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. 1304 _____ Кривоченко Д. И.

Преподаватель _____ Чайка К. В.

Санкт-Петербург

2022

Цель работы.

Научиться работать с линейными списками

Задание.

Вариант 3.

Напишите программу, на вход которой подается массив целых чисел длины 1000.

Программа должна совершать следующие действия:

- отсортировать массив с помощью алгоритма "сортировка пузырьком"
- посчитать время, за которое будет совершена сортировка, используя при этом функцию стандартной библиотеки
- отсортировать массив с помощью алгоритма "быстрая сортировка" (quick sort), используя при этом функцию стандартной библиотеки
- посчитать время, за которое будет совершена сортировка, используя при этом функцию стандартной библиотеки
- вывести отсортированный массив (элементы массива должны быть разделены пробелом)
- вывести время, за которое была совершена сортировка пузырьком
- вывести время, за которое была совершена быстрая сортировка

Отсортированный массив, время сортировки пузырьком, время быстрой сортировки должны быть выведены с новой строки, при этом элементы массива должны быть разделены пробелами.

Выполнение работы.

Для простоты написания используем оператор typedef для структуры MusicalComposition. У структура по заданию объявляются нужные поля, в том числе указатель next на следующий экземпляр структуры.

CreateMusicalComposition нужна для создания экземпляра структуры.

CreateMusicalCompositionList инициализирует список и возвращает головной элемент.

Функция push добавляет новый элемент в конец списка.

Функция removeEl удаляет элемент с заданным полем name.

Функция count возвращает число элементов в списке.

Функция print_names выводит поля names для каждого элемента списка.

Выводы.

В ходе выполнения лабораторной работы была написана программа, создающая двунаправленный список. Также написаны функции, позволяющие работать с ним.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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



typedef struct MusicalComposition
{
    char name[100];
    char author[100];
    int year;

    struct MusicalComposition *next;



} MusicalComposition;

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

```
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]);
    head->next = tmp;
    for (int i = 2; i<n;i++){
        tmp->next = (MusicalComposition*)malloc(sizeof(MusicalComposition));
        strcpy(tmp->next->name, array_names[i]);
        strcpy(tmp->next->author, array_authors[i]);
        tmp->next->year = array_years[i];
        tmp->next->next = NULL;
        tmp=tmp->next;
    }
   return head;
}


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

```
    void removeEl (MusicalComposition* head, char* name_for_remove){
        MusicalComposition* prev = NULL;
        MusicalComposition* cur = head;
        while (strcmp(cur->name, name_for_remove)!=0){
            prev = cur;
            cur = cur->next;
        }
        prev->next = cur->next;
        free(cur);
    }


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

        return count;
    }


    void print_names(MusicalComposition*  head){
                            MusicalComposition *tmp = (MusicalComposition*)malloc(sizeof(MusicalComposition));
        tmp = head;
        while(tmp!=NULL){
```

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

}
```