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 _____ Алясова А.Н.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2017

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студентка Алясова А.Н.

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, которая удаляет каждый третий элемент списка.

Содержание пояснительной записки:
«Содержание», «Введение», «Заключение», «Список использованных источников»

Предполагаемый объем пояснительной записки:
Не менее 10 страниц.

Дата выдачи задания: 23.11.2017

Дата сдачи реферата:

Дата защиты реферата:

Студентка _____ Алясова А.Н.

Преподаватель _____ Берленко Т.А.

АННОТАЦИЯ

В результате выполнения курсовой работы был создан двунаправленный линейный список. Созданы функции для работы с данным списком, а именно: добавление в конец нового элемента списка, удаление элемента списка с заданной композицией, подсчёт количества элементов списка, вывод названий элементов списка, функция удаления каждого третьего элемента списка. Создан удобный интерфейс для работы с программой.

SUMMARY

As a result of performing the course work was created by a bidirectional linear list. Created functions for working with data list, namely: add to end of new list item, delete list item, counting the number of elements in the list, output the names of the list items, delete function of the even list elements. Created user-friendly interface for working with the program.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
1.ОПИСАНИЕ ТЕЛА ФУНКЦИИ MAIN	6
2.ОПИСАНИЕ ФУНКЦИЙ ДЛЯ РАБОТЫ СО СПИ СКОМ	7
2.1.структура MusicalComposition	**7**
2.2.функция createMusicalComposition	7
2.3. функция push	7
2.4. функция removeEL	7
2.5. функция count	7
2.6. функция print_names	7
2.7. функция RemoveEvery	7
ЗАКЛЮЧЕНИЕ	9
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	10
ПРИЛОЖЕНИЕ А. Код программы	11

ВВЕДЕНИЕ

Целью данной курсовой работы является закрепление знаний по созданию структур данных в языке Си. Научиться работать с двунаправленным линейным списком и создавать различные функции для неё. Консолидировать знания в работе с динамической памятью.

1. ОПИСАНИЕ ТЕЛА ФУНКЦИИ MAIN.C

1.1. При помощи функции ввода общего значения scanf() вводится количество композиций элементов, составляющих список.

1.2. Динамически выделяется память под названия композиций, имена авторов и год создания композиции.

1.3. Заполняются данные списка.

1.4. При помощи функции **createMusicalComposition** создаётся двунаправленный список по заполненным данным.

1.5. Далее пользователю предлагается набор функций для работы со списком:

1 – добавление новой композиции в конец списка;

2 – удаление элемента списка с заданной композицией;

3 – нахождение количества элементов списка;

4 – вывод названий композиций;

5 – удаление каждого третьего элемента списка;

6 – окончание работы со списком;

1.6. Освобождается динамическая память, выделенная для работы со списком.

2. ОПИСАНИЕ ФУНКЦИЙ ДЛЯ РАБОТЫ СО СПИСКОМ

2.1. Была создана структура **MusicalComposition**, хранящая в себе два переменных-указателя типа char*, переменную типа int и два указателя на структуру, которые содержат адресса следующего и предыдущего элементов.

2.2. Создана функция для создания элемента списка - **createMusicalComposition**. Данная функция получает на вход три параметра: имя(name), автор(author), и год издания композиции(year). В ней выделяется память при помощи функции malloc() под новый элемент списка и используется функция strncpy() для копирования данных о композиции в новый элемент списка. Функция возвращает указатель на первый элемент списка.

2.3. Функция **push** добавляет элемент в конец списка музыкальных композиции. На вход функции подается два параметра: указатель на элемент, который нужно добавить и указатель на первый элемент. Функция ничего не возвращает.

2.4. Функция **removeEl** удаляет элемент списка, у которого имя композиции совпадает с именем композиции для удаления. На вход программе подаются: указатель на первый элемент и указатель на имя элемента для удаления. При помощи функции strcmp() проверяется список на совпадение имя композиции с именем композиции для удаления. Функция ничего не возвращает.

