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. 2384 _____ Кузьминых Е.М.

Преподаватель _____ Гаврилов А.В.

Санкт-Петербург
2023

Цель работы.

Изучить работу с линейными списками на языке программирования Си.

Задачи.

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 M*usicalComposition*):

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 начале создадим структуру *MusicalComposicion*, в которой будут храниться поля name, author, year, хранящие название композиции, имя автора и год выхода композиции, а также поля next и last типа *MusicalComposition**, в которых будут храниться указатели на предыдущий и следующий элемент в двунаправленном линейном списке. Также для удобства переопределим имя структуры с помощью *typedef*.

Создадим функцию *createMusicalComposition()*, возвращающую указатель на *MusicalComposition* и принимающую в качестве аргументов строку name, *author*,*year*. Функция создает указатель на экземпляр структуры *new_music*, в

поля которого записывает те данные, которые были переданы в качестве аргументов функции *createMusicalComposition()*. Значения *new_music->next*, *new_music->last* ставим равными *NULL*. Функция возвращает созданный указатель на экземпляр структуры.

Функция *push()* принимает в качестве аргумента указатель на первый элемент двухсвязного списка head и указатель на элемент структуры *MusicalComposition*. Пока следующий элемент для *head* не равен *NULL*, функция сдвигает head до конца списка, после на место последнего элемента записывается указатель, полученный в качестве аргумента функции.

Функция *createMusicalCompositionList()* принимает в качестве аргументов 2 массива строк (название композиций и имя автора), а также массив чисел (год издания композиции) и число n – количество элементов в каждом из элементов. В начале создадим указатель на экземпляр структуры *MusicalComposition head*, в качестве значений передадим значения из массивов *array_names*, *array_authors*, *array_years* под нулевым индексом. После с помощью цикла создаем указатель на экземпляр структуры и с помощью функции *push()* добавляем этот указатель в двухсвязный список. Функция возвращает указатель на первый элемент созданного двухсвязного списка.

Функция count принимает в качестве аргументов указатель на первый элемент двухсвязного списка. В функции создается переменная count, которая будет хранить количество элементов в списке. С помощью цикла for мы проходимся по каждому указателю из списка, и пока он не равен *NULL* увеличиваем счетчик count. Функция возвращает count.

Функция *print_names()* принимает указатель на первый элемент двухсвязного списка. Как и в функции *count()* мы с помощью цикла for перебираем все указатели в списке и пока указатель не равен *NULL* печатаем с помощью *printf()* значение name для итерируемого указателя на структуру *MusicalComposition*.

Функция *removeEl()* принимает в качестве аргументов указатель на первый элемент двухсвязного списка и строку. Функция перебирает каждый

элемент в списке помощью цикла for и если поле name совпадает со строкой, полученной в качестве аргумента для функции, то этот элемент удаляется из списка путем изменения указателя на предыдущий и следующий элемент для соседних элементов списка.

Тестирование.

№	Входные данные	Выходные данные	Комментарий
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

Вывод.

В ходе выполнения лабораторной работы была написана программа, в которой была создан двунаправленный список музыкальных композиций *MusicalComposition* и api (в нашем случае набор функций) для работы с этим списком.

Приложение А.

Исходный код программы.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
typedef struct  MusicalComposition{
     char name[80];
     char author[80];
     int year;
     struct MusicalComposition* next;
     struct MusicalComposition* last;
}MusicalComposition;

// Описание структуры 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){
```

```
          MusicalComposition*                                    new_music
=malloc(sizeof(MusicalComposition));
          strcpy(new_music->name,name);
          strcpy(new_music->author,author);
          new_music->year=year;
          new_music->next=NULL;
          new_music->last=NULL;
          return new_music;
     }
     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->last = NULL;
          for (int i = 1; i < n; i++) {
               MusicalComposition*                tmp                 =
createMusicalComposition(array_names[i],array_authors[i],array_years[i]);
               push(head,tmp);
          }
          return head;
     }

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

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

```
}
void print_names(MusicalComposition*  head){
      //int c = count(head);
      for (MusicalComposition* i = head; i!=NULL ; i=i->next) {
            printf("%s\n",i->name);
      }
}
void removeEl (MusicalComposition*  head, char* name_for_remove){
      for (MusicalComposition* i = head; i!=NULL ; i=i->next) {
            if(strcmp(i->name,name_for_remove)==0){
                  i->last->next=i->next;
                  i->next->last=i->last;
            }

      }
}
```