Отчёт по заданию №8 курсового проекта

Черыгова Елизавета
Группа 8О-104Б

Оглавление

Цель работы

Составить и отладить программу на языке Си для обработки линейного списка заданной организации с отображением на динамические структуры. Навигацию по списку следует реализовать с применением итераторов. Тип элементов списка — целый. Предусмотреть выполнение одного нестандартного и четырех стандартных действий:

- печать списка;
- вставка нового элемента в список;
- удаление элемента списка;
- подсчет длины списка.

Входные данные

На стандартный ввод программе подаются команды пяти типов:

p — печать списка; i 2 4 — вставка перед вторым элементом элемента со значением 4. d 5 — удаление первого встретившегося элемента со значением 5. l — печать длины списка. t [params] — выполнение заданного вариантом действия. Параметры, если они есть, перечислены через пробел.

Выходные данные

После чтения и выполнения каждой команды программа должна вывести результат операции.

Вывести все элементы списка через пробел. Вывести сообщение «OK» в случае успеха и «Error» в случае неудачи (при попытке вставить на несуществующую позицию в некольцевом списке). Вывести сообщение «OK» при успешном удалении или «Not found», если элемента с таким значением в списке нет. Вывести длину списка. Вывести сообщение «OK» в случае успешного выполнения операции или «Error» в случае неудачи.

Задание:

Отображение списка на массив.

Только индексный доступ к списку.

Тип элемента списка: литерный.

Тип списка: линейный однонаправленный.

Действие: переставить первую и вторую половины списка.

Алгоритм решения задачи

Делим пополам список, и столько раз, сколько элементов во второй половине. Удаляем последний элемент и ставим его на первое место.

