МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема:” Структуры данных. Линейные списки ”

Студент гр. 7381 _____ Габов Е. С.

Преподаватель _____ Берленко Т. А.

Санкт-Петербург

2017

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Габов Е. С.

Группа 7381

Тема работы : “ Структуры данных. Линейные списки “

Исходные данные:

- Создать двунаправленный список музыкальных композиций MusicalComposition и **api** (***a****pplication* ***p****rogramming* ***i****nterface - в данном случае набор функций*) для работы со списком.
- Структура элемента списка (тип - MusicalComposition)

- name - строка неизвестной длины (гарантируется, что длина не может быть больше 80 символов), название композиции.
- author - строка неизвестной длины (гарантируется, что длина не может быть больше 80 символов), автор композиции/музыкальная группа.
- year - целое число, год создания.
- Функция для создания элемента списка (тип элемента MusicalComposition)

- MusicalComposition* createMusicalComposition(char* name, char* author, int year)
- Функции для работы со списком:

- MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n); // создает список музыкальных композиций MusicalCompositionList, в котором:
  - ***n*** *- длина массивов* ***array_names***, ***array_authors***, ***array_years*****.**
  - поле **name** первого элемента списка соответствует первому элементу списка array_names (**array_names[0]**).
  - поле **author** первого элемента списка соответствует первому элементу списка array_authors (**array_authors[0]**).
  - поле **year** первого элемента списка соответствует первому элементу списка array_authors (**array_years[0]**).
- *Аналогично для второго, третьего, ...* ***n-1****-го элемента массива.*
- *! длина массивов* ***array_names, array_authors, array_years*** *одинаковая и равна n, это проверять не требуется.*

- *Функция возвращает указатель на первый элемент списка.*
- void push(MusicalComposition* head, MusicalComposition* element); // добавляет **element** в конец списка **musical_composition_list**
- void removeEl (MusicalComposition* head, char* name_for_remove); // удаляет элемент **element** списка, у которого значение **name** равно значению **name_for_remove**
- int count(MusicalComposition* head); //возвращает количество элементов списка
- void print_names(MusicalComposition* head); //Выводит названия композиций
- Получить срез списка на вход подаётся указатель на первый элемент списка и 2 индекса (начало и конец)
- Создать удобный для работы пользователя интерфейс.

Предполагаемый объем пояснительной записки:

Не менее 16 страниц.

Дата выдачи задания: 23.11.2017

Дата сдачи реферата:

Дата защиты реферата:

Студент _____ Габов Е. С.

Преподаватель _____ Берленко Т. А.

АННОТАЦИЯ

В результате выполнения курсовой работы создан двунаправленный линейный список. Созданы функции для работы со списком: удаление элемента списка, добавление в конец списка нового элемента, подсчет количества элементов в списке, вывод среза списка. Создан удобный для пользователя интерфейс.

SUMMARY

As a result of performing the coursework, created a bidirectional linear list. Created functions to work with list: delete list item, add to end of list new item, counting the number of elements in the list. Created user-friendly interface.

СОДЕРЖАНИЕ

Введение	6
Описание тела функции main.c	7
Описание функций для работы со списком	8
Заключение	10
Список использованных источников	11
Приложение А. Исходный код программы	12

ВВЕДЕНИЕ

Целью работы является закрепить имеющиеся знания по выделению и очистке динамической памяти. Научиться работать с двунаправленным списком и написать функции для работы с ним: добавлять и удалять элементы, подсчитывать их количество.

1. ОПИСАНИЕ ТЕЛА ФУНКЦИИ MAIN.C

1.1. С помощью функции scanf (процедура ввода общего назначения, которая читает поток stdin и сохраняет информацию в переменных, перечисленных в списке аргументов.) вводиться количество элементов, которые будут созданы в списке.

1.2. Динамически выделяется память под массив названий музыкальных композиций, их авторов и лет создания.

1.3. Заполняются данные списка

1.4. С помощью созданной функции createMusicalCompositionalList создается двусвязный линейный список.

1.5. Пользователю предоставляется право выбора функции для работы со списком

0 – добавить элемент в конец списка

1 – удалить элемент с совпадающим названием музыкальной композиции

2 – вывести количество элементов в списке

3 – вывести названия всех музыкальных композиций

4 – вывести срез списка

1.6. После выполнения выбранной функции предоставляется выбор: продолжить работу со списком или закончить выполнение программы

1.7. С помощью функции free() очищается вся память выделенная динамически.

2. ОПИСАНИЕ ФУНКЦИЙ ДЛЯ РАБОТЫ СО СПИСКОМ

2.1. Создана структура с именем MusicalComposition с членами типов char*,char*,int , и 2 члена с типом указателя на саму структуру, эти члены содержат адреса предыдущих и последующих элементов.

2.2 Далее создается ряд функция для работы с двухсвязным списком:

2.2.1. createMusicalComposition.

