Last(&l);
        iterator iterator1 = First(&l);
        for (iterator1 = First(&l); !Equals(&iterator1, &last); ){
            index++;
            if (index % k == 0) {
                iterator temp = iterator1;
                Delete(&l, &iterator1);
                iterator1 = temp;
            }
            Next(&iterator1);
        }
        printf("\n");
        break;
    }
}
```

Это код, выполняющий задание варинта по удалению каждого k удаления списка. Когда цикл заходит в этот case, считывается значение k и запускается продвижение по циклу со счетчиком. Когда значение счетчика становится кратным k, удаляется элемент, на который указывает движущийся итератор *first*.

2 Вывод

В ходе работы я составил и отладил на языке Си программу для обработки двусвязного линейного списка с барьерным элементом с отображением списка на динамические структуры. Навигацию по списку реализовал с помощью итераторов и выполнил 4 стандартных действия и одно нестандартное, заключающееся в удалении каждого k элемента списка.