МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский Авиационный Институт»
(Национальный Исследовательский Университет)

Институт: №8 «Информационные технологии и прикладная
математика»
Кафедра: 806 «Вычислительная математика и программирование»

Курсовая работа
по курсу «Практикум на ЭВМ»
II семестр
Задание 8
«Линейные списки»

Группа	М8О-107Б-20
Студент	Алапанова Э.Х.
Преподаватель	Найдёнов И.Е.
Оценка	
Дата	

Москва, 2021

Постановка задачи

Составить и отладить программу на языке Си для обработки линейного списка заданной организации с отображением списка на динамические структуры. Навигацию по списку следует реализовать с применением итераторов. Предусмотреть выполнение одного нестандартного действия и четырёх стандартных действий:
Нестандартное действие:
Удалить из середины списка *k* элементов
Стандартные действия:
Печать списка
Вставка нового элемента в список
Удаление элемента из списка
Подсчёт длины списка
Вид списка:
Кольцевой однонаправленный
Тип элемента списка:
Строковый

Теория

Линейный список-структура данных, состоящая из элементов одного типа, связанных с помощью указателей. **Циклический (кольцевой) список** – это структура данных, представляющая собой последовательность элементов, последний элемент которой содержит указатель на первый элемент списка, а первый (в случае двунаправленного списка) – на последний. *Циклический однонаправленный список* похож на линейный однонаправленный список, но его последний элемент содержит указатель, связывающий его с первым элементом.

Исходной код

Файл list.h — заголовочный файл

```
#ifndef LIST_H

#define LIST_H


#include <stdio.h>

#include <stdlib.h>
```

```
typedef char LIST_TYPE;

typedef struct _Item
{
	LIST_TYPE _data;
	int _next;
} Item;

typedef struct _List
{
	Item* _arr;
	int _first;
	int _hole;
	int _capacity;
	int _size;
} List;

typedef struct _Iterator
{
	Item* _begin;
	int _index;
} Iterator;

void listCreate(List* list, const int size);
int findPrev(List* list, const int index);
void listInsert(List* list, const int index, const LIST_TYPE value);
void listRemove(List* list, const int index);
```

```
int listSize(const List* list);
int listEmpty(const List* list);
void listPrint(const List* list);
void listDestroy(List* list);

Iterator itFirst(const List* list);
void itNext(Iterator* it);
LIST_TYPE itFetch(const Iterator* it);

#endif
```

