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: Структуры данных, линейные списки

Студент гр. 1304 _____ Андреев В.В.

Преподаватель _____ Чайка К. В.

Санкт-Петербург
2022

Цель работы.

Изучить структуры в языке Си и применить полученные знания для реализации двусвязного списка.

Задание.

Создайте 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.

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

Функцию main менять не нужно.

Выполнение работы.

Заголовочные файлы:

- stdio.h
- stdlib.h
- string.h

Структуры: Перечень структур представлен в табл. 1

Таблица 1 – Структуры программы

Имя структуры	Поле структуры	Комментарий	Комментарии
Musical Composition	name	Название композиции.	Структура музыкальных композиций. Элементы связаны между собой по принципу двусвязного списка.
	author	Имя автора композиции.	
	year	Год создания композиции.	
	next	Следующая в списке композиция.	
	previous	Предыдущая в списке композиция.	

Функции: Перечень функций представлен в табл. 2

Таблица 2 – Функции программы

Имя функции	Возвращаемое значение	Аргументы	Комментарии
CreateMusical Composition	Инициализированный элемент композиции типа MusicalComposition.	*char* name* — название композиции. *char* autor* — имя автора композиции. *int year* — год создания композиции.	Создает и инициализирует элемент списка композиций.
CreateMusical CompositionList	Указатель на первый элемент в созданном списке композиций.	*char** array_names* – массив названий композиций для добавлений(длинна n). *char** array_authors* – массив имен авторов композиций для. добавлений(длинна n) *int* array_years* – массив дат созданий композиций для добавлений(длинна n). *int n* – количество композиций на добавлений.	Создает список композиций.
push	-	*MusicalComposition* head* - указатель на первый элемент списка композиций. *MusicalComposition* element* – элемент на добавление.	Добавляет в конец списка композиций новый элемент.
removeEl	-	*MusicalComposition* head* - указатель на первый элемент списка композиций. *char* name_for_remove* - строка названия композиции для удаления.	Удаляет из списка композиций первый элемент по названию композиции. Удаленный элемент будет выгружен из памяти, но поля элемента не будут затронуты.
count	Количество композиций в списке.	*MusicalComposition* head* - указатель на первый	Считает количество элементов в списке

		элемент списка композиций.	композиций.
print_names	-	*MusicalComposition* head* - указатель на первый элемент списка композиций.	Выводит в поток вывода названия всех композиций в списке.

Алгоритм работы:

- createMusicalComposition:
  1. Выделяем память под структуру элемента списка композиций.
  2. Инициализируем элемент полученными данными.
- createMusicalCompositionList:
  1. Создаем первый элемент списка(head) и инициализируем его информацией из *array_names, array_authors , array_years* под индексом 0.
  2. Далее добавляем в список n – 1 элемент, информация которых берется из массивов.
- push:
  1. Идем до конца списка, пока *next* существует.
  2. Подсоединяем новый элемент к списку. Поле *next* последнего элемента списка указывает на новый элемент, поле *previous* нового элемента указывает на последний элемент списка.
- removeEl:
  1. Идем по списку пока не встретим элемент с названием == искомому.
  2. Отсоединяем найденный элемент.
     Поле *next* предыдущего за найденным указывает на следующий за найденным.
     Поле *previous* следующего за найденным указывает на предыдущего за найденным.
  3. Если нашли, то завершаем работу.
     Иначе продолжаем идти по списку(*head = head->next;*) пока текущий элемент существует.
- count:
  1. Идем по списку(*head = head→next;)* пока текущий элемент существует.
  2. Увеличиваем счетчик элементов.
- print_names:
  1. Идем по списку(*head = head→next;)* пока текущий элемент существует.
  2. Выводим в поток вывода поле элемента *name*.

Тестирование.

Результаты тестирования представлены в табл. 3.

Таблица 3 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	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	Ответ верный.

Выводы.

Были изучены структуры в языке Си и написана программа, которая реализует двусвязный список. Работа со списком осуществляется через написанный api.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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



//..............MusicalComposition.............................//

typedef struct MusicalComposition
{
    char* name;
    char* author;
    int year;

    struct MusicalComposition* next;
    struct MusicalComposition* previous;
} MusicalComposition;

MusicalComposition* createMusicalComposition(char* name, char* autor,
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, char* name_for_remove);
int count(MusicalComposition* head);
void print_names(MusicalComposition* head);



MusicalComposition* createMusicalComposition
(
      char* name, char* autor, int year
)
{
    if(name == NULL || autor == NULL) return NULL;

    MusicalComposition* Lcomposition =
      (MusicalComposition*)malloc( sizeof(MusicalComposition) );
    LComposition->name = name;
    LComposition->author = autor;
    LComposition->year = year;
    LComposition->next = NULL;
    LComposition->previous = NULL;

    return LComposition;
}
```

```
MusicalComposition* createMusicalCompositionList
(
      char** array_names, char** array_authors, int* array_years,
      int n
)
{
    if
    (
      array_names == NULL || array_authors == NULL ||
      array_years == NULL || n <= 0
    )
    return NULL;

      MusicalComposition* LHead =
      createMusicalComposition(array_names[0], array_authors[0],
      array_years[0]);
    if(LHead == NULL) return NULL;

    for(int i = 1; i < n; ++i)
    {
      push
      (
            LHead, createMusicalComposition(array_names[i],
            array_authors[i], array_years[i])
      );
    }

    return LHead;
}

void push(MusicalComposition* head, MusicalComposition* element)
{
    if(head == NULL || element == NULL) return;

    while(head->next != NULL)
    {
        head = head->next;
    }

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

void removeEl(MusicalComposition* head, char* name_for_remove)
{
    if(name_for_remove == NULL) return;

    while(head != NULL)
    {
        if(strcmp(head->name, name_for_remove) == 0)
        {
            if(head->previous != NULL)
                 head->previous->next = head->next;
            head->next->previous = head->previous;
            free(head);
            return;
        }
```

```
        head = head->next;
    }
}

int count(MusicalComposition* head)
{
    int Result = 0;
    while(head != NULL)
    {
        ++Result;
        head = head->next;
    }

    return Result;
}

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

      }
```