**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Двунаправленный список

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Васильев А.А. |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2016

# АННОТАЦИЯ

В данной курсовой работе подробно описывается процесс создания двунаправленного линейного списка и API для работы со списком (создание/удаление элемента, поиск элементов по значениям полей, добавление нового элемента в конец/начало списка). Для реализации данных функций, в работе мною используются широкие возможности языка программирования С (работа с динамической памятью и указателями и т.д.). Также описывается работа с утилитой для сборки проекта “make” и с репозиторием “github”.

Содержание

[АННОТАЦИЯ 2](#_Toc469526864)

[ВВЕДЕНИЕ 4](#_Toc469526865)

[Цель работы 4](#_Toc469526866)

[Постановка задачи 4](#_Toc469526867)

[Индивидуальное задание 5](#_Toc469526868)

[ХОД РАБОТЫ 6](#_Toc469526869)

[Создание отдельных элементов проекта 6](#_Toc469526870)

[Работа с “make” и сборка проекта 9](#_Toc469526871)

[Работа с “github” 9](#_Toc469526872)

[ЗАКЛЮЧЕНИЕ 11](#_Toc469526873)

[приложения 12](#_Toc469526874)

[ПРИЛОЖЕНИЕ 1 12](#_Toc469526875)

[ПРИЛОЖЕНИЕ 2 13](#_Toc469526876)

[ПРИЛОЖЕНИЕ 3 14](#_Toc469526877)

[ПРИЛОЖЕНИЕ 4 15](#_Toc469526878)

[ПРИЛОЖЕНИЕ 5 16](#_Toc469526879)

[ПРИЛОЖЕНИЕ 6 18](#_Toc469526880)

# 

# ВВЕДЕНИЕ

## Цель работы

* Создание двунаправленного связанного списка, составными элементами которого являются структуры.
* Создание API для работы с созданным списком.
* Закрепление на практике полученных знаний о языке С.
* Закрепление знаний о работе с утилитой “make” и репозиторием “github”.

## Постановка задачи

Создайте двунаправленный список музыкальных композиций MusicalComposition и **api** ( ***a****pplication****p****rogramming****i****nterface - в данном случае набор функций*) для работы со списком.

Структура элемента списка (тип - MusicalComposition):

* name - строка неизвестной длины (гарантируется, что длина не может быть больше 80 символов), название композиции.
* author - строка неизвестной длины (гарантируется, что длина не может быть больше 80 символов), автор композиции/музыкальная группа.
* year - целое число, год создания.

Функция для создания элемента списка:

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

Функции для работы со списком:

* MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n); // создает список музыкальных композиций MusicalCompositionList, в котором:
  + ***n****- длина массивов****array\_names****,****array\_authors****,****array\_years****.*
  + поле **name** первого элемента списка соответствует первому элементу списка array\_names (**array\_names[0]**).
  + поле **author** первого элемента списка соответствует первому элементу списка array\_authors (**array\_authors[0]**).
  + поле **year** первого элемента списка соответствует первому элементу списка array\_authors (**array\_years[0]**).
  + Функция возвращает указатель на первый элемент списка*.*
* void push(MusicalComposition\*  head, MusicalComposition\* element); // добавляет **element**  в конец списка **musical\_composition\_list.**
* void removeEl (MusicalComposition\*  head, char\* name\_for\_remove); // удаляет элемент **element** списка, у которого значение **name** равно значению  **name\_for\_remove.**
* int count(MusicalComposition\*  head); //возвращает количество элементов списка.
* void print\_names(MusicalComposition\*  head); //Выводит названия композиций.

Функция “main” для проверки работоспособности функции уже написана.

## Индивидуальное задание

* Добавить в начало списка еще 3 элемента (имя, автор, год - произвольные)
* Удалить все элементы чьи года не превышают n.

# ХОД РАБОТЫ

## Создание отдельных элементов проекта

1. **struct.h**

Создадим заголовочный файл с объявлением нового типа:

struct MusicalComposition{ //Объявляем структуру

char name[80];

char author[80];

int year;

struct MusicalComposition \*next, \* prev; //Указатели на

}; //связанные элементы.

typedef struct MusicalComposition MusicalComposition;

//Присваиваем, для удобства, новое имя типу-структуре

1. **createMC.c**

Создадим файл, который содержит описание функции для создания нового элемента:

#include <string.h>

#include <stdlib.h>

#include "struct.h" //Подключаем файл с объявлением структуры

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

{

MusicalComposition\* tmp=NULL; //Создаем указатель на структуру.

tmp =(MusicalComposition\*)malloc(sizeof(MusicalComposition)); /\*Выделяем под структуру память.\*/

strcpy(tmp->name, name); //Скопируем переданные в функцию

strcpy(tmp->author, author); //значения в поля созданной структуры

tmp->year = year; return tmp; //Возвращаем указатель на структуру

}

1. **createMCList.c**

Создадим файл с описанием функции для создания связанного списка. Содержание данного файла в Приложении 1.