Кодпрограммы

```
/*      Курсовой проект №8
        Отображение списка на массив.
        Только индексный доступ к списку.
        Тип элемента списка: литерный.
        Тип списка: линейный однонаправленный.
        Действие: переставить первую и вторую половины списка.*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "function8.h"


int main(){
        list l;
        list_init(&l);
        char com[10];
        unsigned char liter;
        int pos;
        int err_num;            //переменная ошибок.

        help();

        while(strcmp(com,"exit")!=0){
                printf("=>");
                scanf("%s", com);

                if(strcmp(com, "help")==0){
                        help();
                }
                else if(strcmp(com, "in")==0){
                        scanf("%d %c", &pos, &liter);
                        err_num=insert(&l, pos, liter);
                        if(err_num){
                                printf("Error. Code %d\n", err_num);
                        }
                }
                else if(strcmp(com, "pb")==0){
                        scanf(" %c", &liter);
                        err_num=push_back(&l,liter);
                        if(err_num){
                                printf("Error. Code %d\n", err_num);
                        }
                }
```

```
            else if(strcmp(com,"del")==0){
                scanf(" %c", &liter);
                err_num=del(&l, liter);
                if(err_num){
                    printf("Error. Code %d\n", err_num);
                }
            }
            else if(strcmp(com, "print")==0){
                list_print(&l);
            }
            else if(strcmp(com,"size")==0){
                printf("%d\n", list_size(l));
            }
            else if(strcmp(com, "solution")==0){
                solution(&l);
            }
            else if(strcmp(com,"exit")==0){
                ;
            }
            else printf("Unknown command.\n");
        }

        free(l.vec);
        return 0;
}
```

```
#ifndef FUNCTION8_H
#define FUNCTION8_H

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

#define START_LIST_SIZE 10                                  // Начальный размер списка.
#define MEM_EPS 5                                           // Разница между размером
списка и выделенной памятью, после которой происходит удвоение.

typedef struct{
        unsigned char* vec;                                 // Массив, хранящий список.
        int mem_inf;                                        // Инормация о памяти,
занимаемой массивом.
}list;

/*******Объявление функций.*********/
void help();                                                // Печать помощи.

void list_print(list*);                                     // Печать списка.
int insert(list*, int, unsigned char);              // Вставка элемента в список. Возврат: 0 - ок, 1 -
индекс вне размеров списка.
int del(list*, unsigned char);                      // Удаление элемента из списка. Возврат:
0 - ок, -1 - элемент не найден.
int list_size(list);                                        // Подсчет длины списка.
Возвращает количество элементов в списке.
int list_find(list, unsigned char);                 // Поиск элемента в списке. Возвращает
индекс или -1, если элемент не наден.

void list_init(list*);                                      // Инициализация списка.
void list_resize(list*);                            // Удваивает память для хранения списка.

unsigned char pop_back(list *l);                    //Возвращает последний элемент и
удаляет его.
int solution(list*);                                        //Выполняет задание, т.е.
переставляет 1-ю и вторую половину списка.
/*********************************/

void help(){
        puts("============================================================");
        puts("help.....................print help");
        puts("print.....................pint list");
        puts("in <pos> <let>.............insert letter into list at pos");
        puts("pb <let>.....................add letter in the list end");
        puts("del <let>...................delete letter from list");
        puts("size.....................print list size");
```

```
	puts("solution.....solve the problem.");
	puts("============================================================");
}

void list_init(list* l){
	l->vec=NULL;
				// Зануляем указатель (на всякий случай)
	l->vec=(unsigned char*)malloc(START_LIST_SIZE*sizeof(unsigned char)); // Выделяем
память на START_LIST_SIZE элементов
	l->mem_inf=START_LIST_SIZE;
				// Сообщаем, сколько памяти было выделено.
	l->vec[0]=0;
				// Присваиваем первому элементу 0(конец массива).
}

void list_resize(list*l){
	l->vec=realloc(l->vec,2*sizeof(l->vec));			// Перераспределяем
память(выделяем в 2 раза больше)
	l->mem_inf*=2;
}

void list_print(list* l){
	int i;
	for(i=0;i<list_size(*l);i++){
		printf("%d. %c\n",i, l->vec[i]);
	}
	return;
}

int list_size(list l){
	int size = 0;
	while(l.vec[size]!=0){
		size++;
	}
	return size;
}

int insert(list* l,int index, unsigned char c){
	//index--;
	// Если первый элемент нулевой, убрать эту строку.
	int size=list_size(*l),i;
	unsigned char temp[2];
	if((l->mem_inf - size) < MEM_EPS) list_resize(l);	// Проверяем размер массива.
Увеличиваем, если надо.
	if(index>size || index<0){					// Если индекс
вне границ списка, выходим с ошибкой.
```

```
                return -1;
        }
        else{
        // Если все хорошо, вставляем элемент.
                temp[index%2]=c;
                for(index; index<size+1; index++){
                        //printf("DEBUG: iter\n");
                        temp[(index+1)%2]=l->vec[index];
                        l->vec[index]=temp[index%2];
                }
                l->vec[index]=0;
        }
        return 0;
}

int list_find(list l, unsigned char c){
        int index=0;
        while(l.vec[index]!=0){
                if(l.vec[index]==c) return index;
                index++;
        }
        return -1;                                                      // Не нашли,
возвращаем -1.
}

int del(list* l, unsigned char c){
        int index, size=list_size(*l), i;
        index=list_find(*l, c);
        if(index==-1){
                return -1;
        }
        else{
                for(i=index;i<size;i++){
                        l->vec[i]=l->vec[i+1];
                }
        }
        return 0;
}

int push_back(list* l, unsigned char c){
        return insert(l, list_size(*l), c);
}


unsigned char pop_back(list *l){
        unsigned char c;
```

```
	int size=list_size(*l);
	c = l->vec[size-1];
	l->vec[size-1]=l->vec[size];
	return c;
}

int solution(list *l){
	int size = list_size(*l);
	int i,k = (size-size%2)/2;
	unsigned char c;
	for(i=0;i<k;i++){
		c = pop_back(l);
		insert(l,0,c);
	}
	return 0;
}




#endif
```

Вывод программы

```
============================================================
help.......................print help
print......................pint list
in <pos> <let>.............insert letter into list at pos
pb <let>...................add letter in the list end
del <let>..................delete letter from list
size.......................print list size
solution...................solve the problem
============================================================
=>зис
Unknown command.
=>pb a
=>pb f
=>pb s
=>pb g
=>pb v
=>pb x
=>pb e
=>pb a
=>print
0. a
1. f
2. s
3. g
4. v
5. x
6. e
7. a
=>solution
=>print
0. v
1. x
2. e
3. a
4. a
5. f
6. s
7. g
=>del f
=>print
0. v
1. x
2. e
3. a
4. a
5. s
6. g
=>in 4 d
=>print
0. v
1. x
2. e
3. a
4. d
5. a
6. s
```

```
7. g
=>size
8
=>solution
=>print
0. d
1. a
2. s
3. g
4. v
5. x
6. e
7. a
=>pb u
=>print
0. d
1. a
2. s
3. g
4. v
5. x
6. e
7. a
8. u
=>solution
=>print
0. x
1. e
2. a
3. u
4. d
5. a
6. s
7. g
8. v
=>
```

Заключение

Благодаря, этому заданию, я смогла составить и отладить программу на языке Си для обработки линейного списка заданной организации с отображением на динамические структуры.