**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. 7303 _____ Юсковец А. В.

Преподаватель _____ Берленко Т. А.

Санкт-Петербург
2017

**ЗАДАНИЕ**

**НА КУРСОВУЮ РАБОТУ**

Студент Юсковец А. В.

Группа 7303

Тема работы: Линейные списки

Исходные данные (цель):
Разделить список на две части (индекс центрального элемента округлить в большую сторону), поменять их местами и соединить в один список

Содержание пояснительной записки:

1. Содержание
2. Введение
3. Функции для работы со структурой
4. Реализация swap_halfs
5. Функция main(тестирование)
6. Заключение
7. Список использованных источников

Дата выдачи задания: 22.11.2017

Дата сдачи реферата:

Дата защиты реферата:

Студент _____ Юсковец А. В.

Преподаватель _____ Берленко Т. А.

## АННОТАЦИЯ

Было описано несколько функций, ипользующихся в функции swap_halfs, реализующая требуемый в задании функционал. Также в точке входа написан код с закомментированными входными данными для тестирования swap_halfs.

**СОДЕРЖАНИЕ**

1. Функции для работы со структурой.....6
1.1. Функция count.....6
1.2. Функция createMusicalComposition.....6
1.3 Функция createMusicalCompositionList.....7
1.4 Функция print_names.....8
2. Реализация swap_halfs.....9
2.1. Код функции.....9
2.2. Описание реализации.....9
3. Функция main(тестирование).....11
3.1. Функция main.....11
3.2. Пример работы функции.....12

# ВВЕДЕНИЕ

Цель данной работы: написать функцию, разделяющую список на две части (индекс центрального элемента округлить в большую сторону), меняющую их местами и соединяющую в один список. Список содержит в себе элементы соответствующие структуре, имеющей два указателя: на предыдущий и на следующий элемент.

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

В функции swap_halfs была применена функция count, которая будет описана позже.

## 1. ФУНКЦИИ ДЛЯ РАБОТЫ СО СТРУКТУРОЙ

В этом разделе будут описаны некоторые функции, применяющиеся в требуемой и в ее тестировании.

### 1.1. Функция count

```
int count(MusicalComposition* head) {
   if (head == NULL) { return 0; }

   MusicalComposition* cur = head;
   int ret = 1;
   while (cur = cur->next) { ++ret; }

   return ret;
}
```

В случае, если список пустой функция возвращает 0. Указатель на композицию cur инициализируется head. ret — значение, возвращаемое функцией. Пока цикл не дойдет до конца списка ret будет увеличиваться на единицу.

### 1.2. Функция createMusicalComposition

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

   ret->name = name;
   ret->author = author;
   ret->year = year;
   ret->prev = NULL;
   ret->next = NULL;

   return ret;
}
```

Динамически выделяется память по композицию и соответствующим полям присваиваются переданные значения. Указатели на prev и next зануляются.

### 1.3 Функция createMusicalCompositionList

```
MusicalComposition* createMusicalCompositionList(char** array_names,
                                                 char** array_authors,
                                                 int* array_years, int n) {
   MusicalComposition* head =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
   head = createMusicalComposition(array_names[0],
                                   array_authors[0],
                                   array_years[0]);

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

   MusicalComposition* cur =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
   for (int i = 1; i != n; ++i) {
      cur = createMusicalComposition(array_names[i],
                                     array_authors[i],
                                     array_years[i]);
      cur->prev = prev;
      cur->prev->next = cur;
      cur->next = NULL;
      prev = cur;
   }

   return head;
}
```

head – голова списка. Соответствующая композиция инициализируется первыми элементами массивов.

prev – указатель на предыдущий элемент списка, инициализируется head

cur – текущий, создающийся на данной итерации, элемент списка

В цикле cur присваивается указатель на соответствующую музыкальную композицию.

Далее указатель на предыдущий элемент списка становится prev, указатель на следующий элемент prev становится cur, указатель на следующий за cur элемент зануляется. prev сдвигается на cur и цикл повторяется.

Функция возвращает голову списка.

### 1.4 Функция print_names

```
void print_names(MusicalComposition* head) {
   if (head == NULL) { return ; }

