**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

**Тема: Линейные списки**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3344 |  | Коняева М.В. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Целью работы является освоение работы с линейными двусвязными списками в языке Си на примере использующей их программы.

## Задание

Создайте двунаправленный список музыкальных композиций *MusicalComposition* и *api* (*application programming interface* - 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.

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.

В функции *main* написана некоторая последовательность вызова команд для проверки работы вашего списка.

## Выполнение работы

Подключим стандартные библиотеки *stdio.h* для работы с вводом, *string.h* для работы со строками, а также *stdlib.h* для работы с памятью. Описываем поля структуры *MusicalComposition*, воспользуемся конструкцией *typedef* для того, чтобы сократить прописывание словосочетания *struct MusicalCompositon* каждый раз*.*

1) Функция *MusicalComposition\* createMusicalComposition(char\* name, char\* autor, int year)*

Данная функция принимает на вход две строки и число. Выделяем память с помощью функции *malloc* под создаваемую структуру, и присваиваем полям нужные данные. Поля *next* и *prev*для связи элементов списка устанавливаются в *NULL*. Функция возвращает сформированную структуру.

2) Функция *MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n)*

Данная функция принимает на вход массивы строк – названий композиций и авторов, массив чисел – года созданий, а также длину этих массивов. Сделаем «голову» списка, для этого создаем структуру с помощью ранее описанной функции *MusicalComposition\* createMusicalComposition.* Оставшиеся элементы создаются и «связываются» с помощью функции *push* в теле цикла *for*. По окончании работы цикла, возвращается указатель на начало списка.

3) Функция *push (MusicalComposition\* head, MusicalComposition\* element)*

Данная функция принимает на вход начало списка, а также структуру, которую нужно добавить в конец списка. Создаем временную переменную *MusicalComposition\* tmp* для перемещения по списку. Доходим до конца списка с помощью цикла *while*. Для конечного элемента присваиваем полю *next* указатель добавляемого элемента, а полю *prev* уже у него самого найденный конечный элемент списка, полю *next* присваивается значение *NULL*.

4) Функция *void removeEl(MusicalComposition\* head, char\* name\_for\_remove)*

Данная функция принимает на вход начало списка, а также строку, которую нужно исключить из списка. Создаем временную переменную *MusicalComposition\* tmp* для перемещения по списку. В цикле *while* проходимся по элементам списка и сравниваем строки с помощью функции *strcmp*. Если найдено совпадение, связываются предыдущий и следующий за удаленным элементом. Отдельно проверяются конечный элемент списка, для присваивания значения *NULL*. Очищаем память выделенную под удаляемый элемент.

5) Функция *int count(MusicalComposition\* head)*

Данная функция принимает на вход начало списка. Создаем временную переменную *MusicalComposition\* tmp* для перемещения по списку и переменную *int count = 0* для подсчета количества элементов в списке. В цикле *while* проходимся по элементам списка и увеличиваем счетчик. Функция возвращает *count* - количество элементов в списке.

6) Функция *void print\_names(MusicalComposition\* head)*

Данная функция принимает на вход начало списка. Создаем временную переменную *MusicalComposition\* tmp* для перемещения по списку. В цикле *while* проходимся по элементам списка и выводим поля структуры.

7) Функция *main()*

В данной функции прописана основная логика программы, которая была заранее заготовлена в задании.

Разработанный программный код см. в приложении А. Результаты тестирования см. в приложении Б.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | 7  Fields of Gold  Sting  1993  In the Army Now  Status Quo  1986  Mixed Emotions  The Rolling Stones  1989  Billie Jean  Michael Jackson  1983  Seek and Destroy  Metallica  1982  Wicked Game  Chris Isaak  1989  Points of Authority  Linkin Park  2000  Sonne  Rammstein  2001  Points of Authority | Fields of Gold Sting 1993  7  8  Fields of Gold  In the Army Now  Mixed Emotions  Billie Jean  Seek and Destroy  Wicked Game  Sonne  7 | Данные обработаны корректно. |
|  | 2  Wicked Game  Chris Isaak  1989  Points of Authority  Linkin Park  2000  Sonne  Rammstein  2001  Sonne | 2 3 Wicked Game Points of Authority 2 | Данные обработаны корректно. |

## Выводы

Были изучена работа с линейными списками. Также была создана программа, в которой реализованы различные действия с двусвязными списками и их элементами (создание списка, добавление элемента, удаление элемента по параметру, подсчет количества элементов).

