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 _____ Вологдин М.Д.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2017

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Вологдин М.Д.

Группа 7381

Тема работы: Структуры данных, линейные списки

Исходные данные:
Создать двунаправленный список музыкальных композиций MusicalComposition и **api** (***a****pplication* ***p****rogramming* ***i****nterface* - *в данном случае набор функций*) для работы со списком.
Создать функцию для удаления всех элементов списка, год которых меньше n (n подается на вход функции).
Создать удобный интерфейс для работы программы.

Предполагаемый объем пояснительной записки:
Не менее 10 страниц.

Дата выдачи задания: 23.11.2017

Дата сдачи реферата:

Дата защиты реферата:

Студент _____ Вологдин М.Д.

Преподаватель _____ Берленко Т.А.

АННОТАЦИЯ

В результате выполнения курсовой работы была создана программа на языке СИ, которая позволяет работать со списком и с функциями для него, такими как: удаление элемента, добавление элемента в конец списка, подсчет количества элементов, удаление всех элементов, год которых меньше данного, а также вывод самого списка. Создан удобный интерфейс для пользования программой.

SUMMARY

As a result of the course work, a C program was created that allows you to work with the list and with functions for it, such as: deleting an item, adding an item to the end of the list, counting the number of items, deleting all elements in which the year is less than this, and also list output. A convenient interface for using the program was created.

СОДЕРЖАНИЕ

Оглавление

Введение

В данной работе необходимо научиться работать с двунаправленными списками, уметь создавать функции для работы с ними. Для этого была создана программа на языке СИ, наглядно показывающая данные умения.

Функции программы

1. ФУНКЦИЯ MAIN.C

1.1. Вводиться количество элементов списка.

1.2. Динамически выделяется память под массивы для хранения названий композиций, авторов и дат создания.

1.3. Заполняются данные массива

1.4. Создается двунаправленный список по данным из массива.

1.5. Пользователю предлагают многократный выбор дальнейших действий

1.5.1. №1 – Вывести количество элементов списка.

1.5.2. №2 – Добавить элемент в конец списка.

1.5.3. №3 – Удалить элемент с данным названием.

1.5.4. №4 – Удалить все элементы, в которых год меньше заданного.

1.5.5. №5 – Вывести названия композиций.

1.6. Освобождается вся выделенная динамически память.

2. ФУНКЦИИ ДЛЯ РАБОТЫ СО СПИСКОМ

2.1. MusicalComposition* createMusicalComposition(char* name, char* author,int year)

Данной функции передают три параметра: название (char* name), имя автора (char* author) и год создания композиции (int year); которые записываются в элемент после выделения памяти для структуры MusicalComposition. Функция возвращает указатель на созданный элемент списка.

2.2. MusicalComposition* push(MusicalComposition* head, MusicalComposition* element)

Данной функции передается указатель на начало линейного списка (MusicalComposition* head) и элемент, который необходимо добавить (MusicalComposition* element). Функция перемещается по списку, пока не доходит до последнего элемента, а затем соединяет последний и новый элементы списка посредством указателей. Функция возвращает указатель на первый элемент списка.

2.3. MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n)

Функция получает количество существующих композиций и указатели на три массива: массив названий (char** array_names), массив авторов (char** array_authors) и массив годов созданий (int* array_years); и формирует линейный двунаправленный список. Функция возвращает указатель на первый элемент списка.

2.4. MusicalComposition* removeEl(MusicalComposition* list, char* name_for_remove)

Функция получает указатель на начало списка (MusicalComposition* head) и название композиции (char* name_for_remove), которую нужно удалить. Перемещаясь по списку, она сверяет название композиции с переданной строкой. При совпадении значений функция связывает

предыдущий и следующий элементы посредством указателей, а затем очищает удаленный элемент. Функция возвращает указатель на первый элемент списка.

2.5. int count(MusicalComposition* head)

Функция получает указатель на первый элемент списка (MusicalComposition* head), и, пробегая весь список, подсчитывает количество элементов в нем. Функция возвращает количество элементов списка.

2.6. void print_names(MusicalComposition* head)

Данной функции передается указатель на первый элемент списка (MusicalComposition* head). Перебирая все элементы, функция печатает названия каждой композиции.

2.7. MusicalComposition* remove_all_year(MusicalComposition *list, int n)

Функция получает указатель на первый элемент списка (MusicalComposition *list) и год(int n), по которому удаляются элементы. Пробегая весь список, функция проверяет, удовлетворяет ли год композиции условию, и в противном случае удаляет этот элемент. Функция возвращает указатель на первый элемент списка.

Заключение

В ходе выполнения данной курсовой работе создан двунаправленный линейный список, функции для работы с ним и удобный интерфейс для пользования. Были получены необходимые навыки для работы со списками и функциями для них. Также были закреплены знания по выделению и очищению динамической памяти.

