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

Студент гр. 7381 _____ Смирнов М.А.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2017

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Смирнов М. А.

Группа 7381

Тема работы Структуры данных. Линейные списки

Исходные данные:

Необходимо составить список музыкальных композиций (название, автор, год), Написать функцию, сортирующую первую часть списка по возрастанию, вторую по убыванию.

Содержание пояснительной записки:

Содержание, Введение, Описание тела функции main.c, Описание дополнительных функций работы со списками, Функция сортировки, Примеры работы программы, Заключение, Список использованных источников

Предполагаемый объем пояснительной записки:

Не менее 24 страницы.

Дата выдачи задания: 23.11.2017

Дата сдачи реферата:

Дата защиты реферата:

Студент _____ Смирнов М.А.

Преподаватель _____ Берленко Т.А.

АННОТАЦИЯ

В ходе выполнения курсовой работы была создана программа на языке СИ, которая реализует список: композиций, автор композиции, год создания и на основе данных пользователя и предоставляет пользователю возможность отсортировать введенные данные по году, поделив его на две части. Одна часть сортируется по возрастанию, другая по убыванию. Также программа содержит удобный пользователю интерфейс с доступом к необходимым инструментам.

SUMMARY

During the course work, a program was developed in the SI language, which implements the list based on user data and provides the user with the ability to sort the entered data by year, dividing it into two parts. One part is sorted in ascending order, the other in descending order. The program also contains a user-friendly interface with access to the necessary tools.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью работы, является закрепление, необходимых в дальнейшем, знаний создания структур данных в языке СИ. Для достижения поставленной цели требуется изучить основы структурирования данных в справочных источниках, создать условия для работы со списком: программу-оболочку, Makefile и ряд необходимых функций, собрать программу в доступном компиляторе и провести несколько тестов над ней.

1. ОПИСАНИЕ ТЕЛА ФУНКЦИИ MAIN.C

1.1. В начале необходимо ввести элементы будущего списка. При помощи функции *scanf* вводится их количество.

1.2. Динамически выделяется память под массивы для хранения названий композиций, авторов и дат создания. Выделяется с помощью функции выделения malloc.

1.3. Заполняются данные списка. Создается двунаправленный список по данным из массива.

1.4. Пользователю предлагается набор функций для работы со списком:

1.4.1. №1 - Отсортировать половину списка по возрастанию года, а вторую по убыванию года.

1.4.2. №2 - Для проверки на количество введенных элементов есть возможность вывести количество элементов списка.

1.4.3. №3 - Для тестирования работы функции сортировки присутствует добавление элемента в конец списка.

1.4.4. №4 - Возможность безопасно выйти из программы.

1.4.5. После выполнения одной из функций предоставляется возможность вернуться к выбору функций.

1.5. В конце работы динамическая память, выделенная для работы освобождается с помощью функции *free*.

2. ОПИСАНИЕ ДОПОЛНИТЕЛЬНЫХ ФУНКЦИЙ ДЛЯ РАБОТЫ СО СПИСКОМ

2.1. Создана структура Musical Composition хранящая в себе 2 переменные-указателя *char**, *int* и 2 указателя на структуру, содержащую адреса следующего и предыдущего элементов**.**

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

2.2. Функция createMusicalComposition .Функция создает элемент списка, на вход она получает 3 параметра с именем, автором и годом создания. Функция возвращает указатель на первый элемент списка.

```
MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n)
    {
        int g;
        if (n==0) return NULL;
        MusicalComposition *head=createMusicalComposition(array_names[0],array_authors[0], array_years[0]);
        MusicalComposition *tp=(MusicalComposition*)malloc(sizeof(MusicalComposition));
                                        for(g=1;g<n;g++)
                    {
                        tp=createMusicalComposition(array_names[g],array_authors[g],array_years[g]);
                        push(head,tp);
                    }
                       return head;
    }
```

2.3. Функция push, добавляющая элемент в конец списка. На вход подается 2 параметра: указатель на элемент добавления и на первый элемент.

Функция возвращает head.