Создана функция createMusicalComposition. Эта функция создает элемент списка. На вход она получает 3 параметра с именем, автором и годом создания. В ней используются такие функции как malloc(динамическое выделение памяти) и strcpy(копирование символов).Функция возвращает указатель на первый элемент списка.

2.2.2. Push

Создана функция push. Эта функция добавляет элемент списка в конец списка. На вход получает указатель на первый элемент и элемент который нужно добавить в список. С помощью цикла while определяется местоположение последнего элемента. Функция ничего не возвращает.

2.2.3. createMusicalCompositionList

Создана функция **createMusicalCompositionList.** Эта функция создает целый список. На вход получает массив имен, авторов, лет издания и количество элементов которые будут в списке. Если количество элементов равняется нулю функция возвращает NULL. Иначе список заполняет при помощи цикла for. В нём вызываются описанные функции push и createMusicalComposition. Функция возвращает указатель на первый элемент списка.

2.2.4. removeEl

Создана функция removeEl. Эта функция удаляет элемент из списка, если значение члена name совпало с полученной функцией строкой. На вход поступает указатель на первый элемент и определенная строка символов. Функция ничего не возвращает.

2.2.5. Count

Создана функция count. Эта функция считает количество элементов в списке. На вход поступает указатель на первый элемент. В цикле while перебираются все элементы и счетчик с каждым шагом увеличивается на еденицу. Функция возвращает количество элементов.

2.2.6. Print_names

Создана функция print_names. Функция выводит на экран значение члена name всех элементов. На вход получает указатель на первый элемент списка. Функция ничего не возвращает.

2.2.7 print_srez_spiska

Создана функция print_srez_spiska. Функция выводит на экран срез списка. На вход функция получает указатель на первый элемент списка и границы среза. Функция ничего не возвращает.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работе создан двунаправленный линейный список, функции для работы со списком и удобный для пользователя интерфейс. Получены знания по работе со списком, закреплены знания по выделению и очищению динамической памяти. Создана функция удаления элемента не только из середина списка, также реализовано удаление элемента из конца списка и удаление первого элемента списка.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Б.Керниган Д.Риччи **“язык прогаммирование Си”**
2. Демидович Е. **“Основы алгоритмизации и программирования. Язык Си** “
3. *Подбельский В.С. Фомин С.С.* “**Курс программирования на Си: учебник** “
4. *Robert Sedgewick, Kevin Wayne* **“Algorithms in C”**

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

1) ТЕЛО ФУНКЦИИ MAIN.C

```
#include <stdlib.h>
                    #include <stdio.h>
                    #include <string.h>
                    #include <stddef.h>
                    #include "Gabov_Egor_lr4.h"
                    int main()
                    {
                              int length;
                              printf("введите сколько элементов будет содержать список:\n");
                              scanf("%d" , &length );
                              char** names = (char**)malloc(sizeof(char)*(length+1));
                              char** authors = (char**)malloc(sizeof(char)*(length+1));
                              int* years = (int*)malloc(sizeof(int)*length);

                              for ( int i=0; i<length; i++ )
                              {

                                        names[i] = (char*)malloc(sizeof(char)*81);
                                      authors[i] = (char*)malloc(sizeof(char)*81);
                                        printf("введите название музыкальной композиции [%d]: " , i);
                                        scanf( "%s" , names[i] );
                                        //fgets( names[i] , 81 , stdin );

                                        printf("введите автора музыкальной композиции [%d]: " , i);
                                        scanf( "%s" , authors[i] );
                                        //fgets( authors[i] , 81 , stdin );

                                        printf("введите год создания музыкальной композиции [%d]: " , i);
                                        scanf("%d" , &years[i]);
                              }
                              MusicalComposition* head = createMusicalCompositionList(names, authors, years, length);

                              int choice;
                              char* name_for_push = (char*)malloc(sizeof(char)*81);
                             char* author_for_push = (char*)malloc(sizeof(char)*81);
                             int year_for_push;
                              char* name_for_remove = (char*)malloc(sizeof(char)*81);
                              int c = 1;
                              int first , last;

                              while ( c == 1 )
                              {
                                        printf("выбирете операцию которую вы хотите совершить со списком:\n ");
                              printf("0 - добавить элемент в конец списка \n");
                                      printf("1 - удалить элемент с совпадающим названием композиции \n");
                              printf("2 - вывести количество элементов в списке \n");
                              printf("3 - вывести название всех музыкальных композиций\n");
                                        printf("4 - вывести срез списка\n");

                                        scanf("%d" , &choice);
                                        switch(choice)
                                        {
                                                  case 0:
                                                            printf("введите название добавляемой композиции: ");
```

```
                        scanf("%s" , name_for_push );

                                printf("введите автора добавляемой композиции: ");
                                scanf("%s" , author_for_push );

                                printf("введите год создания добавляемой композиции: ");
                                scanf("%d", &year_for_push);
                        MusicalComposition* element_for_push =
createMusicalComposition(name_for_push, author_for_push, year_for_push);
                                push( head , element_for_push );
                                break;
                        case 1:
                                printf("введите название композиции которую нужно убрать из
списка: ");
                                scanf( "%s" , name_for_remove );
                                removeEl(head, name_for_remove);
                                break;
                        case 2:
                                printf("количество элементов в списке: %d \n", count(head));
                                break;

                        case 3:
                                print_names(head);
                                break;

                        case 4:
                                printf("введите номер начального элемента среза списка: ");
                                scanf("%d" , &first);

                                printf("введите номер конечного элемента среза списка: ");
                            scanf("%d" , &last );
                                print_srez_spiska( head , first , last );
                                break;

                        default: printf("введенно неправильное число :( \n");
                }

                printf("введите [1] если хотите продолжить и любое другое число для выхода: ");
                scanf("%d" , &c );
        }
        for (int i=0;i<length;i++)
        {
            free(names[i]);
            free(authors[i]);
        }
        free(names);
        free(authors);
        free(years);
        free(name_for_push);
        free(author_for_push);
        free(name_for_remove);

        return 0;
}
```