Список использованных источников

- Б.Керниган Д.Риччи “язык прогаммирование Си”
- Демидович Е. “Основы алгоритмизации и программирования. Язык Си “
- Подбельский В.С. Фомин С.С. “Курс программирования на Си: учебник “
- Robert Sedgewick, Kevin Wayne “Algorithms in C”

Приложение А
Исходный код программы

1. Main.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stddef.h>
#include "func.h"
#include "year.h"
int main(){
   printf("Number of songs=");
   int length;
   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 (int i=0;i<length;i++)
   {
      printf("%d\n",i+1);
      char name[80];
      char author[80];
      printf("Song name: ");
      getchar();
      fgets(name, 80, stdin);
      printf("Author: ");
      fgets(author, 80, 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("What do you want?\n");
   printf("1-Amount of elements\n");
   printf("2-Add an item to the end of the list\n");
   printf("3-Delete the item with the name you typed\n");
   printf("4-Delete all items in the list, in which the year is less than the one you type\n");
   printf("5-Show list\n");

   scanf("%d",&choice);
   switch(choice)
   {
   case 1:
      printf("Count = %d\n",count(head));
      break;
   case 2:

      printf("Song name: ");
      getchar();
      fgets(name_for_push, 80, stdin);

      printf("Author: ");
      fgets(author_for_push, 80, 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_push));
      break;
   case 3:
```

```
            printf("Enter the name\n");
            getchar();
            fgets(name_for_remove,80,stdin);
            (*strstr(name_for_remove,"\n"))=0;
            head = removeEl(head,name_for_remove);
            break;
        case 4:
            printf("Enter the year\n");
            scanf("%d",&year_for_remove);
            head=remove_all_year(head,year_for_remove);
            break;
        case 5:
            if (head)
                print_names(head);
            else
                printf("The list is empty!\n");
            break;
        default:
            printf("WRONG! Please try again\n");
        }
        printf("Contnue? Yes-1 No-0\n");
        scanf("%d",&da);
    }
    for (int i=0;i<length;i++){
        free(names[i]);
        free(authors[i]);
    }
    free(names);
    free(authors);
    free(years);

    clear_list(head);
    return 0;

}
```

2. Func.h

```
#pragma once
#include <stdlib.h>
#include <stdio.h>
```

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

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

// Создание структуры MusicalComposition

MusicalComposition* createMusicalComposition(char* name, char* author,int year)
{
   MusicalComposition* create = (MusicalComposition*)malloc(sizeof(MusicalComposition));
   create->name = (char*)malloc(sizeof(char)*81);
   create->author = (char*)malloc(sizeof(char)*81);
   strcpy(create->name,name);
   strcpy(create->author,author);
   create->year=year;
   create->next=NULL;
   create->prev=NULL;
   return create;
}


// Функции для работы со списком MusicalComposition
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)
{
    if (n==0)
        return NULL;
    MusicalComposition *head = createMusicalComposition(array_names[0],array_authors[0],array_years[0]);
    MusicalComposition *temp;
    for (int i=1;i<n;i++ )
    {
        temp=createMusicalComposition(array_names[i],array_authors[i],array_years[i]);
        push(head,temp);
    }
    return head;
}



MusicalComposition* removeEl(MusicalComposition* list, char* name_for_remove)
{
    MusicalComposition *head = list;
    if (head==NULL)
        return head;
    MusicalComposition *temp = head;

    if (strcmp(name_for_remove, temp->name) == 0)
    {
        if (temp->next)
```

```
            temp->next->prev=NULL;
        free(temp->name);
        free(temp->author);
        list=list->next;
        free(temp);
        return list;
    }
    while(temp && (strcmp(temp->name,name_for_remove) != 0))
    {
        temp=temp->next;
    }
    if (!temp)
        return list;
    if (temp->next)
    {
        temp->next->prev = temp->prev;

    }
    temp->prev->next = temp->next;
    free(temp->author);
    free(temp->name);
    free(temp);

    return list;
}

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

    }
    return i;
}

void print_names(MusicalComposition* head)
```

```
{
   MusicalComposition* current = head;
   while(current)
   {
      printf("%s\n",current->name);
      current = current->next;
   }
   return;
}
void clear_list(MusicalComposition *head)
{
   if (!head)
   {
      return;
   }
   while(head->next)
   {
      head=head->next;
      free(head->prev->author);
      free(head->prev->name);
      free(head->prev);

   }
   free(head->author);
   free(head->name);
   free(head);
   return;
}
```

3. Year.h

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

MusicalComposition* remove_all_year(MusicalComposition *list, int n)
{
   MusicalComposition *temp;
   MusicalComposition *head;
```

```
    head=list;
    while(head)
    {
        if (head->year < n)
        {
            temp=head;
            if (temp->prev)
                temp->prev->next=temp->next;

            else
            {
                if (temp->next)
                    temp->next->prev=NULL;
                free(temp->name);
                free(temp->author);
                list=list->next;
                head=head->next;
                free(temp);
                temp=NULL;
                continue;
            }
            if (temp->next)
                temp->next->prev=temp->prev;
            head=temp->next;
            free(temp->name);
            free(temp->author);
            free(temp);
            temp=NULL;
        }
        else
            head=head->next;
    }
    return list;
}
```