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

        MusicalComposition *temp = head->next;

        while (temp->next)
        {
                temp = temp->next;
        }

        element->next = NULL;
        element->prev = temp;
        temp->next = element;
        return head;
}
```

2.4. Функция count, считывающая количество элементов списка. На вход подается указатель на первый элемент списка. При помощи цикла *for* идет подсчет количества элементов списка.

```
int count(MusicalComposition* head)
    {
        int n=1;
        MusicalComposition* tp=(MusicalComposition*)malloc(sizeof(MusicalComposition));
        tp = head->next;
        while (tp)
        {
                tp = tp->next;
                n++;

        }
        return n;
    }
```

2.5 Функция print_names выводит названия композиций, необходимых для удобства функции сортировки. На вход программе подается указатель на первый элемент списка. При помощи printf выводятся названия композиций.

```
void print_names(MusicalComposition* head)
        {
       if(head== NULL) return;
       MusicalComposition *tp=head;
        do {
            printf("%d - %s - %s\n",tp->year,tp->name,tp->author);
            tp = tp->next;
        } while (tp !=NULL);
          }
```

3. ФУНКЦИЯ СОРТИРОВКИ

3.1. Ключевая курсового проекта функция halfsort. Функция делит элементы списка на две части, где одна часть сортируется по возрастанию значения year, другая по убыванию этого же значения year. Принцип работы функции заключается в сравнении двух элементов и замены их местами в соответствии их значения и цели (возрастание/убывание), указатель перемещается по списку с помощью нескольких циклов *for*.

```
void halfsort(MusicalComposition* head)
{
    if(head==NULL)
    {
        printf("spisok pust\n");
        return;
    }
    MusicalComposition* tp;
    MusicalComposition* a;
    MusicalComposition* prom;
    int n=1,N,B,i,j,nonstop;
    int year;
    char name[81];
    char author[81];
    tp=head->next;
    n = count(head);
    if (n<=2) return;
    N=n/2;
    if (n%2==0) B=N-1;
        else B=N;

    for(i=N-1;i>=0;i--)
    {
        nonstop=1;
        tp=head;
        a=tp->next;
        for(j=0;j<N;j++)
        {
            if (tp->year>a->year)
            {
                year=tp->year;
                tp->year = a->year;
                a->year = year;

                strcpy(name,tp->name);
                strcpy(tp->name,a->name);
                strcpy(a->name,name);

                strcpy(author,tp->author);
                strcpy(tp->author,a->author);
                strcpy(a->author,author);
                nonstop=0;
            }
```

```
            tp=tp->next;
            a=a->next;
        }
        if(nonstop==1) break;
    }

    prom=tp;

  for(i=N-1;i>=0;i--)
    {
        nonstop=1;
        tp=prom;
        a=tp->next;
        for(j=0;j<B;j++)
        {
            if (tp->year<a->year)
            {
                year=tp->year;
                tp->year = a->year;
                a->year = year;

                strcpy(name,tp->name);
                strcpy(tp->name,a->name);
                strcpy(a->name,name);

                strcpy(author,tp->author);
                strcpy(tp->author,a->author);
                strcpy(a->author,author);
                nonstop=0;
            }
            tp=tp->next;
            a=a->next;
        }
        if(nonstop==1) break;
    }
}
```

4. ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ:

4.1. Первый тест

4.1. 1. Ввод данных:

```
Number of songs=4
1
Song name: Africa
Author: Toto
Year: 1980
2
Song name: Werewolves of London
Author: Warren Zevon
Year: 1990
3
Song name: 505
Author: Arctic Monkeys
Year: 2005
4
Song name: Pogo
Author: Digitalism
Year: 2010
```

4.1.2. Результат работы функции halfsort:

```
Choose function
1-Amount of elements
2-Add element to end
3-Sort and show list
4-Exit
3
1980 - Africa - Toto
1990 - Werewolves of London - Warren Zevon
2010 - Pogo - Digitalism
2005 - 505 - Arctic Monkeys
```

4.2. Второй тест

4.2.1. Ввод данных:

