«Московский Авиационный Институт»
(Национальный Исследовательский Университет)

Институт: №8 «Прикладная математика и информатика»
Кафедра: 806 «Вычислительная математика и программирование»

Курсовая работа
II семестр
По теме
«Линейные списки»

Группа	М8О-107Б-20
Студент	Чекменев В.А.
Преподаватель	Найдёнов И.Е.
Оценка	
Дата	

Москва, 2021

Теория

Связный список — базовая динамическая структура данных в информатике, состоящая из узлов, каждый из которых содержит как собственно данные, так и одну или две ссылки («связки») на следующий и/или предыдущий узел списка. Принципиальным преимуществом перед массивом является структурная гибкость: порядок элементов связного списка может не совпадать с порядком расположения элементов данных в памяти компьютера, а порядок обхода списка всегда явно задаётся его внутренними связями.

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 и четырех стандартных действий:

1.Печать списка;

2.Вставка нового элемента в список;

3.Удаление элемента из списка;

4.Подсчет длины списка;

Вариант:

ТИП ЭЛЕМЕНТА СПИСКА:

3. перечислимый.

ВИД СПИСКА:

2. Линейный однонаправленный.

НЕСТАНДАРТНОЕ ДЕЙСТВИЕ:

8. дополнить список копиями заданного значения до указанной длины K. Если в списке уже имеется K элментов, то не менять его.

Общий метод решения

Программа должна выводить меню, и в зависимости от запроса, выдавать результат в интерактивном режиме. Предусматривается возможность завершения программы пользователем.

Линейный однонаправленный список - это структура данных, состоящая из элементов одного типа, связанных между собой последовательно посредством массива элементов типа enum. Реализация списка будет производиться с помощью структуры вектора.

Листинг программ:

list.h----------

```
#ifndef list_h
#define list_h

#include <stdlib.h>
#include <stdio.h>

typedef enum {
    STEP_FORW = 0, // 0
    STEP_BACK, // 1
    STEP_LEFT, // 2
    STEP_RIGHT, // 3
    CH_DIR_180, // 4
} Actions;

typedef struct list {
    int number_of_elements;
    int capacity;
    Actions *elements;
} list;

void create_list(list *l,int capacity);
void printAction(Actions action);
int is_list_empty(list *l);
int size(list *l);
void resize(list *l);
```

```
void push_back(list *l, Actions act);
void print_list(list *l);
void delete_element_by_index(list *l);
void add_by_index(list *l);
int number_of_elements(list *l);
void add_to_K_elements(list *l, Actions act, int k);

#endif
```

list_functions.c----------

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

void create_list(list *l, int capacity) // нужно
{
    l->capacity = capacity;
    l->number_of_elements = 0;
    l->elements = malloc(sizeof(Actions) * l->capacity);
}

void printAction(Actions action)
{
    if (action == STEP_FORW)
        printf("[step forward]");
    else if (action == STEP_BACK)
        printf("[step back]");
    else if (action == STEP_LEFT)
```

```
        printf("[step left]");
    else if (action == STEP_RIGHT)
        printf("[step right]");
    else if (action == CH_DIR_180)
        printf("[change direction by 180 degrees]");
}

int is_list_empty(list *l)
{
    if (l->number_of_elements == 0) {
        return 1;
    } else {
        return 0;
    }
}

int size(list *l)
{
    return l->capacity;
}

void resize(list *l)
{
    l->capacity++;
    l->elements = realloc(l->elements, sizeof(Actions) * l->capacity);
}

void push_back(list *l, Actions act) // нужно
{
```

```
    if (l->number_of_elements == l->capacity) {
        resize(l);
    }
    l->elements[l->number_of_elements] = act;
    l->number_of_elements++;
}

void print_list(list *l) // нужно
{
    printf("Печать списка:\n");

    if (is_list_empty(l)) {
        printf("Список пуст((\n");
    }
    for (int i = 0; i < l->number_of_elements; i++) {
        if (i == l->number_of_elements - 1) {
            printAction(l->elements[i]); printf("\n");
        } else {
            printAction(l->elements[i]); printf("->");
        }
    }
}

void delete_element_by_index(list *l) // нужно
{
    int ind;

    printf("Введите индекс элемента, который хотите удалить: ");
    scanf("%d", &ind);
```