2.5. Функция **count** возвращает количество элементов списка. На вход подаётся указатель на первый элемент списка. При помощи цикла for подсчитывается количество элементов списка.

2.6. Функция **print_names** выводит названия композиций. На вход программе подаётся указатель на первый элемент списка. При помощи цикла for и функции printf() выводятся названия композиций. Функция ничего не возвращает.

2.7. Создана функция **RemoveEvery** которая удаляет каждый третий элемент списка. На вход функции подаётся указатель на первый элемент списка

и период, с каким будет удаляться каждый последующий элемент. При помощи циклических операторов происходит удаление каждого третьего элемента списка, с учётом того что в списке может быть один и более элементов. Функция возвращает указатель на первый элемент.

ЗАКЛЮЧЕНИЕ

В ходе данной курсовой работы были закреплены знания по работе со структурами данных в языке Си. Получены знания по созданию двунаправленного линейного списка, знания по работе и созданию функций для работы со списком. Были закреплены знания по выделению и очищению динамической памяти. Реализована функция, удаляющая каждый третий элемент списка, создан удобный интерфейс для работы со списком.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Б.Керниган, Д.Ритчи “Язык программирования Си”
2. В.В.Подбельский, С.С.Фомин “Программирование на языке Си”.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

MAIN.C

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

int main(){

int length;
printf("Введите количество композиций:");
scanf("%d", &length);
char w = getchar();

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

MusicalComposition* head = NULL;
MusicalComposition* element;

        if (length == 0){
        head = NULL;
        printf("Пустой список\n");
        }
        else{
        for (int i=0;i<length;i++){
```

```
        char name[80];
        char author[80];

        printf("Введите название композиции:");
        fgets(name, 80, stdin);

        printf("Введите исполнителя композиции:");
        fgets(author, 80, stdin);

        printf("Введите год выхода композиции:");
        fscanf(stdin, "%d", &years[i]);
        char u =  getchar();

        (*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);
        }
head = createMusicalCompositionList(names, authors, years, length);
}
        char name_for_push[80];
        char author_for_push[80];
        int year_for_push;
        char name_for_remove[80];

        printf("Опции для работы со списком:\n");
```

```
printf("1 - Добавить композицию в конец списка\n");
printf("2 - Удаление элемента списка с заданной композицией\n");
printf("3 - Найти количество элементов списка\n");
printf("4 - Вывести названия композиции\n");
printf("5 - Удалить каждый третий элемент списка\n");
printf("6 - Закончить работу со списком\n");

int option = 0;
int ru = 1;
while(ru){
printf("Напишите номер команды:\n");
scanf("%d", &option);
char w = getchar();

switch (option){
        case 1:
                printf("Введите имя композиции:\n");
                fgets(name_for_push, 80, stdin);
                (*strstr(name_for_push,"\n"))=0;

                printf("Введите автора композиции:\n");
                fgets(author_for_push, 80, stdin);
                (*strstr(author_for_push,"\n"))=0;

                printf("Введите год выхода композиции:\n");
                fscanf(stdin, "%d", &year_for_push);

                MusicalComposition* element =
createMusicalComposition(name_for_push, author_for_push, year_for_push);
```

```
                if(head == NULL)
                head = element;
                else
                push(head,element);

                printf("Названия композиций вместе с добавленным
элементом:\n");
                print_names(head);
                break;

        case 2:
                if(head == NULL)
                printf("Пустой список\n");
                else{
                printf("Напишите композицию, которую вы хотите
удалить\n");
                fgets(name_for_remove, 81, stdin);
                (*strstr(name_for_remove,"\n"))=0;
                removeEl(head, name_for_remove);
                printf("Список композиции без удаленного элемента:\n");
                print_names(head);
                }
                break;
        case 3:
                if(head == NULL)
           printf("Пустой список\n");
                else
                printf("Количество композиции равно:%d\n",count(head));
                break;
```