2)ТЕЛО ФУНКЦИИ MAIN.H

```
#pragma once
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct MusicalComposition
{
	char* name;
	char* author;
	int year;
	struct MusicalComposition* next;
	struct MusicalComposition* prev;
} MusicalComposition;
MusicalComposition* createMusicalComposition(char* name, char* author,int year)
{
	MusicalComposition* element_for_push = (MusicalComposition*)malloc(sizeof(MusicalComposition));
	element_for_push->name = (char*)malloc(sizeof(char)*81);
	element_for_push->author = (char*)malloc(sizeof(char)*81);
	strcpy(element_for_push->name , name );
	strcpy(element_for_push->author,author);
	element_for_push->year = year;
	element_for_push->next = NULL;
	element_for_push->prev = NULL;
	return element_for_push;
}
void push(MusicalComposition* head, MusicalComposition* element)
{
	MusicalComposition* tmp;
	if ( head->next == NULL )
	{
		element->next = NULL;
		element->prev = head;
		head->next = element;
		return;
	}
	tmp = head->next;
	while (tmp->next)
	{
		tmp = tmp->next;
	}
	element->next = NULL;
	element->prev = tmp;
	tmp->next = element;
}
MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int*
array_years,int length)
{
	if ( length == 0 )
	    return NULL;
	MusicalComposition* head  = (MusicalComposition*)malloc(sizeof(MusicalComposition));
	head->name = (char*)malloc(sizeof(char)*81);
	head->author = (char*)malloc(sizeof(char)*81);
	strcpy(head->name , array_names[0]);
	strcpy(head->author , array_authors[0]);
	head->year = array_years[0];
	MusicalComposition* tmp;

	 for ( int i = 1; i < length ; i++ )
	{
```

```
                tmp = createMusicalComposition( array_names[i] , array_authors[i] , array_years[i] );
                push(head , tmp);
        }
        return head;
}
void removeEl(MusicalComposition* head, char* name_for_remove)
{
        MusicalComposition *tmp ;
        tmp = head;
        while ( tmp )
        {
                if ( strcmp( tmp->name , name_for_remove ) == 0 )
                {
                        tmp->next->prev = tmp->prev;
                        tmp->prev->next = tmp->next;
                        free(tmp);
                }
                tmp = tmp->next;
        }
}
int count(MusicalComposition* head)
{
        int i=1;
        MusicalComposition* tmp;
        tmp = head->next;
        while (tmp)
        {
                tmp = tmp->next;
                i++;
        }
         free(tmp);
        return i;
}
void print_names(MusicalComposition* head)
{
         MusicalComposition* tmp = (MusicalComposition*)malloc(sizeof(MusicalComposition));
        tmp = head->next;
                printf( "%s\n" , head->name );
                printf( "%s\n" , tmp->name );
        while (tmp->next)
        {
                if ( tmp->next->year != -1 )
                        printf("%s\n"  , tmp->next->name );
                tmp = tmp->next;
        }
}
void print_srez_spiska( MusicalComposition* head , int start , int end )
{
         MusicalComposition* tmp = (MusicalComposition*)malloc(sizeof(MusicalComposition));
         tmp = head->next;

         if ( start == 0 )
         {
                   printf("название музыкальной композиции [0]: %s "  , head ->name );
                printf("автор музыкальной композиции [0]: %s "  , head -> author );
                printf("год создания музыкальной композиции [0]: %d\n "  ,head->year );
         }
         int i=1;
         for ( ; i < start ; i++ )
                   tmp = tmp->next;
```

```
        for ( ; i <= end ; i++ )
        {
                printf("название музыкальной композиции [%d]: %s " , i , tmp ->name );
                printf("автор музыкальной композиции [%d]: %s " , i , tmp -> author );
                printf("год создания музыкальной композиции [%d]: %d\n " , i ,tmp->year );
                tmp = tmp->next;
        }
        free(tmp);
}
```