Файл list.c

```
#include "list.h"

void listCreate(List* list, const int size)
{
	int i;

	if (size <= 0)
		return;

	list->_arr = (Item*)malloc(sizeof(Item) * size);

	for (i = 0; i < size - 1; i++)
		list->_arr[i]._next = i + 1;

	list->_arr[size - 1]._next = -1;
```

```
	list->_first = 0;
	list->_hole = 0;
	list->_capacity = size;
	list->_size = 0;
}

int findPrev(List* list, const int index)
{
	int i, prev = list->_first;

	if (list->_size <= 1)
		return prev;

	if (index == 0 || index == list->_size)
		while (list->_arr[prev]._next != list->_first)
			prev = list->_arr[prev]._next;
	else
		for (i = 0; i < index - 1; i++)
			prev = list->_arr[prev]._next;

	return prev;
}

void listInsert(List* list, const int index, const LIST_TYPE value)
{
	int i, prev, nextHole;
```

```
if (list->_size == list->_capacity)
{
	printf("Ошибка. Список полон\n");

	return;
}

if (list->_size)
{
	if (index < 0 || index > list->_size)
	{
		printf("Ошибка. Позиция не найдена\n");

		return;
	}
}
else if (index != 0)
{
	printf("Ошибка. Позиция не найдена\n");

	return;
}

prev = findPrev(list, index);
nextHole = list->_arr[list->_hole]._next;
list->_arr[list->_hole]._data = value;
```

```
	list->_arr[list->_hole]._next = list->_arr[prev]._next;

	if (index == 0)
		list->_first = list->_hole;

	list->_arr[prev]._next = list->_hole;
	list->_hole = nextHole;
	list->_size++;

	printf("Элемент %c вставлен в список\n", value);
}

void listRemove(List* list, const int index)
{
	int prev, cur;

	if (listEmpty(list))
	{
		printf("Список пуст\n");

		return;
	}
	else if (index < 0 || index >= list->_size)
	{
		printf("Ошибка. Позиция не найдена\n");

		return;
```

```
	}

	prev = findPrev(list, index);
	cur = list->_arr[prev]._next;

	printf("Элемент %c удален из списка\n", list->_arr[cur]._data);

	list->_arr[prev]._next = list->_arr[cur]._next;

	if (index == 0)
		list->_first = list->_arr[prev]._next;

	list->_arr[cur]._data = 0;
	list->_arr[cur]._next = list->_hole;
	list->_hole = cur;

	list->_size--;
}

int listSize(const List* list)
{
	return list->_size;
}

int listEmpty(const List* list)
{
	return list->_size == 0;
```

```
}

void listPrint(const List* list)
{
	int i, cur = list->_first;

	for (i = 0; i < list->_size; i++)
	{
		printf("%c ", list->_arr[cur]._data);

		cur = list->_arr[cur]._next;
	}

	printf("\n");
}

void listDestroy(List* list)
{
	if (list->_arr != NULL)
	{
		free(list->_arr);

		list->_arr = NULL;
	}

	list->_first = 0;
	list->_hole = 0;
```

```
	list->_capacity = 0;
	list->_size = 0;
}

Iterator itFirst(const List* list)
{
	Iterator it;

	it._begin = list->_arr;
	it._index = list->_first;

	return it;
}

void itNext(Iterator* it)
{
	it->_index = it->_begin[it->_index]._next;
}

LIST_TYPE itFetch(const Iterator* it)
{
	return it->_begin[it->_index]._data;
}
```

Файл kp8.c

```
#include <stdio.h>
#include "list.h"
```

```
int main(void)
{
	const int N = 10;
	int i, isFound, action, pos;
	char arg;
	List list;
	Iterator it;
	int k, n;
	char rubish;

	listCreate(&list, N);

	do
	{
		printf("Меню:\n");
		printf("1) Вставить элемент\n");
		printf("2) Удалить элемент\n");
		printf("3) Печать списка\n");
		printf("4) Подсчет длины списка\n");
		printf("5) Выполнить задание над списком\n");
		printf("6) Выход\n");
		printf("Выберите действие: ");
		scanf("%d", &action);

		switch (action)
		{
```

```
case 1:
{
	printf("Введите позицию элемента: ");
	scanf("%d", &pos);
	scanf("%c", &rubish);
	printf("Введите значение элемента: ");
	scanf("%c", &arg);
	listInsert(&list, pos - 1, arg);

	break;
}

case 2:
{
	printf("Введите номер элемента: ");
	scanf("%d", &pos);

	listRemove(&list, pos - 1);

	break;
}

case 3:
{
	if (listEmpty(&list))
		printf("Список пуст\n");
	else
```

```
                listPrint(&list);

            break;
        }

        case 4:
        {
            printf("Длина списка: %d\n", listSize(&list));

            break;
        }

        case 5:
        {
            printf("Введите k:\n");
            scanf("%d", &k);
            if (k > listSize(&list)) printf("Введённое значение больше, чем текущий размер списка!\n");
            else {
                for (int j = 0; j < k; j++) {
                    n = listSize(&list);
                    listRemove(&list, n / 2);
                }
            }

            break;
        }
```

```
            case 6: break;

            default:
            {
                printf("Ошибка. Такого пункта меню не существует\n");

                break;
            }
            }
        } while (action != 6);

        listDestroy(&list);

        return 0;
}
```

Протокол исполнения и тесты

Тест №1. Добавление элемента + печать

Меню:

1) Вставить элемент

2) Удалить элемент

3) Печать списка

4) Подсчет длины списка

5) Выполнить задание над списком

6) Выход

Выберите действие: 1

Введите позицию элемента: 4

Введите значение элемента: 4

Элемент 4 вставлен в список

_____

Таким образом составляем список.