1. **push.c**

Создадим исходник с описанием функции, которая добавляет один элемент в конец списка:

#include <stdlib.h>

#include "struct.h" //Подключаем файл с объявлением структуры

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

{

MusicalComposition\* tmp = head; //Сохраняем указатель на "head".

while(head->next) //Пока указатель на следующий элемент

{ //не будет равен нулю будем передвигать

head = head->next; //указатель на следующий.

} //После цикла "head" указывает на последний элемент

head->next = element; //Вместо указателя на "NULL" присваиваем ему

element->prev = head; //указатель на "element". Предыдущий указатель

element->next = NULL; // пермещаем на"head",а указатель на следующий на

head = tmp; //"NULL". Возвращаем сохраненное значение.

}

1. **removeEl.c**

Создадим файл с описанием функции, которая удаляет элемент с определенным названием композиции. (см. Приложение 2).

1. **count.c**

Создадим файл с функцией, подсчитывающей количество элементов в списке:

#include "struct.h"

int count(MusicalComposition\* head)

{ //Создаем указатель и присваиваем

MusicalComposition\* hold = head; //ему значение передаваемой структ.

int count=0; //Создаем счетчик со значением 0.

while(head) //Цикл "while" будет выполнятся пока

{ //указатель не примет значение "NULL".

count++; //При каждом выполнение цикла увеличиваем

head = head->next; //счетчик на единицу.

}

head = hold; //Присваиваем "head" исходное значение

return count; //Возвращаем счетчик

}

1. **AddToBegin.c**

Создадим файл с описанием функции, которая добавляет перед списком еще n элементов. (см. Приложение 3).

1. **removeYear.c**

Создадим файл с описанием функции, которая удаляет все элементы с годом меньше заданного (см. Приложение 4)

1. **print\_names.c**

Создадим исходник с описанием функции, которая выводит композиции из списка:

#include <stdio.h>

#include "struct.h" //Подключаем заголовочный файл с объявлением структуры

void print\_names(MusicalComposition\* head)

{

MusicalComposition\* hold = head; //Сохраняем значение "head"

while(head) //Пока указатель не равен "NULL"

{ //выводим текущее значение "name"

printf("%s\n", head->name);

head = head->next; //Делаем текущим указатель на след.

} //структуру

head = hold; //Возвращаем исходное значение указателю

};

1. **main.c**

Отредактируем код функции main для удобной проверки (Приложение 5):

* Заменим fgets() на scanf()
* Добавим в конец вызов функций, созданных для решения индивидуального задания
* После каждого изменения списка будем выводить его на экран
* Также будем выводить каждый раз число элементов в списке

1. **func.h**

Создадим заголовочный файл, в который поместим прототипы функций:

MusicalComposition\* createMusicalComposition(char\*, char\*, int);

MusicalComposition\* createMusicalCompositionList(char\*\*, char\*\*, int\*, int);

void push(MusicalComposition\*, MusicalComposition\*);

void removeEl(MusicalComposition\*, char\*);

int count(MusicalComposition\*);

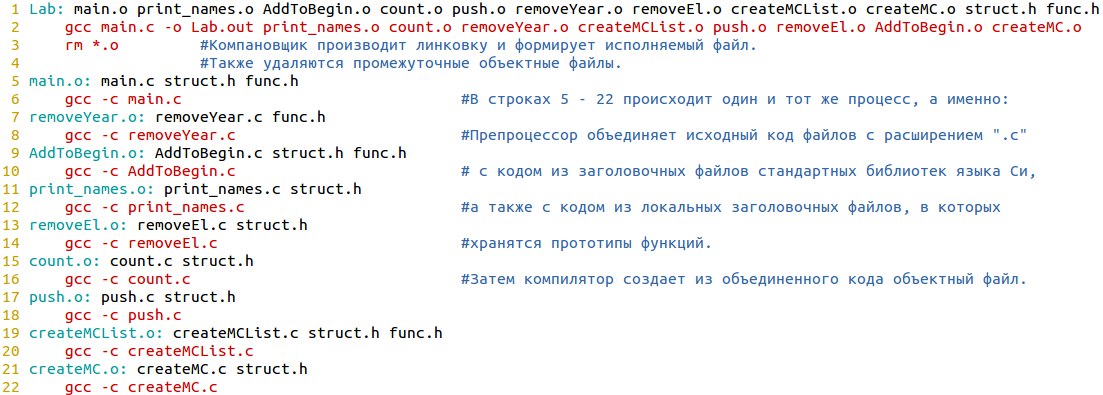
void print\_names(MusicalComposition\*);

