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

КУРСОВАЯ РАБОТА

По дисциплине «Программирование»
Тема: структуры данных. Линейные списки.

Студентка гр. 7381 _____ Мартьянова Н.М.

Преподаватель _____ Берленко Т. А.

Санкт-Петербург

2017

ЗАДАНИЕ

Студенка Мартьянова Н. М.

Группа 7381

Тема работы: Структуры данных. Линейные списки

Необходимо составить список музыкальных композиций; написать функцию, разбивающую список на 3 части, меняющую их местами и соединяющую в один список.

Объем пояснительной записки:

Дата выдачи задания:

Дата сдачи:

Дата защиты:

Студентка _____ Мартьянова Н. М.

Преподаватель _____ Берленко Т. А.

АННОТАЦИЯ

В ходе выполнения курсовой работы была создана программа на языке СИ, которая реализует список композиций. Для работы со списком были созданы функции, позволяющие добавлять элемент в список; выводить названия композиций; разделять список на 3 части, менять эти части местами, и соединять в один список.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью работы является закрепить знания по созданию структур данных, по выделению и очищению динамической памяти, научиться работать с двунаправленными списками, создавать функции для работы с ними.

ФУНКЦИИ ПРОГРАММЫ

Функций для работы со списком

1.1 Создание элемента списка.

TElem *createMusicalComposition( )

Эта функция создает элемент списка. В ней используется функция динамического выделения памяти(malloc), функции вывода и ввода (printf, scanf). Возвращает указатель на созданный элемент.

1.2 Добавление элемента

void add(MusicalComposition **list, TElem *elem)

В функцию передается ссылка указателя на голову, указатель на элемент. Первый элемент списка содержит адрес на последний. Функция добавляет элемент в список. Ничего не возвращает.

1.3Вывод названий композиций на экран.

void print_names(MusicalComposition *list)

Функция получает указатель на первый элемент списка. Двигаясь в конец списка, печатает название каждой композиции. Ничего не возвращает.

1.4 Освобождение списка.

void freeAll(MusicalComposition *list)

Функция получает указатель на первый элемент списка. Двигаясь в конец списка, удаляет узлы списка и его элементы. Ничего не возвращает.

1.5 Соединение двух списков

TElem *concat(MusicalComposition *list1, MusicalComposition *list2)

Функция получает указатели на две части списка . Присоединяет list2 к list1. Возвращает указатель на начало списка list1.

1.6 Подсчет количества элементов

int count(MusicalComposition* head)

Функция получает указатель на первый элемент списка, и, перебирая все элементы, подсчитывает их количество. Возвращает количество элементов списка.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы создан двунаправленный список музыкальных композиций, функции для работы с ним. Были закреплены знания по выделению и очищению динамической памяти, получены знания по работе со списками. Создана программа, которая разделяет список на 3 части, меняет их местами и соединяет в один список.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Язык программирования СИ / Керниган Б., Риччи Д. Москва: Издательский дом «Вильямс» 2017. 000с.
2. Полный справочник по С++ / Герберт Шилдт. Москва: Издательский дом «Вильямс», 2008. 000 с.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД:

MAIN.C

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

042F

// для краткости записи - обьявим так тип

typedef struct LinkedList MusicalComposition;


// элемент списка

typedef struct Element {

   char name[80];
}TElem;


// узел двусвязного списка

struct LinkedList {

   TElem *elem;          // информационное поле

   MusicalComposition *prev;
   MusicalComposition *next ;        // указатели на следующий и предыдущий
элементы

} *head;  //


// для удобства - первый элемент списка будет хранить адрес на последний
элемент.

// передаем ссылку указателя на голову, чтоб функция стала более
универсальной.

void add(MusicalComposition **list, TElem *elem) {
```

```
    MusicalComposition *last, *tmp = (MusicalComposition*)
malloc(sizeof(MusicalComposition));
    if(!*list){

        *list = last = tmp;
        (*list)->prev = tmp;
    }
    last = (*list)->prev;
    tmp->elem = elem;
    last->next = tmp;
    tmp->next = NULL;
    tmp->prev = last;
    (*list)->prev = tmp;
}


// функция вывода списка на экран

void print_names(MusicalComposition *list) {

    while(list){
        printf("Name: %s\n", list->elem->name);
        list = list->next;
    }
}

// функция освобождения списка
// мы должны удалять не только узлы списка, но и все, для чего выделяли
память - также и элементы

void freeAll(MusicalComposition *list) {
    while(list){
        MusicalComposition *tmp = list;
        list = list->next;
        if(tmp->elem) free(tmp->elem);
        free(tmp);
    }

}


// создание нового элемента списка и возвращение указателя на него
TElem *createMusicalComposition(){
```

```
    TElem *tmp = (TElem*) malloc(sizeof(TElem));
    printf("Введите название композиции: ");
    scanf(" %[^\n]",tmp->name);
    return tmp;

}

// соединение двух списков

// приклеивает list2 к list1

// возвращает указатель на начало списка

TElem *concat(MusicalComposition *list1, MusicalComposition *list2) {

    MusicalComposition *tmp = list1;
    while(tmp->next) tmp = tmp->next;
    list2->prev = tmp;
    tmp->next = list2;

    return list1;

}

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

int main()

{

    head = NULL;
    // ввод элементов списка
    printf("Информация о музыкальных композициях\n");

    int choice;

    do {

        TElem *tmp = createMusicalComposition();
        add(&head, tmp);
```

```
        printf("Continue?  (1 - yes, 0 - no): ");
        scanf("%d", &choice);
    } while(choice);
int k=count(head);
    printf("Количество элементов: %d\n", k);
    printf("Список композиций до изменений:\n");
    print_names(head);

    // разделение списка на 3 равные части

    // нужно 3 указателя.В конечном итоге

    // head2  указывает на начало второй части

    // head3 - на начало 3 части

    // части могу быть и не равные - зависит от количества элементов.

    // При некратном 3 числе элементов максимальное число элементов будет в
третьей части

    // 3 числе элементов максмальное число

    // элементов будет в третьей части.

    printf("Разделение на 3 части...\n");
    MusicalComposition *head0, *head2, *head3;
    head0 = head2 = head3 = head;


    while(head0 && head0->next && head0->next->next){

        head0 = head0->next->next->next;

        head3 = head3->next->next;

        head2 = head2->next;

    }


    // необходимо "разорвать" части

    // что они не указывали друг на друга
```

```
  // и предыдущий элемент "головы" указывал на

  // последний элемент своего списка

  head0 = head->prev;
  head->prev = head2->prev;
  head2->prev = head3->prev;
  head3->prev = head0;



  head->prev->next = NULL;
  head2->prev->next = NULL;
  head3->prev->next = NULL;


// head2->prev = head2->prev->next = NULL;
// head3->prev = head3->prev->next = NULL;


  printf("1st part\n");

  print_names(head);

  printf("2nd part\n");

  print_names(head2);

  printf("3rd part\n");

  print_names(head3);

  printf("Соединение 3 частей...\n");

  head = concat(head2, head);
  head = concat(head3, head);

  printf("3-2-1 parts\n");
  print_names(head);

  // обязательно освобождаем память

  freeAll(head);

  return 0;
```

}