```
    if (l->number_of_elements == 0) {
        printf("Нечего удалять, список пуст((\n");
        return;
    }
    while ((ind >= l->number_of_elements) || (ind < 0)) {
        printf("Слишком большой или маленький индекс, введите поменбше/поболбше...\n");
        scanf("%d", &ind);
    }
    l->elements[ind] = 0;

    for (int i = 0; i < l->number_of_elements - 1 - ind; i++) {
        l->elements[i + ind] = l->elements[i + ind + 1];
    }
    l->number_of_elements--;
    printf("удаляем элемент с индексом %d...\n", ind);
}

void add_by_index(list *l) // нужно
{
    if ((l->capacity < l->number_of_elements + 1) && (l->number_of_elements != 0)) {
        resize(l);
    }
    int index, flag = 0;
    Actions act, tmp_1, tmp_2;

    printf("Введите какое-нибудь действие: STEP_FORW, STEP_BACK, STEP_LEFT, STEP_RIGHT или CH_DIR_180 которое вы хотите добавить: ");
```

```
    scanf("%d", &act);
    printf("введите индекс, куда хотите добавить: ");
    scanf("%d", &index);

    if (index == l->number_of_elements) {
        push_back(l, act);
        flag = 1;
    } else {
        while ((index > l->number_of_elements - 1) || (index < 0)) {
            printf("Слишком большой индекс, введите поменбше. максимально
возможный индекс = %d\n", l->number_of_elements);
            scanf("%d", &index);
            if (index == l->number_of_elements && flag == 0) {
                push_back(l, act);
                break;
            }
        }
        tmp_2 = l->elements[index];
        tmp_1 = act;

        l->number_of_elements++;

        for (int i = 0; i < l->number_of_elements - index + 1; i++) {
            if (l->capacity < l->number_of_elements + 1) {
                resize(l);
            }
            l->elements[index + i] = tmp_1;
            tmp_1 = tmp_2;
            tmp_2 = l->elements[index + i + 1];
```

```
        }
    }
}

int number_of_elements(list *l) // нужно
{
    return l->number_of_elements;
}

void add_to_K_elements(list *l, Actions act, int k) // нужно
{
    if (l->number_of_elements < k) {
        while (l->number_of_elements < k) {
            push_back(l, act);
        }
    }
}
```

client.c----------

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

int main()
{
    list l;
    Actions act;
    char c;
```

```
    int k;


    printf("Напишите '?' для получения помощи в использовании программы:\
n");


    while ((c = getchar()) != EOF) {
        if (c == '?') {
            printf("Набор команд:\n");
            printf("c - создать списочек.\n");
            printf("b - добавить элемент в конец списка.\n");
            printf("p - напечатать список.\n");
            printf("d - удалить элемент по индексу.\n");
            printf("a - добавить элемент по индексу.\n");
            printf("e - закончить сеанс.\n");
            printf("k - количество элементов в списке.\n");
            printf("f - увеличить список до K элементов введенным элементом.\n");
        }
        else if (c == 'c') {
            printf("создаем списочек\n");
            create_list(&l, 0);
            printf("все, список создан, вводите следующую команду\n");
        }
        else if (c == 'b') {
            printf("введите действие: ");
            scanf("%d", &act);
            printf("добаляем элемент ");
            printAction(act);
            printf(" в конец списка\n");
            push_back(&l, act);
```

```
            printf("все, элемент добавлен в конец, вводите следующую команду\
n");
        }
        else if (c == 'p') {
            printf("печатаем списочек\n");
            print_list(&l);
            printf("все, список напечатан, вводите следующую команду\n");
        }
        else if (c == 'd') {
            printf("удаляем элемент по индексу\n");
            delete_element_by_index(&l);
            printf("все, элемент удален, вводите следующую команду\n");
        }
        else if (c == 'a') {
            printf("добавляем элемент по индексу\n");
            add_by_index(&l);
            printf("все, элемент добавлен, вводите следующую команду\n");

        }
        else if (c == 'e') {
            printf("все на сегодня...\n");
            return 0;
        }
        else if (c == 'k') {
            printf("выводим количество элементов в списке\n");
            printf("%d\n", number_of_elements(&l));
            printf("все, готово, вводите следующую команду\n");
        }
        else if (c == 'f') {
```

```
            printf("увеличиваем список\n");
            printf("введите действие: ");
            scanf("%d", &act);
            printf("введите новое количество элементов: ");
            scanf("%d", &k);
            add_to_K_elements(&l, act, k);
            printf("все, элементы добавлен, вводите следующую команду\n");
        }
    }
    return 0;

}
```

Тестирование программы

Неполадки
Была ошибка в функции добавления элемента по индексу. При вводе флага «a» и индекса, большего количества элементов в списке, цикл while не останавливался.
Как исправил:
добавил условие if index >= number_of_elements, в этом случае делал push_back.