```
Number of songs=6
1
Song name: Daisuki with TeddyLoid
Author: DAOKO
Year: 2030
2
Song name: Step by Step
Author: Evilwave
Year: 2001
3
Song name: Mask off
Author: Future
Year: 1990
4
Song name: Feel Good inc
Author: Gorillaz
Year: 1999
5
Song name: Fantastic Lets Go
Author: IOSYS
Year: 2003
6
Song name: I WARE HOUSE
Author: JOYRYDE
Year: 2015
```

4.2.2.Дополняем список еще одной композицией:

```
Choose function
1-Amount of elements
2-Add element to end
3-Sort and show list
4-Exit
2
Song name: Pumped Up
Author: Klingande
Year: 2014
```

4.2.3. Результат работы функции halfsort:

```
1990 - Mask off - Future
1999 - Feel Good inc - Gorillaz
2001 - Step by Step - Evilwave
2030 - Daisuki with TeddyLoid - DAOKO
2015 - I WARE HOUSE - JOYRYDE
2014 - Pumped Up - Klingande
2003 - Fantastic Lets Go - IOSYS
```

4.3. Третий тест

4.3.1. Ввод данных:

```
Number of songs=3
1
Song name: After Dark
Author: Mr.Kitty
Year: 2003
2
Song name: Nicotine
Author: Panic!
Year: 2013
3
Song name: Let me Alone
Author: moow
Year: 2005
```

4.3.2. Результат работы функции halfsort:

```
Choose function
1-Amount of elements
2-Add element to end
3-Sort and show list
4-Exit
3
2003 - After Dark - Mr.Kitty
2013 - Nicotine - Panic!
2005 - Let me Alone - moow
```

ЗАКЛЮЧЕНИЕ

Во время выполнения курсовой работы были освоены навыки по работе со структурами данных, а также навыки создания двунаправленного линейного списка в языке СИ. На основе полученных навыков была составлена программа, дающая истинный результат, согласно условию, представленная в доступном пользователю.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Язык программирования / Керниган Б., Риччи Д. Москва: Издательский дом "Вильямс" 2017. 000 с.

2. Полный справочник по C++ / Герберт Шилдт. Москва: Издательский дом "Вильямс", 2008. 000 с.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД:

MAIN.C:

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

int main()
 {
  printf("Number of songs=");

    int length; int i;
    scanf("%d", &length);

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

    for ( i=0;i<length;i++)
    {
       printf("%d\n",i+1);
       char name[81];
       char author[81];
       printf("Song name: ");
       getchar();
       fgets(name, 81, stdin);
       printf("Author: ");
       fgets(author, 81, stdin);
       printf("Year: ");

       fscanf(stdin, "%d", &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);
        }
        printf("\n");
            MusicalComposition* head = createMusicalCompositionList(names,
authors, years, length);
        int da=1;
        int choice = 0;
        int year_for_remove=0;
        char *name_for_push=(char*)malloc(sizeof(char*) * 81);
        char *author_for_push=(char*)malloc(sizeof(char*) * 81);
        int year_for_push=0;
        char *name_for_remove = (char*)malloc(sizeof(char)*81);
        while (da)
        {
            printf("Choose function\n");
            printf("1-Amount of elements\n");
            printf("2-Add element to end\n");
            printf("3-Sort and show list\n");
            printf("4-Exit\n");
            scanf("%d",&choice);
            switch(choice)
            {
            case 1:
                printf("Count = %d\n",count(head));
                break;
            case 2:

                                        printf("Song name: ");MusicalComposition*
remove_all_year(MusicalComposition *list, int n);
```

```
                getchar();
                fgets(name_for_push, 81, stdin);

                printf("Author: ");
                fgets(author_for_push, 81, stdin);

                printf("Year: ");
                fscanf(stdin, "%d", &year_for_push);

                (*strstr(name_for_push,"\n"))=0;
                (*strstr(author_for_push,"\n"))=0;
                                                                    head =
push(head,createMusicalComposition(name_for_push,author_for_push,year_for_pus
h));
                break;
            case 3:
                halfsort(head);
                  print_names(head);

                break;
                case 4: return 0;
            default:
                printf("WRONG! Please try again\n");
            }
        }
        for ( i=0;i<length;i++){
            free(names[i]);
            free(authors[i]);
        }
        free(names);
        free(authors);
        free(years);
        return 0;
    }
```

