**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

|  |  |  |
| --- | --- | --- |
| Студент гр. 3341 |  | Моисеева А.Е. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Научиться работе с линейными односвязными и двусвязными списками в языке Си и использовании их в программном коде.

Для этого необходимо:

* изучить теоретические сведения о линейных списках
* ознакомиться с базовыми операциями над списками: вставка, удаление, объявление, вывод, поиск элементов
* освоить создание структур в языке Си для реализации линейных списков
* научиться применять односвязные и двусвязные списки в коде программы
* разработать программу, которая будет решать индивидуальные задачи по работе с данными – списком музыкальных композиций с помощью двусвязного списка

## 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.

!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.

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

Функцию *main* менять не нужно.

## Выполнение работы

Объявлена структура *MusicalComposition*, состоящая из:

* *char\* name -* название песни
* *char\* author –* автор песни
* int *year -* года выпуска песни
* *struct* *MusicalComposition\* next -* указателя на следующую песню
* *struct* *MusicalComposition\* prev -* указателя на предыдущую песню

Функции:

* *MusicalComposition\* createMusicalComposition(char\* name, char\* author, int year)* принимает указатель на название, автора и год выпуска композиции, выделяется память для создания элемента списка с помощью *malloc* выделяет*,* полям *name, author, year* присваиваются значения, подающиеся на вход функции, поля *prev, next* инициализируются как *NULL*, возвращается указатель на созданную песню.
* *MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n)* принимает указатель на массив названий, авторов, дат выпуска и количество песен; если количество песен *n* равно нулю, то возвращается *NULL*;с помощью функции *createMusicalComposition* создаётся начальный элемент списка, полям *name*, *author*, *year* которого присваиваются начальные значения массивов *array\_names*, *array\_authors*, *array\_years* соответственно, создаётся элемент списка *current* – текущая музыкальная композиция, вначале ей присваивается значение *head*, затем через цикл создаётся элемент *next\_composition*, поля *name*, *author* и *year* которого соответствуют *i*-ым элементам массивов, подающихся на вход функции. Полю *next* элемента current присваивается указатель на *next\_composition*, а полю *prev* элемента *next\_composition* присваивается значение *current*. Текущей композиции присваивается указатель на следующую композицию. Возвращается указатель на *head* – начальный элемент списка.
* *void push(MusicalComposition\* head, MusicalComposition\* element)* принимает указатель на начальный элемент списка и песню для её включения в список;если *head* пуст, то происходит выход из функции; объявляется элемент *current* – текущая песня, ему присваивается значение *head*, с помощью условия цикл *while* проходит по списку до его конца, т.е. пока поле *next* текущего элемента не равно *NULL;* после цикла полю *next* последней в списке песни присваивается *element*, полю *prev* добавленной песни присваивается *current.*
* *void removeEl (MusicalComposition\* head, char\* name\_for\_remove)* принимает указатель на начальный элемент списка и на название композиции для удаления из списка; объявляется элемент *current* – текущая композиция; если *head* пуст, то происходит выход из функции; с помощью условия цикл *while* доходит до композиции, которую необходимо удалить, т. е. пока *strcmp* не обнаружит совпадения поля *name* текущего элемента и имени для удаления; после цикла при обнаружении совпадения полю *next* предыдущей песни до текущей присваивается указатель на следующую песню, а полю *prev* следующей песни присваивается указатель на предыдущую песню до текущей (устанавливаются связи между элементами до и после удалённой песни). Освобождается память, выделенная для хранения текущего предложения.
* *int count(MusicalComposition\* head)* принимает указатель на начальный элемент списка;переменной *number\_of\_elements*, обозначающей количество песен, присваивается 0. Затем с помощью цикла *while* проходим до конца списка, пока указатель на current не равен *NULL* и увеличиваем счётчик. Возвращается количество элементов списка.
* void *print\_names(MusicalComposition\* head)* принимает указатель на начальный элемент списка; с помощью цикла *while* проходим до конца списка и выводим названия песен(поля name элементов списка), пока указатель на *current* не равен *NULL.*
* *main* содержит программный код, состоящий из неких команд для проверки корректности работы остальных функций.

Разработанный программный код см. в приложении А.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | 2  Mixed Emotions  The Rolling Stones  1989  Billie Jean  Michael Jackson  1983  Sonne  Rammstein  2001  Billie Jean | Mixed Emotions The Rolling Stones 1989  2  3  Mixed Emotions  Sonne  2 | Программа считывает 2 элемента списка, затем песню, которую нужно добавить и название песни, которую нужно удалить. Выводится название, автор и год создания первой песни, количество элементов списка, затем добавляется ещё одна песня и выводится новое количество элементов, затем происходит удаление заданной песни и выводятся оставшиеся названия песен и их количество. |
|  | 3  Mixed Emotions  The Rolling Stones  1989  Billie Jean  Michael Jackson  1983  Sonne  Rammstein  2001  Billie Jean  The Rolling Stones  1989  Sonne | Mixed Emotions The Rolling Stones 1989  3  4  Mixed Emotions  Billie Jean  Billie Jean  3 | В список добавляется ещё одна композиция с именем, идентичным одной из уже присутствующих в списке композиций, что заметно при выводе. |

## Выводы

Цель работы достигнута. Изучен теоретический материал по линейным спискам, изучены базовые операции для работы над списками, освоены навыки, необходимые для создания структур в языке Си. В результате с помощью полученных знаний реализована программа, которая принимает на вход список песен с названиями, авторами и годами создания. Через линейный двусвязный список реализовано создание двусвязного линейного списка, работа с его элементами – вычисление количества, вывод названий, добавление новых элементов списка и удаление элементов по названию, заданному пользователем.

# Приложение А Исходный код программы

Название файла: main.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;

MusicalComposition\* createMusicalComposition(char\*, char\*, int);

MusicalComposition\* createMusicalCompositionList(char\*\*, char\*\*, int\*, int);

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

void removeEl(MusicalComposition\*, char\*);

int count(MusicalComposition\*);

void print\_names(MusicalComposition\*);

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

MusicalComposition\* new\_composition = (MusicalComposition\*)malloc(sizeof(MusicalComposition));

new\_composition->name = name;

new\_composition->author = autor;

new\_composition->year = year;

new\_composition->next = NULL;

new\_composition->prev = NULL;

return new\_composition;

}

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

if (n == 0){

return NULL;

}

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

MusicalComposition\* current = head;

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

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

current->next = next\_composition;

next\_composition->prev = current;

current = next\_composition;

}

return head;

}

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

if (head == NULL)

return;

MusicalComposition\* current = head;

while(current->next)

current = current->next;

current->next = element;

element->prev = current;

}

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

if (head == NULL)

return;

MusicalComposition\* current = head;

while(strcmp(current->name, name\_for\_remove) != 0)

current = current->next;

current->prev->next = current->next;

current->next->prev = current->prev;

free(current);

}

int count(MusicalComposition\* head){

int number\_of\_elements = 0;

MusicalComposition\* current = head;

while(current){

current = current->next;

number\_of\_elements++;

}

return number\_of\_elements;

}

void print\_names(MusicalComposition\* head){

MusicalComposition\* current = head;

while(current){

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

current = current->next;

}

}

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;

}