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

Студентка гр.7381 _____ Кревчик А.Б.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2017

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Кревчик А.Б.

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.

Создать функцию, которая удаляет элементы списка, в поле author которых присутствует некоторая подстрока(подаётся на вход функции).

Предполагаемый объем пояснительной

записки: Не менее 10 страниц.

Дата выдачи задания: 23.11.2017

Дата сдачи реферата:

Дата защиты реферата:

Студент _____ Кревчик А.Б.

Преподаватель _____ Берленко Т. А.

АННОТАЦИЯ

Реализована программа на языке СИ. В результате выполнения программы был создан двунаправленный линейный список и функции для работы с ним, а именно удаление элемента, добавление элемента в конец списка, подсчёт количества элементов списка, вывод полей элементов списка.

SUMMARY

The program was done with usage C language. As a result of acomplising the program a bidirectional linear list and functions to work with it were made, specifically deleting of the element, adding the element to the end of the list, counting the number of the list's elements, outputting fields of the list's elements.

СОДЕРЖАНИЕ

Введение

Целью данной работы является закрепление знаний о структурах в языке СИ, выделении и очистке динамической памяти, написать функции для работы со списком.

1. ОПИСАНИЕ ТЕЛА ФУНКЦИИ MAIN.C

1.1. При помощи функции ввода общего значения scanf вводится количество количество композиций элементов, составляющих список.

1.2. Динамически выделяется память под названия композиций, имена авторов, и год создания композиции.

1.3. Заполняются данные списка.

1.4. При помощи функции createMusicalComposition создаётся двунаправленный список по заполненным данным.

1.5. Далее пользователю предлагается набор функций для работы со списком:

0-добавление элемент в конец списка;

1-удаление элемента с определённым названием композиции;

2-вывод количества элементов в списке;

3-вывод названий всех музыкальных композиций;

4-удаление элемента, в поле ‘автор’ которых есть некоторая подстрока;

5-выход из программы.

1.6. Освобождается динамическая память, выделенная для работы со списком.

2. ОПИСАНИЕ ФУНКЦИЙ ДЛЯ РАБОТЫ СО СПИСКОМ

2.1**CreateMusicalCompositionList**

На вход функция получает количество композиций и

указатели на три массива: массив названий ,массив авторов и массив годов созданий В результате работы функция формирует линейный двунаправленный список. Функция возвращает указатель на первый элемент списка.

2.2.**Push**

Данной функции передается указатель на голову списка и элемент, который необходимо добавить. Функция соединяет последний и новый элементы списка посредством указателей, если же список был пуст, то указатель на голову теперь указывает на данный элемент. Функция ничего не возвращает.

2.3. **RemoveEl**

На вход программе подаются: указатель на первый элемент и указатель на имя элемента для удаления. С помощью функции strcmp список проверяется на совпадение имя композиции с именем композиции для удаления. Функция ничего не возвращает.

2.4.**Count**

Функция получает указатель на первый элемент списка и с помощью цикла for подсчитывает количество элементов в нем. Функция возвращает количество элементов

2.5. **Print_names**

Данной функции передается указатель на первый элемент списка. Переходя от

одного элемента списка к другому, функция выводит названия композиций.

Функция ничего не возвращает.

2.6. **Substring**

На вход программе подаются: указатель на первый элемент и указатель на подстроку для поиска. С помощью функции strstr список проверяется на наличие в имени автора подстроки для удаления. Функция ничего не возвращает.

Заключение

В ходе выполнения данной курсовой работы был создан двунаправленный линейный список, функции для работы с ним и удобный интерфейс для пользования. Были закреплены знания по выделению и очищению динамической памяти, получены необходимые навыки для создания списков функций для работы с ними.

Список использованных источников

Б. Керниган, Д. Ритчи «Язык программирования Си»

Приложение А. Исходный код программы

Файл «main.c»

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "f.h"

int main(){

int length;
char string[80];
printf("Введите количество элементов списка\n");
scanf("%d", &length);

char** names = (char**)malloc(sizeof(char*)*length);
char** authors = (char**)malloc(sizeof(char*)*length);
int* years = (int*)malloc(sizeof(int)*length);


if (length==0)
   printf("Вы создали пустой список!\n");
else{
    for (int i=0;i<length;i++) {
      getchar();
      char name[80];
      char author[80];

      printf("Введите название музыкальной композиции\n");
      fgets(name, 80, stdin);
      printf("Введите имя автора композиции\n");
      fgets(author, 80, stdin);
      printf("Введите год создания музыкальной композиции\n");
      scanf( "%d", &years[i]);