Вывод программы будет выделен **жирным**.

[suraba04@asusx512fl cp8]$ gcc *.c -o test0
[suraba04@asusx512fl cp8]$ ./test0
Напишите '?' для получения помощи в использовании программы:
?
Набор команд:
c - создать списочек.
b - добавить элемент в конец списка(введите цифры от 0 до 4, они обозначают действия: STEP_FORW(0), STEP_BACK,(1) STEP_LEFT(2), STEP_RIGHT(3) или CH_DIR_180(4)).
p - напечатать список.
d - удалить элемент по индексу.
a - добавить элемент по индексу.
e - закончить сеанс.
k - количество элементов в списке.
f - увеличить список до K элементов введенным элементом.
c
создаем списочек
все, список создан, вводите следующую команду
p
печатаем списочек
Печать списка:
Список пуст((
все, список напечатан, вводите следующую команду
b
введите действие: 0
добаляем элемент [step forward] в конец списка
все, элемент добавлен в конец, вводите следующую команду
b
введите действие: 1
добаляем элемент [step back] в конец списка
все, элемент добавлен в конец, вводите следующую команду
b
введите действие: 2
добаляем элемент [step left] в конец списка
все, элемент добавлен в конец, вводите следующую команду
b
введите действие: 3
добаляем элемент [step right] в конец списка
все, элемент добавлен в конец, вводите следующую команду
b
введите действие: 4
добаляем элемент [change direction by 180 degrees] в конец списка
все, элемент добавлен в конец, вводите следующую команду

p
печатаем списочек
Печать списка:
[step forward]->[step back]->[step left]->[step right]->[change direction by 180 degrees]
все, список напечатан, вводите следующую команду
d
удаляем элемент по индексу
Введите индекс элемента, который хотите удалить: 1
удаляем элемент с индексом 1...
все, элемент удален, вводите следующую команду
p
печатаем списочек
Печать списка:
[step forward]->[step left]->[step right]->[change direction by 180 degrees]
все, список напечатан, вводите следующую команду
?
Набор команд:
c - создать списочек.
b - добавить элемент в конец списка(введите цифры от 0 до 4, они обозначают действия: STEP_FORW(0), STEP_BACK,(1) STEP_LEFT(2), STEP_RIGHT(3) или CH_DIR_180(4)).
p - напечатать список.
d - удалить элемент по индексу.
a - добавить элемент по индексу.
e - закончить сеанс.
k - количество элементов в списке.
f - увеличить список до K элементов введенным элементом.
a
добавляем элемент по индексу
Введите какое-нибудь действие: STEP_FORW(0), STEP_BACK,(1) STEP_LEFT(2), STEP_RIGHT(3) или CH_DIR_180(4), которое вы хотите добавить: 1
введите индекс, куда хотите добавить: 1
все, элемент добавлен, вводите следующую команду
p
печатаем списочек
Печать списка:
[step forward]->[step back]->[step left]->[step right]->[change direction by 180 degrees]
все, список напечатан, вводите следующую команду
k
выводим количество элементов в списке
5
все, готово, вводите следующую команду
f
увеличиваем список
введите действие: 0
введите новое количество элементов: 20
все, элементы добавлен, вводите следующую команду
p
печатаем списочек
Печать списка:
[step forward]->[step back]->[step left]->[step right]->[change direction by 180 degrees]->[step forward]->[step forward]->[step forward]->[step

forward]->[step forward]->[step forward]->[step forward]->[step forward]->[step forward]->[step forward]->[step forward]->[step forward]->[step forward]->[step forward]->[step forward]
все, список напечатан, вводите следующую команду
e
все на сегодня...
[suraba04@asusx512fl cp8]$

Вывод:

В данной курсовой работе я научился на базовом уровне работать со списками на векторе, вспомнил как работает перечислимый тип. Опять же, больше всего понравилось писать интерфейс. Самой сложной частью в работе было отлавливание крайних случаев.