MusicalComposition\* AddToBegin(MusicalComposition\*);

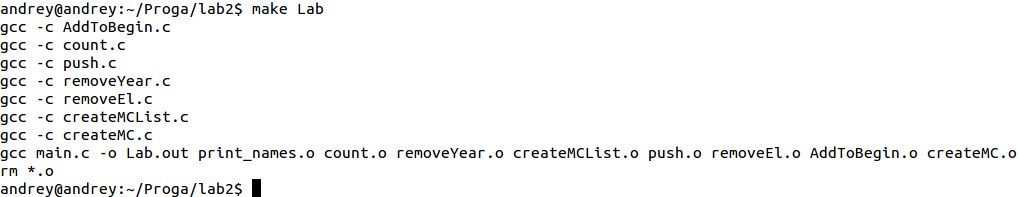
MusicalComposition\* removeYear(MusicalComposition\*);

## Работа с “make” и сборка проекта

1. Для удобства сборки проекта создадим **makefile** следующего содержания:



1. Соберем программу, используя комманду make Lab:

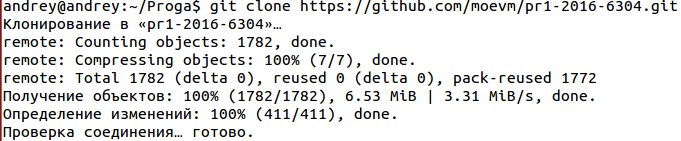


Сборка прошла без ошибок

1. Протестируем работоспособность программы (См. Приложение 6).

## Работа с “github”

1. Клонируем репозиторий группы



1. Переходим в нужную ветку

https://pp.vk.me/c637722/v637722666/23a43/6tFwFmeWshQ.jpg

1. Копируем папку CourseWork в репозиторий
2. Добавляем папку командой **git add CourseWork**
3. Проверим, добавились ли файлы командой **git status**



1. Описываем изменения командой **git commit –m “Added\_new\_work”**
2. Пушим ветку в репозиторий командой **git push origin**
3. Вводим имя пользователя и пароль
4. Операция прошла успешно, файлы помещены в репозиторий

# ЗАКЛЮЧЕНИЕ

Выполнив данную работу, мы закрепили на практике знания о структурах, создав связанный двунаправленный список, состоящий из структур. Мы научились создавать API для работы с такими списками. Также мы попрактиковались в работе с проектом, состоящим из нескольких частей: создали заголовки и файлы с кодом и создали makefile для удобного обращения с таким большим количеством файлов.

# приложения

**ПРИЛОЖЕНИЕ 1**

**createMCList.c**

#include <stdlib.h>

#include "struct.h" //Подключаем заголовочный файл с объявлением структуры

#include "func.h" //и файл с прототипом функции "createMusicalComposition"

MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n)

{

MusicalComposition\* hold = NULL; //Создаем указатели на вспомогательные структуры

MusicalComposition\* temp = NULL; // и присваиваем им нулевые указатели

MusicalComposition\* start=NULL;

start = createMusicalComposition(array\_names[0], array\_authors[0], array\_years[0]); //Создаем первый элемент списка

hold = start; //и присваиваем значение "start" временной переменной, чтобы

int i; //вернуть значение в конце

start->prev = NULL;

for (i = 1; i<=n; i++) /\*Цикл начинается с единицы и выполняется, пока счетчик "i" не будет равен количеству элементов\*/

{

if(i<n){

//При условии: счетчик меньше "n" - присваиваем временной переменной

//"temp" текущее значение. Далее создаем следующий элемент и делаем его текущим.

temp = start;

start->next = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);

start = start->next;

//Теперь элемент "temp" является предыдущим для текущего элемента "start"

start->prev = temp;

} else start->next = NULL; /\*При достижении последнего элемента присваиваем следующему элементу нулевой указатеь\*/