```
            case 4:
                if(head == NULL)
               printf("Пустой список\n");
            else{
               printf("Названия композиции:\n");
                print_names(head);
                }
                break;
            case 5:
                if(head == NULL)
               printf("Пустой список\n");
                else{
                printf("Удаление каждого третьего элемента\n");
                RemoveEvery(&head, 3);
                print_names(head);
                }
                break;
            case 6:
                printf("До скорой встречи!\n");
                ru = 0;
                break;
            default:
                printf("Данные некорректны! Попробуйте снова.\n");
}
}
    for (int i=0;i<length;i++){
    free(names[i]);
    free(authors[i]);
    }
    free(names);
```

```
        free(authors);
        free(years);

        MusicalComposition*tnp = head->next;

        while(tnp != NULL){
        free(tnp->prev->name);
        free(tnp->prev->author);
        free(tnp->prev);
        tnp = tnp->next;
        }
        if(tnp = NULL){
        free(head->name);
       free(head->author);
       free(head);
        }

        free(tnp);
        return 0;
}
```

OPTIONS.C

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

void RemoveNode(MusicalComposition* node){
```

```
        if (node->prev){
        node->prev->next = node->next;
                }
        if (node->next){
        node->next->prev = node->prev;
        }
        free(node);
}


MusicalComposition* RemoveEvery(MusicalComposition** head, size_t number){

size_t counter = 0;
MusicalComposition* node = *head;
MusicalComposition* tmp;

while (node){
        if (++counter == number){
        counter = 0;
        tmp = node;
        node = node->next;
        RemoveNode(tmp);
        }
        else{
        node = node->next;
        }
}
        if (number == 1){
        *head = NULL;
}
```

```
        return *head;
}


MusicalComposition* createMusicalComposition(char* name, char* author,int
year){

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

element->name=(char*)malloc(sizeof(char)*81);
element->author=(char*)malloc(sizeof(char)*81);

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

return element;
}


void push(MusicalComposition* head, MusicalComposition* element){

MusicalComposition* two_element =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
        if ( head->next == NULL ){
        element->next = NULL;
        element->prev = head;
```

```
        head->next = element;
        return;
}
        two_element = head->next;

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


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

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

head->name = (char*)malloc(sizeof(char)*81);
head->author = (char*)malloc(sizeof(char)*81);

strcpy(head->name , array_names[0]);
strcpy(head->author , array_authors[0]);
head->year = array_years[0];

MusicalComposition* two_element;//не выделяю память malloc
```

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


void removeEl(MusicalComposition* head, char* name_for_remove){

	while(head){
	if(!strcmp(head->name,name_for_remove)){
	if(!(head->prev) && !(head->next)){
	free(head);
	return;
}
	else if(!head->prev){
		*head = *(head->next);
		free(head->next->prev);
		head->next->prev = head;
		return;
		}
	else if(!head->next){
		head->prev->next = NULL;
		free(head);
		return;
		}
	else{
```

```
                head->prev->next = head->next;
                head->next->prev = head->prev;
                free(head);
                return;
                }
        }
head = head->next;
        }
}


int count(MusicalComposition* head){

int i=1;
MusicalComposition* two_element =
(MusicalComposition*)malloc(sizeof(MusicalComposition));
two_element = head->next;
        while (two_element != NULL){
        two_element = two_element->next;
        i++;
}
return i;
}


void print_names(MusicalComposition* head){

        do{
        puts(head->name);
        head = head->next;
```

```
}
        while(head != NULL);
}
```

OPTIONS.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);
void push(MusicalComposition* head, MusicalComposition* element);
MusicalComposition* createMusicalCompositionList(char** array_names, char**
array_authors, int* array_years, int n);
void removeEl(MusicalComposition* head, char* name_for_remove);
int count(MusicalComposition* head);
void print_names(MusicalComposition* head);
MusicalComposition* RemoveEvery(MusicalComposition** head, size_t number);
```