   MusicalComposition* cur = head;
   do {
      printf("%s\n", cur->name);
   } while (cur = cur->next);
}
```

В случае, если список пустой функция завешрается. Указатель на композицию cur инициализируется head.

Пока цикл не дойдет до конца списка, будет выводиться поле name каждого элемента.

## 2. РЕАЛИЗАЦИЯ SWAP_HALFS

В этом разделе будет описана реализация функции, решающая постановленную задачу.

### 2.1. Код функции

```
void swap_halfs(MusicalComposition** head) {
    if (head == NULL || *head == NULL) { return; }

    int list_len = count(*head);
    int half_index = list_len / 2;
    MusicalComposition* cur = *head;
    MusicalComposition* new_head = NULL;

    for (int i = 0; i != half_index; ++i) { cur = cur->next; }
    cur->prev->next = NULL;
    cur->prev = NULL;
    new_head = cur;

    for (int i = half_index; i != list_len - 1; ++i) { cur = cur->next; }
    cur->next = *head;
    (*head)->prev = cur;

    *head = new_head;
}
```

### 2.2. Описание реализации

Функция принимает указатель на указатель на голову списка (для того, чтобы в дальнейшем изменить расположение головы).

В случае если был переда нулевой указатель или указатель ссылающийся на NULL функция завершает свою работу.

Иначе инициализируются переменные:

- list_len — длина списка;
- half_index — индекс срединного элемента;
- cur — текущий элемент списка;
- new_head — элемент, который станет первым после перестановки половин.

В цикле программа доходит до середины списка и присваивает cur срединный элемент. Элемент перед текущим становится хвостом списка, посредством удаления (prev→next = NULL) у него поля next. А cur стоановится головой списка, посредством удаления у него поля prev (cur→prev = NULL). На данном этапе образовалось две, никак не связанных части списка.

Далее в коде связывается последний элемент правой части с первым элементом левой: в цикле for cur доходит до последнего элемента. Его полю next присваивается голова (то есть первый элемент левой части).

Последний шаг: идущий перед прошлой головой элемент становится последний элемент правой части, и указатель на голову сменяется новой головой (new_head), а именно первым элементом правой части изначального списка.

## 3. ФУНКЦИЯ MAIN(ТЕСТИРОВАНИЕ)

В этом разделе будет описана функция main, в которой будет приведено тестирование написанной функции.

### 3.1. Функция main

```
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);

    print_names(head);
    printf("\n");
    swap_halfs(&head);
    print_names(head);

    return 0;
}
```

В функции main происходи считывание тестировочных данных, их вывод на консоль, вызов функции swap_halfs и повторный вывод списка на консоль для убеждения в том, что функция корректно работает.

### 3.2. Пример работы функции

Входные данные:

```
6
Fields of Gold
Sting
1993
In the Army Now
Status Quo
1986
Mixed Emotions
The Rolling Stones
1989
Billie Jean
Michael Jackson
1983
Wicked Game
Chris Isaak
1989
Points of Authority
Linkin Park
2000.
```

Выходные данные:

```
Fields of Gold
In the Army Now
Mixed Emotions
Billie Jean
Wicked Game
Points of Authority

Billie Jean
Wicked Game
Points of Authority
Fields of Gold
In the Army Now
Mixed Emotions
```

Входные данные:

```
5
In the Army Now
Status Quo
1986
Mixed Emotions
The Rolling Stones
1989
Billie Jean
Michael Jackson
1983
Wicked Game
Chris Isaak
1989
Points of Authority
Linkin Park
2000.
```

Выходные данные:

```
In the Army Now
Mixed Emotions
Billie Jean
Wicked Game
Points of Authority

Billie Jean
Wicked Game
Points of Authority
In the Army Now
Mixed Emotions
```

# ЗАКЛЮЧЕНИЕ

Была реализована функция, меняющая местами две половины списка, правильно работающая на предоставленных тестах. При компиляции и работе всей программы не возникает ошибок и предупреждений, связанных с логикой программы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Б. Керниган, Д. Ритчи «Язык программирования С», третье издание. Издательство: «Невский Диалект», 2001г.