МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: «Линейные списки»

Студент гр. 7381 _____ Тарасенко Е.А.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2017

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент: Тарасенко Е. А.

Группа: 7381

Тема работы: Линейные списки

Исходные данные: Двунаправленный список музыкальных композиций, созданный в рамках лабораторной работы №4; набор функций для работы с этим списком.

Содержание пояснительной записки: Введение, содержание, понятие структуры, создание элемента списка и формирование списка в целом, выполнение поставленной задачи.

Объем пояснительной записки:

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студент гр. 7381 _____ Тарасенко Е.А.

Преподаватель _____ Берленко Т.А.

АННОТАЦИЯ

В данной работе требуется на основе уже имеющегося двунаправленного линейного списка музыкальных композиций из четвертой лабораторной работы создать программу, формирующую список композиций, считывая элементы из входного потока и удаляя некоторые элементы списка (если в имени элемента присутствует подстрока, которая вводится пользователем в конце).

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является создание двунаправленного линейного списка музыкальных композиций, который состоит из трех полей: имени автора (названия группы), названия композиции и года ее издания.

Программа должна производить поиск введенной пользователем подстроки в именах авторов. В случае совпадения элемент списка с таким именем должен быть удален. Примечание: задача распространяется на головной и последний элементы списка.

На вход программе подается количество элементов списка, сами элементы и подстрока, которую программа будет искать в именах авторов композиций.

1. Структура в коде. Создание элемента списка

Структура - это совокупность переменных, объединенных одним именем, предоставляющая общепринятый способ совместного хранения информации. Объявление структуры приводит к образованию шаблона, используемого для создания объектов структуры. Переменные, образующие структуру, называются членами структуры. Члены структуры также часто называются элементами или полями.

В данной работе элементы списка – это и есть объекты структуры.

1.1. Структура в коде. Ее объявление и инициализация

```
typedef struct MusicalComposition
{
    char* name;
    char* author;
    int year;
    struct MusicalComposition* next;
    struct MusicalComposition* prev;
} MusicalComposition;
```

Данная структура содержит следующие поля: name – название композиции, author – имя автора, year – год издания, next и prev – указатели на следующий и предыдущий элементы списка (наличие указателя на предыдущий элемент отличает двунаправленный список от однонаправленного, или односвязного). Typedef – спецификатор, упрощающий последующие обращения к объектам структуры.

1.2. Создание элемента списка

```
MusicalComposition* createMusicalComposition(char* name, char* author,int year)
{
    MusicalComposition* element_for_push =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
```

```
    element_for_push->name = (char*)malloc(sizeof(char)*N);
    element_for_push->author = (char*)malloc(sizeof(char)*N);
    strcpy(element_for_push->name, name);
    strcpy(element_for_push->author, author);
    element_for_push->year = year;
    element_for_push->next = NULL;
    element_for_push->prev = NULL;
    return element_for_push;
}
```

2. Формирование списка

Принятие программой числа элементов списка:

```
int length;
printf("Size of your list:\n");
scanf("%d\n", &length);
```

Выделение памяти под двумерные массивы имен авторов и названий песен, под одномерный массив с годами изданий:

```
char** names = (char**)malloc(sizeof(char*)*length);
char** authors = (char**)malloc(sizeof(char*)*length);
int* years = (int*)malloc(sizeof(int)*length);
```

Ввод данных, их обработка:

```
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);
}
```

Создание «головного» элемента списка:

```
MusicalComposition* head = createMusicalCompositionList(names, authors, years, length);
```

Принятие и обработка подстроки для удаления некоторых элементов списка:

```
char name_for_remove[80];
fgets(name_for_remove, 80, stdin);
(*strstr(name_for_remove,"\n"))=0;
```

Далее следует вызов функции удаления элементов, в именах которых найдена принятая подстрока, вывод результатов и освобождение памяти:

```
removeEl(head, name_for_remove);
printf("Total list of authors:\n");
print_names(head);
for (int i=0;i<length;i++)
{
free(names[i]);
free(authors[i]);
}
free(names);
free(authors);
free(years);
```

Создание первого («головного») элемента списка и всех последующих на основе введенных данных:

```
MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int length)
{
    if(length == 0)
      return NULL;
    MusicalComposition* head  = (MusicalComposition*)malloc(sizeof(MusicalComposition));
```

```
    head->name = (char*)malloc(sizeof(char)*N);
    head->author = (char*)malloc(sizeof(char)*N);
    strcpy(head->name , array_names[0]);
    strcpy(head->author , array_authors[0]);
    head->year = array_years[0];
    MusicalComposition* tmp;
    for(int i=1; i<length; i++)
    {
        tmp = createMusicalComposition(array_names[i], array_authors[i], array_years[i]);
        push(head , tmp);
    }
    return head;
}
```

Функция, добавляющая к списку новые элементы:

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

3. Удаление и вывод элементов списка

Функция удаления элементов списка, в поле «name» которых присутствует данная подстрока:

```
void removeEl(MusicalComposition* head, char* name_for_remove)
{
    MusicalComposition *tmp;
    tmp = head;
    while (tmp->next)
    {
```

Проверка на совпадение имени с подстрокой:

```
        if(strstr(tmp->name, name_for_remove) != NULL)
        {
```

Выполнение условий задачи для «головного» элемента списка:

```
            if(tmp->prev == NULL)
            {
                MusicalComposition *new_tmp = tmp->next;
                strcpy(tmp->name, new_tmp->name);
                strcpy(tmp->author, new_tmp->author);
                tmp->year = new_tmp->year;
                new_tmp->next->prev = tmp;
                tmp->next = new_tmp->next;
                free(new_tmp);
            }
```