KURS.C:

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stddef.h>
#include "function.h"
    void halfsort(MusicalComposition* head)
    {
       if(head==NULL)
       {
          printf("spisok pust\n");
          return;
       }
       MusicalComposition* tp;
       MusicalComposition* a;
       MusicalComposition* prom;
       int n=1,N,B,i,j,nonstop;
       int year;
       char name[81];
       char author[81];
       tp=head->next;
       n = count(head);
       if (n<=2) return;
       N=n/2;
       if (n%2==0) B=N-1;
          else B=N;
       for(i=N-1;i>=0;i--)
       {
          nonstop=1;
          tp=head;
          a=tp->next;
          for(j=0;j<N;j++)
          {
             if (tp->year>a->year)
```

```
        {
            year=tp->year;
            tp->year = a->year;
            a->year = year;

            strcpy(name,tp->name);
            strcpy(tp->name,a->name);
            strcpy(a->name,name);

            strcpy(author,tp->author);
            strcpy(tp->author,a->author);
            strcpy(a->author,author);
            nonstop=0;
        }
        tp=tp->next;
        a=a->next;
    }
    if(nonstop==1) break;
  }
  prom=tp;
for(i=N-1;i>=0;i--)
  {
    nonstop=1;
    tp=prom;
    a=tp->next;
    for(j=0;j<B;j++)
    {
        if (tp->year<a->year)
        {
            year=tp->year;
            tp->year = a->year;
            a->year = year;

            strcpy(name,tp->name);
```

```
                strcpy(tp->name,a->name);
                strcpy(a->name,name);

                strcpy(author,tp->author);
                strcpy(tp->author,a->author);
                strcpy(a->author,author);
                nonstop=0;
            }
            tp=tp->next;
            a=a->next;
        }
        if(nonstop==1) break;
    }
}
```

OTHERS.C:

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

//создание структуры


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

```
    addmusic->next=NULL;
    addmusic->prev=NULL;
    return addmusic;
    }

                    MusicalComposition*  push(MusicalComposition*  head,
MusicalComposition* element)
    {
        if (!head)
        {
           head=element;
           return head;
        }
        if ( head->next == NULL )
        {
            element->next = NULL;
            element->prev = head;
            head->next = element;
            return head;
        }

        MusicalComposition *temp = head->next;

        while (temp->next)
        {
            temp = temp->next;
        }

        element->next = NULL;
        element->prev = temp;
        temp->next = element;
        return head;
    }
```

```
MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n)
{
  int g;
  if (n==0) return NULL;
                                                    MusicalComposition *head=createMusicalComposition(array_names[0],array_authors[0], array_years[0]);
                                                    MusicalComposition *tp=(MusicalComposition*)malloc(sizeof(MusicalComposition));
                                        for(g=1;g<n;g++)
                    {
                      tp=createMusicalComposition(array_names[g],array_authors[g],array_years[g]);
                     push(head,tp);
                    }
                     return head;
}

int count(MusicalComposition* head)
{
  int n=1;
                                                    MusicalComposition* tp=(MusicalComposition*)malloc(sizeof(MusicalComposition));
  tp = head->next;
  while (tp)
  {
        tp = tp->next;
        n++;

  }
  return n;
}
  void print_names(MusicalComposition* head)
      {
```

```
        if(head== NULL) return;
        MusicalComposition *tp=head;
         do {
            printf("%d - %s - %s\n",tp->year,tp->name,tp->author);
            tp = tp->next;
         } while (tp !=NULL);
          }
```

FUNCTION.H:

```
#pragma once

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* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n);

MusicalComposition* push(MusicalComposition* head, MusicalComposition* element);

int count(MusicalComposition* head);

void halfsort(MusicalComposition* head);