_____

Меню:

1) Вставить элемент

2) Удалить элемент

3) Печать списка

4) Подсчет длины списка

5) Выполнить задание над списком

6) Выход

Выберите действие: 3

1 2 3 4

1 2 4

Тест No2. Удаление элемента.

Меню:

1) Вставить элемент

2) Удалить элемент

3) Печать списка

4) Подсчет длины списка

5) Выполнить задание над списком

6) Выход

Выберите действие: 2

Введите номер элемента: 3

Элемент 3 удален из списка

Меню:

1) Вставить элемент

2) Удалить элемент

3) Печать списка

4) Подсчет длины списка

5) Выполнить задание над списком

6) Выход

Выберите действие: 3

_____

Меню:

1) Вставить элемент

2) Удалить элемент

3) Печать списка

4) Подсчет длины списка

5) Выполнить задание над списком

6) Выход

Выберите действие: 3

5 1 2 4

Меню:

1) Вставить элемент

2) Удалить элемент

3) Печать списка

4) Подсчет длины списка

5) Выполнить задание над списком

6) Выход

Выберите действие: 2

Введите номер элемента: 2

Элемент 1 удален из списка

Меню:

1) Вставить элемент

2) Удалить элемент

3) Печать списка

4) Подсчет длины списка

5) Выполнить задание над списком

6) Выход

Выберите действие: 3

5 2 4

Тест №3. Длина списка

Меню:

1) Вставить элемент

2) Удалить элемент

3) Печать списка

4) Подсчет длины списка

5) Выполнить задание над списком

6) Выход

Выберите действие: 3

5 2 4

Меню:

1) Вставить элемент

2) Удалить элемент

3) Печать списка

4) Подсчет длины списка

5) Выполнить задание над списком

6) Выход

Выберите действие: 4

Длина списка: 3

Тест №4. Выполняем задание

Меню:

1) Вставить элемент

2) Удалить элемент

3) Печать списка

4) Подсчет длины списка

5) Выполнить задание над списком

6) Выход

Выберите действие: 3

7 5 8 2 5 4

Меню:

1) Вставить элемент

2) Удалить элемент

3) Печать списка

4) Подсчет длины списка

5) Выполнить задание над списком

6) Выход

Выберите действие: 5

Введите k:

2

Элемент 2 удален из списка

Элемент 8 удален из списка

Меню:

1) Вставить элемент

2) Удалить элемент

3) Печать списка

4) Подсчет длины списка

5) Выполнить задание над списком

6) Выход

Выберите действие: 3

7 5 5 4

Тест №5. Ошибка

Меню:

1) Вставить элемент

2) Удалить элемент

3) Печать списка

4) Подсчет длины списка

5) Выполнить задание над списком

6) Выход

Выберите действие: 4

Длина списка: 4

Меню:

1) Вставить элемент

2) Удалить элемент

3) Печать списка

4) Подсчет длины списка

5) Выполнить задание над списком

6) Выход

Выберите действие: 2

Введите номер элемента: 5

Ошибка. Позиция не найдена

Меню:

1) Вставить элемент

2) Удалить элемент

3) Печать списка

4) Подсчет длины списка

5) Выполнить задание над списком

6) Выход

Выберите действие: 5

Введите k:

5

Введённое значение больше, чем текущий размер списка!

Вывод

Хорошая курсовая работа. Понравилось, что заставили изучить классические алгоритмы.

Примечание: ссылка на GitHub с кодом программы https://github.com/alpnva/MAI/tree/main/kp8