Выполнение условий задачи для элемента, находящегося между первым и последним:

```
            else
            {
        tmp->next->prev = tmp->prev;
        tmp->prev->next = tmp->next;
        free(tmp);
            }
    }
```

```
            tmp = tmp->next;
    }
```

Выполнение условий задачи для «хвостового» элемента списка:

```
    if((strstr(tmp->name, name_for_remove) != NULL) && (tmp->next == NULL))
    {
            tmp->prev->next = NULL;
            free(tmp);
    }
}
```

Функция вывода элементов списка:

```
void print_names(MusicalComposition* head)
{
    MusicalComposition* tmp;
  tmp = head->next;
     printf("%s\n", head->name);
     printf("%s\n", tmp->name);
  while (tmp->next)
  {
       if ( tmp->next->year != -1 ) printf("%s\n", tmp->next->name);
       tmp = tmp->next;
  }
}
```

ЗАКЛЮЧЕНИЕ

В ходе проделанной работы был переработан двунаправленный линейный список музыкальных композиций, созданный в четвертой лабораторной работе, для выполнения особой функции. Программа принимает на вход число элементов списка и сами элементы, являющиеся объектами структуры данных, содержащие поля названий песен, имен авторов (названий групп), лет изданий и указателей на следующие и предыдущие элементы списка. В последнюю очередь принимается подстрока, совпадения с которой ищутся в названиях песен (в полях «name») каждого элемента списка, включая «головной» и «хвостовой». Элементы, в полях названий которых найдена введенная подстрока, удаляются из списка.

Для проверки выполнения кода программы выводится перечень названий композиций, принадлежащих оставшимся элементам списка.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Б. Керниган, Д. Ритчи; Язык программирования СИ, издание 3-е исправленное; перевод с английского под редакцией Вс. С. Штаркмана; Санкт-Петербург, 2003 г.
2. http://all-ht.ru/inf/prog/c/func/; Все о High-Tech, описание функций языка СИ.
3. http://www.c-cpp.ru/books/struktury; Программирование на языках С и С++, понятие структуры данных.

ПРИЛОЖЕНИЯ

КОД ПРОГРАММЫ:

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stddef.h>
#define N 81

typedef struct MusicalComposition
{
	char* name;
	char* author;
	int year;
	struct MusicalComposition* next;
	struct MusicalComposition* prev;
} MusicalComposition;

MusicalComposition* createMusicalComposition(char* name, char* author,int year)
{
	MusicalComposition* element_for_push =
(MusicalComposition*)malloc(sizeof(MusicalComposition));

	element_for_push->name = (char*)malloc(sizeof(char)*N);
	element_for_push->author = (char*)malloc(sizeof(char)*N);

	strcpy(element_for_push->name, name);
	strcpy(element_for_push->author, author);
	element_for_push->year = year;
	element_for_push->next = NULL;
	element_for_push->prev = NULL;

	return element_for_push;
}
```

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

    while (tmp->next)
    {
        tmp = tmp->next;
    }
    element->next = NULL;
    element->prev = tmp;
    tmp->next = element;
}

MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int*
array_years, int length)
{
    if(length == 0)
      return NULL;

    MusicalComposition* head  = (MusicalComposition*)malloc(sizeof(MusicalComposition));

    head->name = (char*)malloc(sizeof(char)*N);
    head->author = (char*)malloc(sizeof(char)*N);

    strcpy(head->name , array_names[0]);
    strcpy(head->author , array_authors[0]);
```

```
    head->year = array_years[0];

    MusicalComposition* tmp;

    for(int i=1; i<length; i++)
    {
        tmp = createMusicalComposition(array_names[i], array_authors[i], array_years[i]);
        push(head , tmp);
    }
    return head;
}


void removeEl(MusicalComposition* head, char* name_for_remove)
{
    MusicalComposition *tmp;
    tmp = head;

    while (tmp->next)
    {
        if(strstr(tmp->name, name_for_remove) != NULL)
        {
                    if(tmp->prev == NULL)
                    {
                        MusicalComposition *new_tmp = tmp->next;
                        strcpy(tmp->name, new_tmp->name);
                        strcpy(tmp->author, new_tmp->author);
                        tmp->year = new_tmp->year;
                        new_tmp->next->prev = tmp;
                        tmp->next = new_tmp->next;
                        free(new_tmp);
                    }
                    else
                    {
```

```
                tmp->next->prev = tmp->prev;
                tmp->prev->next = tmp->next;
                free(tmp);
                        }
         }
                tmp = tmp->next;
    }
        if((strstr(tmp->name, name_for_remove) != NULL) && (tmp->next == NULL))
        {
                tmp->prev->next = NULL;
                free(tmp);
        }
}

void print_names(MusicalComposition* head)
{
        MusicalComposition* tmp;
    tmp = head->next;
       printf("%s\n", head->name);
       printf("%s\n", tmp->name);
    while (tmp->next)
    {
         if ( tmp->next->year != -1 ) printf("%s\n", tmp->next->name);
         tmp = tmp->next;
    }
}

int main(){
        int length;
        printf("Size of your list:\n");
        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_remove[80];

fgets(name_for_remove, 80, stdin);
(*strstr(name_for_remove,"\n"))=0;

removeEl(head, name_for_remove);
printf("Total list of authors:\n");
print_names(head);

for (int i=0;i<length;i++)
{
free(names[i]);
```

```
        free(authors[i]);
        }

        free(names);
        free(authors);
        free(years);

        return 0;
}
```

MAKEFILE:

```
all: main.o
        gcc main.o
main.o: main.c
        gcc -c main.c
```

ПРИМЕРЫ ВЫПОЛНЕНИЯ: