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 Ефремов А.А.

Преподаватель Чайка К.В.

Санкт-Петербург

2022

Цель работы.

Изучить структуры данных, линейные списки в языке С и применить полученные знания для реализации двунаправленного 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.

В ходе программы используются следующие заголовочные файлы:

- stdlib.h
- stdio.
- string.h

Была описана структура MusicalComposition, которая содержит элементы описания музыкальных композиций, а также указатели на предыдущую и следующую композиции.

Были реализованы следующие функции:

- MusicalComposition* createMusicalComposition(char* name, char* author,int year) — функция принимает название композиции, автора и год выхода и создает на основе полученных данных структуру MusicalComposition.
- MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n) — функция принимает списки названий композиций, авторов и годов выхода и на основе полученных данных, вызывая функцию createMusicalComposition, создает список композиций, связывая их между собой через указатели next и previous, хранящиеся в поле структуры MusicalComposition.
- void push(MusicalComposition* head, MusicalComposition* element) — функция принимает указатель на начало списка и на новую структуру композиции и добавляет ее в конец списка. Для этого, путем перехода от одного элемента списка к следующему до тех пор, пока указатель на следующую композицию не равен NULL, происходит поиск последней композиции в списке. Далее указатель next последней композиции приравнивается к указателю на добавляемую композицию, а указатель previous добавляемой композиции приравнивается к указателю на последнюю композицию из списка, тем самым «связывая» композиции.
- void removeEl(MusicalComposition* head, char* name_for_remove) — функция принимает указатель на начало списка и название композиции, которую необходимо удалить. Для этого происходит пробег по списку, описанный ранее, и поочередно сравнивается при помощи функции strcmp названия композиций. Если названия совпадают, то указатель next предыдущей композиции приравнивается к указателю на

следующую от удаляемой композицию, а указатель previous следующей композиции приравнивается к указателю на предыдущую от удаляемой композицию. Таким образом происходит «связывание» соседних от удаляемой композиций. Далее память, выделенная на удаляемую композицию, очищается при помощи функции free.

- int count(MusicalComposition* head) — функция принимает указатель на начало списка и возвращает значение количества композиций в списке. Для это используется переменная counter типа int, повышающая свое значение каждый раз, когда выполняется цикл пробега по списку, описанный ранее.
- void print_names(MusicalComposition* head) — функция принимает указатель на начало списка и выводит в консоль названия всех композиций в списке. Для этого выполняется пробег по списку, описанный ранее, и с помощью функции printf печатается название композиции, хранящееся в поле структуры MusicalComposition в переменной name типа char*.

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
1.	7 Fields of Gold Sting 1993 In the Army Now Status Quo 1986 Mixed Emotions The Rolling Stones 1989 Billie Jean	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	Успешно

	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		

Выводы.

Были изучены структуры данных, линейные списки в языке С.

В ходе выполнения лабораторной работы была написана программа, реализующая создание двухстороннего списка на основе получаемых от пользователя данных. Работа со списком осуществляется через api.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Efremov_Artom_lb2.c

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


// Описание структуры MusicalComposition
typedef struct MusicalComposition{
char* name;
char* author;
int year;
struct MusicalComposition* next;
struct MusicalComposition* previous;
}MusicalComposition;//теперь эта структура называется просто Musical Composition

// Создание структуры MusicalComposition

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

// Функции для работы со списком MusicalComposition

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]);
MusicalComposition* tmp = createMusicalComposition(array_names[1], array_authors[1], array_years[1]); //выделили память под вторую композицию
head->next = tmp;
tmp->previous = head;
for (int i = 2; i < n; i++){
tmp->next = createMusicalComposition(array_names[i], array_authors[i], array_years[i]);
tmp->next->next = NULL;
tmp->next->previous = tmp;
tmp = tmp->next;
}
```

```
return head;
}

void push(MusicalComposition* head, MusicalComposition* element){
MusicalComposition* tmp = head;
while(tmp->next != NULL)
tmp = tmp->next;
tmp->next = element;
element->next=NULL;
element->previous = tmp;
}

void removeEl(MusicalComposition* head, char* name_for_remove){
MusicalComposition* tmp = head;
while (strcmp(tmp->name, name_for_remove))
tmp = tmp->next;
tmp->previous->next = tmp->next;
tmp->next->previous = tmp->previous;
free(tmp);
}

int count(MusicalComposition* head){
int counter = 1;
MusicalComposition* tmp = head->next;
while(tmp != NULL){
counter++;
tmp = tmp->next;
}
return counter;
}

void print_names(MusicalComposition* head){
MusicalComposition* tmp = head;
while(tmp != NULL){
printf("%s\n", tmp->name);
tmp = tmp->next;
}
}


int main(){
int length;
scanf("%d\n", &length); //считал длину

char** names = (char**)malloc(sizeof(char*)*length);//массив имен
char**  authors  =  (char**)malloc(sizeof(char*)*length);//массив
авторов
int* years = (int*)malloc(sizeof(int)*length);//массив годов

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; //заменяю конец строки нулем?
      (*strstr(author,"\n"))=0;

      names[i]   =   (char*)malloc(sizeof(char*)   *   (strlen(name)+1));
//выделение нужного количества памяти
      authors[i] = (char*)malloc(sizeof(char*) * (strlen(author)+1));

      strcpy(names[i], name); //заполнение массива
      strcpy(authors[i], author);

      }
      MusicalComposition*  head  =  createMusicalCompositionList(names,
authors, years, length); //первый элемент списка
      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;

}
```