# Приложение А Исходный код программы

Название файла: lb2.c

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

// Описание структуры MusicalComposition

typedef struct MusicalComposition {

int n;

char\* name;

char\* author;

int year;

struct MusicalComposition \*next;

struct MusicalComposition \*prev;

} MusicalComposition;

// Создание структуры MusicalComposition

MusicalComposition\* createMusicalComposition(char\* name, char\* autor,int year);

// Функции для работы со списком MusicalComposition

MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n);

void push(MusicalComposition\* head, MusicalComposition\* element);

void removeEl(MusicalComposition\* head, char\* name\_for\_remove);

int count(MusicalComposition\* head);

void print\_names(MusicalComposition\* head);

int main(){

int length;

scanf("%d\n", &length);

char\*\* names = (char\*\*)malloc(sizeof(char\*)\*length);

char\*\* authors = (char\*\*)malloc(sizeof(char\*)\*length);

int\* years = (int\*)malloc(sizeof(int)\*length);

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

{

char name[80];

char author[80];

fgets(name, 80, stdin);

fgets(author, 80, stdin);

fscanf(stdin, "%d\n", &years[i]);

(\*strstr(name,"\n"))=0;

(\*strstr(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];

fgets(name\_for\_push, 80, stdin);

fgets(author\_for\_push, 80, stdin);

fscanf(stdin, "%d\n", &year\_for\_push);

(\*strstr(name\_for\_push,"\n"))=0;

(\*strstr(author\_for\_push,"\n"))=0;

MusicalComposition\* element\_for\_push = createMusicalComposition(name\_for\_push, author\_for\_push, year\_for\_push);

fgets(name\_for\_remove, 80, stdin);

(\*strstr(name\_for\_remove,"\n"))=0;

printf("%s %s %d\n", head->name, head->author, head->year);

int k = count(head);

printf("%d\n", k);

push(head, element\_for\_push);

k = count(head);

printf("%d\n", k);

removeEl(head, name\_for\_remove);

print\_names(head);

k = count(head);

printf("%d\n", k);

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

free(names[i]);

free(authors[i]);

}

free(names);

free(authors);

free(years);

return 0;

}

MusicalComposition\* createMusicalComposition(char\* name, char\* author,int year){

struct MusicalComposition \*item = (struct MusicalComposition\*) malloc(sizeof(struct MusicalComposition));

item->year = year;

item->name = name;

item->author = author;

item->next = NULL;

item->prev = NULL;

return item;

}

MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n){

MusicalComposition \*head = createMusicalComposition(array\_names[0], array\_authors[0], array\_years[0]);

head->prev = NULL;

for (int i = 1; i < n; i++){

MusicalComposition\* element = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);

push(head,element);

}

return head;

}

void push(MusicalComposition\* head, MusicalComposition\* element){

MusicalComposition\* tmp = head;

while(tmp->next){

tmp = tmp->next;

}

tmp->next = element;

element->prev = tmp;

element->next = NULL;

return;

}

void removeEl(MusicalComposition\* head, char\* name\_for\_remove){

MusicalComposition\* tmp = head;

while (tmp->next){

if (strcmp(tmp->name, name\_for\_remove) == 0){

if (tmp->next == NULL) {

tmp->next = NULL;

} else {

tmp->next->prev = tmp->prev;

tmp->prev->next = tmp->next;

}

free(tmp);

break;

}

tmp = tmp->next;

}

return;

}

int count(MusicalComposition\* head){

MusicalComposition\* tmp = head;

int count = 0;

while(tmp){

tmp= tmp->next;

count++;

}

return count;

}

void print\_names(MusicalComposition\* head){

MusicalComposition\* tmp = head;

while(tmp){

printf("%s\n",tmp->name);

tmp = tmp->next;

}

return;

}