      (name,"\n")=0;
      (author,"\n")=0;

      names[i] = (char*)malloc(sizeof(char*) * (strlen(name)+1));
      authors[i] = (char*)malloc(sizeof(char*) * (strlen(author)+1));

      strcpy(names[i], name);
      strcpy(authors[i], author);
      }
```

```
}


MusicalComposition* head = createMusicalCompositionList(names, authors,
years, length);
char name_for_push[80];
char author_for_push[80];
int year_for_push;

char name_for_remove[80];

int choice,flag=0;
int k;


while ( flag == 0 )
{
printf("Выберите, что вы хотите сделать со списком:\n");
printf("0 - Добавить элемент в конец списка. \n");
printf("1 - Удалить элемент с определённым названием композиций. \n");
printf("2 - Вывести количество элементов в списке. \n");
printf("3 - Вывести название всех музыкальных композиций из списка.\n");
printf("4 - Удаить элемент, в поле 'автор' которого есть подстрока.\n");
printf("5 - Выход.\n");


scanf("%d" , &choice);
switch(choice)
{
case 0:
    getchar();
    printf("Введите название добавляемой композиции\n" );
    fgets(name_for_push, 80, stdin);
    printf("Введите автора добавляемой композиции\n" );
    fgets(author_for_push, 80, stdin);
    printf("Введите год создания музыкальной композиции\n" );
    scanf( "%d", &year_for_push);
    (name_for_push,"\n")=0;
    (author_for_push,"\n")=0;
     MusicalComposition* element_for_push =
createMusicalComposition(name_for_push, author_for_push, year_for_push);
     getchar();
     push(head, element_for_push);
     break;
```

```
case 1:
      getchar();
      printf("Введите название композиции, которое нужно убрать из
списка.\n");
     fgets(name_for_remove, 80, stdin);
     (name_for_remove,"\n")=0;
      removeEl(&head, name_for_remove);
      break;

case 2:
     k = count(head);
     printf("количество элементов в списке %d\n", k);
     break;

case 3:
     print_names(head);
     break;

case 4:
     getchar();
     printf("Введите подстроку для удаления\n");
     fgets(string,80,stdin);
     (string,"\n")=0;
     substring(&head,string);
     break;

case 5:
     flag=1;
     break;

default:
     printf("Неверный ввод\n");
}

}

for (int i=0;i<length;i++){
free(names[i]);
free(authors[i]);
}
free(names);
free(authors);
free(years);
```

```
    return 0;
}
```

Файл «f.c»

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

typedef struct MusicalComposition
{
char* name;
char* author;
int year;
struct MusicalComposition* next;
struct MusicalComposition* prev;
}MusicalComposition;// Описание структуры MusicalComposition

// Создание структуры MusicalComposition

MusicalComposition* createMusicalComposition(char* name, char* author,int year)
{ MusicalComposition* Musical_Composition;
Musical_Composition=(MusicalComposition*)malloc(sizeof(MusicalComposition));
Musical_Composition->name=name;
Musical_Composition->author=author;
Musical_Composition->year=year;
Musical_Composition->prev=NULL;
Musical_Composition->next=NULL;
return Musical_Composition;
}

// Функции для работы со списком MusicalComposition

MusicalComposition* createMusicalCompositionList(char** array_names, char**
array_authors, int* array_years, int n)
{ int i=0;


MusicalComposition* head=NULL, *tmp, *pred=NULL;

for(i=0; i<n;i++)
{tmp = createMusicalComposition ( array_names[i], array_authors[i],array_years[i]);
tmp->prev=pred;
```

```
if (i!=0)
pred->next=tmp;


else

head=tmp;

pred=tmp;
}
return head;
}

void push(MusicalComposition* head, MusicalComposition* element)
{if (head==NULL)
*head=*element;
else{
for(;head->next!=NULL;head=head->next);

head->next=element;
element->prev=head;
}
}

void removeEl(MusicalComposition** head_p, char* name_for_remove)
{
MusicalComposition* head = *head_p;

for (;head!=NULL;head=head->next)
if (strcmp(head->name,name_for_remove)==0)
if (head->next==NULL && head->prev==NULL)
{*head_p=NULL;
free(head);
}
else if(head->next!=NULL && head->prev==NULL)
{head->next->prev=NULL;
*head_p=head->next;
free(head);
}
else if (head->next==NULL && head->prev!=NULL)
{head->prev->next=NULL;
free (head);
}
```

```
else
{head->next->prev=head->prev;
head->prev->next=head->next;
free (head);
}



}

int count(MusicalComposition* head)
{int k=0;
for (;head!=NULL;head=head->next)
k++;
return k;
}

void print_names(MusicalComposition* head)
{
for (;head!=NULL;head=head->next)
puts(head->name);
}

void substring(MusicalComposition** head_p, char* string)
{
MusicalComposition* head = *head_p;

for (;head!=NULL;head=head->next)
if (strstr(head->author,string)!=NULL)
if (head->next==NULL && head->prev==NULL)
{*head_p=NULL;
free(head);
}
else if(head->next!=NULL && head->prev==NULL)
{head->next->prev=NULL;
*head_p=head->next;
free(head);
}
else if (head->next==NULL && head->prev!=NULL)
{head->prev->next=NULL;
free (head);
}
else
{head->next->prev=head->prev;
```

```
head->prev->next=head->next;
free (head);
}
}
```

Файл «f.h»

```
typedef struct MusicalComposition
{
char* name;
char* author;
int year;
struct MusicalComposition* next;
struct MusicalComposition* prev;
}MusicalComposition;// Описание структуры MusicalComposition

MusicalComposition* createMusicalComposition(char* name, char* author,int year);
MusicalComposition* createMusicalCompositionList(char** array_names, char**
array_authors, int* array_years, int n);
void push(MusicalComposition* head, MusicalComposition* element);
void removeEl(MusicalComposition** head_p, char* name_for_remove);
int count(MusicalComposition* head);
void print_names(MusicalComposition* head);
void substring(MusicalComposition** head_p, char* string);
```