}

start = hold; //Возвращаем "start"у исходное значение, сохраненное раннее.

return start; //Возвращаем указатель на "start"

}

**ПРИЛОЖЕНИЕ 2**

**removeEl.c**

#include <stdlib.h>

#include "struct.h" //Подключаем файл с объявлением структуры

void removeEl(MusicalComposition\* head, char\* name\_for\_remove)

{

MusicalComposition \*hold = NULL; //Создаем переменную и

hold = head; //сохраняем в ней "head".

while(head) //Цикл выполняется пока указатель

{ //станет равен "NULL".

if (strcmp(head->name, name\_for\_remove)==0) //Сравниваем строки

{

if (head->next == NULL){ //Если элемент последний, то текущим

head = head->prev; //становится предыдущий элемент, а

head->next = NULL; //его указатель на след. становится NULL break;

} else if (head->prev == NULL){ //Если удаляемый элемент первый, то

head->next->prev = NULL; //"start" присваивается следующий

start = head->next; //элемент списка, т.е.новое начало списка.

} else {

//В случае, когда элемент является не крайним, перемещаем указатели данным образом

head->next->prev = head->prev;

head->prev->next = head->next;

}

}

head = head->next; /\*Если строки не совпали, делаем текущим указатель на след. структуру \*/

}

head = hold; //Возвращаем "head" сохраненное значение

}

**ПРИЛОЖЕНИЕ 3**

**AddToBegin.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "struct.h" //Подключаем заголовочные файлы собъявлением структуры и с

#include "func.h" //прототипом "createMusicalCompositionList"

MusicalComposition\* AddToBegin(MusicalComposition\* head)

{

int i, k; //Считываем количество printf("Количество добавляемых элементов в начало: "); //добавляемых элементов

scanf("%d", &k);

char\*\* names = (char\*\*)malloc(sizeof(char\*)\*k); //Выделяем память под

char\*\* authors = (char\*\*)malloc(sizeof(char\*)\*k); //списки имен, авторов

int\* years = (int\*)malloc(sizeof(int)\*k); //и годов.

for (i=0;i<k;i++)

{ //С помощью цикла заполняем

char name[80]; //списки значениями

char author[80];

printf("[%d] Композиция, автор и год: ", i+1);

scanf("%s", name); //Для удобства используем scanf()

scanf("%s", author);

scanf("%d", &years[i]);

names[i] = (char\*)malloc(sizeof(char) \* (strlen(name)+1));

authors[i] = (char\*)malloc(sizeof(char) \* (strlen(author)+1));

strcpy(names[i], name); //Копируем считанные значения в список

strcpy(authors[i], author);

}

MusicalComposition\* NewHead; //Создаем "новую голову" списка

//Заполняем новый мини список с помощью созданной раннее функции

NewHead= createMusicalCompositionList(names, authors, years, k);

MusicalComposition\* temp = NewHead; //Сохраняем указатель на голову

while(NewHead->next){

NewHead = NewHead->next; //Передвигаемся по списку до конечного элемента

}

NewHead->next = head; //Связываем конец мини списка с началом основного

head->prev = NewHead;

return temp; //Возвращаем указатель на начало нового дополненного списка

}

**П**РИЛОЖЕНИЕ **4**

**removeYear.c**

#include <stdlib.h>

#include <stdio.h>

#include "struct.h" //Подключаем файл с объявлением структуры

MusicalComposition\* removeYear(MusicalComposition\* head)

{

MusicalComposition \*i, \*start = head; //Создаем вспомогательную переменную

printf("Удалить элементы, чей год меньше: "); //которая будет указывать на начало

int k; //списка

scanf("%d", &k); //Вводим год

while(head){ //Цикл будет выполнятся, пока указатель не ноль

/\*Если год данного элемента меньше введенного, то в зависимости от расположения элемента, данный элемент будет удален при помощи изменения указателей на связанные структуры\*/

if (head->year < k){

if (head->next == NULL){ //Если элемент последний, то текущим

head = head->prev; //становится предыдущий элемент, а

head->next = NULL; //его указатель на след. становится NULL break;

} else if (head->prev == NULL){ //Если удаляемый элемент первый, то

head->next->prev = NULL; //"start" присваивается следующий

start = head->next; //элемент списка, т.е.новое начало списка.

} else {

head->next->prev = head->prev; //В данном случае по аналогии

head->prev->next = head->next; //с removeEl

}

}

head = head->next; //Если условие не выполняется, двигаемся дальше

}

return start; /\*Возвращаем указатель на новое (если изменилось) начало списка \*/

}

**ПРИЛОЖЕНИЕ 5**

**main.c**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include "struct.h"

#include "func.h"

int main(){

int length, i;

printf("Количество элементов: ");

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++)

{

while (getchar() != '\n');

char name[80];

char author[80];

printf("[%d] Название композиции, группа и год: ", i+1);

scanf("%s %s %d", name, author, &years[i]);

names[i] = (char\*)malloc(strlen(name)\*sizeof(char));

strcpy(names[i], name);

authors[i] = (char\*)malloc(strlen(author)\*sizeof(char));

strcpy(authors[i], author);

}

MusicalComposition\* head = createMusicalCompositionList(names, authors, years, length);

printf("Введенный список: \n");

print\_names(head);

int k = count(head);

printf("Кол-во элементов сейчас: %d\n", k);

char name\_for\_push[80];

char author\_for\_push[80];

int year\_for\_push;

char name\_for\_remove[80];

while(getchar() != '\n');

printf("\nКомпозиция, группа и год для добавления: ");

scanf("%s %s %d", name\_for\_push, author\_for\_push, &year\_for\_push);

MusicalComposition\* element\_for\_push = createMusicalComposition(name\_for\_push, author\_for\_push, year\_for\_push);

push(head, element\_for\_push);

printf("Список после дополнения:\n");

print\_names(head);

k = count(head);

while(getchar() != '\n');

printf("Кол-во элементов сейчас: %d\n", k);

printf("\nКомпозиция для удаления: ");

scanf("%s", name\_for\_remove);

removeEl(head, name\_for\_remove);

k = count(head);

printf("Список после удаления:\n");

print\_names(head);

printf("Кол-во элементов сейчас: %d\n", k);

printf("\n");

head = AddToBegin(head);

k = count(head);

printf("Список после добавления в начало:\n");

print\_names(head);

printf("Кол-во элементов сейчас: %d\n", k);

printf("\n");

head = removeYear(head);

k = count(head);

printf("Список поcле удаления нескольких элементов:\n");

print\_names(head);

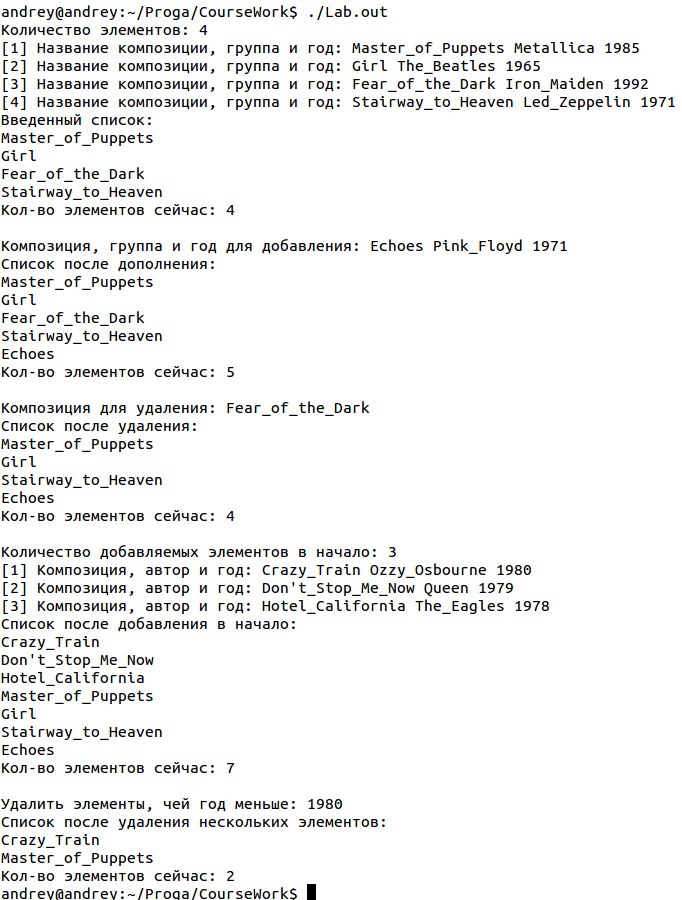
printf("Кол-во элементов сейчас: %d\n", k);

return 0;

}

**ПРИЛОЖЕНИЕ 6**

**Тест работоспособности программы** (одна линия – ввод, две –вывод)

**