**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Двунаправленные линейные списки

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Виноградов К.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| **Студент**  Виноградов Кирилл Алексеевич | | |
| **Группа** 6304 | | |
| **Тема работы (проекта):** Двунаправленные списки | | |
| **Исходные данные:**  Поставлена задача создать двунаправленный список и обеспечить возможность его заполнения, добавления а также удаления элементов разными способами. | | |
| **Содержание пояснительной записки:**   * Оглавление * Введение * Теория двунаправленных списков * Описание необходимых для работы со списком функций. * Описание работы с git и github * Заключение * Исходный код программы | | |
| **Предполагаемый объем пояснительной записки:**  24 страницы. | | |
| **Дата выдачи задания:** 08.12.2016 | | |
| **Дата сдачи реферата:** | | |
| **Дата защиты реферата:** | | |
| Студент |  | Виноградов К.А. |
| Преподаватель |  | Берленко Т.А. |

# **Аннотация**

Создание двунаправленного линейного списка, а также API для работы с ним, средствами языка С. В частности, с использованием сложных типов, указателей на них и работы с динамической памятью. Реализация стандартных операций таких как вставка, удаление, создание элемента списка, поиск по значениям полей и т.д. Работа с проектом в целом: разбиение его на заголовки (header files) и файлы кода (source files), а также написание makefile для сборки программы.

Оглавление

[**Аннотация** 3](#_Toc470338300)

[**Введение** 5](#_Toc470338301)

[*Цель работы:* 5](#_Toc470338302)

[*Формулировка задачи:* 5](#_Toc470338303)

[Индивидуальное задание: 6](#_Toc470338304)

[**Теория** 6](#_Toc470338305)

[**Решение задачи** 6](#_Toc470338306)

[*Создание структуры данных:* 6](#_Toc470338307)

[*Функция создания нового экземпляра данного типа:* 7](#_Toc470338308)

[*Функция создания списка из элементов:* 7](#_Toc470338309)

[*Функция добавления элемента в конец списка:* 8](#_Toc470338310)

[*Функция удаления элемента*: 8](#_Toc470338311)

[*Функция подсчета элементов списка:* 9](#_Toc470338312)

[*Функция печати имен:* 9](#_Toc470338313)

[*Функция добавления трех элементов после каждого четного элемента:* 9](#_Toc470338314)

[*Функция удаления элементов по подстроке ,содержащейся в имени:* 10](#_Toc470338315)

[*Меню* 11](#_Toc470338316)

[*Makefile:* 13](#_Toc470338317)

[*Работа с GitHub:* 14](#_Toc470338318)

[**Примеры работы программы** 15](#_Toc470338319)

[**Заключение** 17](#_Toc470338320)

[**Исходный код программы** 18](#_Toc470338321)

[***Файл mclist.c*** 18](#_Toc470338322)

[***Файл createfunc.c*** 21](#_Toc470338323)

[***Файл addel.c*** 22](#_Toc470338324)

[***Файл remove.c*** 22](#_Toc470338325)

[***Файлы прототипов*** 23](#_Toc470338326)

[ **createfunc.h** 23](#_Toc470338327)

[ **addel.h** 23](#_Toc470338328)

[ **remove.h** 23](#_Toc470338329)

[ **type.h** 24](#_Toc470338330)

# 

# **Введение**

## *Цель работы:*

Обучение применению на практике сложных типов (struct) в языке С. Использование их для реализации сложных структур данных. В частности, одно- и двунаправленных линейных списков. Создание API для работы с ними. Закрепление знаний об указателях (в том числе на сложные типы), динамической памяти, массивах, стандартном вводе-выводе и основных функций библиотек “stdio.h”, “stdlib.h”, “string.h”.

*Формулировка задачи:*

Создайте двунаправленный список музыкальных композиций MusicalComposition и **api** (application***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 (тип элемента MusicalComposition)

* 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]**).
* 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

Индивидуальное задание:

1. Добавить после каждого четного элемента списка еще 3 элемента (имя, автор, год - произвольные).  
   2. Удалить все элементы где в имени автора присутствует подстрока s.

# **Теория**

Двунаправленный список есть набор структур, связанных между собой указателями. В каждой структуре есть указатель на следующую и предыдущую. У первой структуры указатель предыдущую – NULL, и у последней на следующую – NULL. Доступ к списку можно получить через первый элемент структуры – голову списка(head).

Для доступа к следующим элементам структуры необходимо совершить ряд присваивания, вида head = head->next, где head – голова списка.

К предыдущим же доступ можно получить с помощью конструкции head=head->prev.

**Решение задачи**

*Создание структуры данных:*

С помощью типа struct объявим новый сложный тип struct MusicalComposition,. Для удобства используем спецификатор typedef. Выделим отдельный файл “type.h” для объявление типа MusicalComposition. Для связи файла со структурой с остальными файлами в каждом из них добавим строку #include “type.h”.

/\* Структура "Музыкальная композиция \*/

**typedef struct MusicalComposition {**

**char name[80];** // Название композиции

**char author[80];** // Автор композиции

**int year;** // Год написания композиции

**struct MusicalComposition\* next;** // Указатель на следующий элемент

**struct MusicalComposition\* prev;** // Указатель на предыдущий элемент

**}MusicalComposition;**

*Функция создания нового экземпляра данного типа:*

/\* Функция возвращает указатель на созданную структуру \*/

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

/\* Выделение памяти на структуру \*/

**MusicalComposition\* song = (MusicalComposition\*)malloc(sizeof(MusicalComposition));**

**strcpy(song->name, name);** // Копирование в поля структуры нужных данных

**strcpy(song->author, author); //** из ранее выделенных массивов

**song->year = year;**

**song->prev = NULL;** // Структура пока ни с чем не соединяется

**song->next = NULL;** // так как еще не находится в списке

**return song;**

**}**

*Функция создания списка из элементов:*

Создадим сначала один элемент и заполним его поля, а затем с помощью еще двух указателей в цикле создадим остальные элементы и свяжем их и вернем указатель на первый элемент в списке.

/\* Функция создания списка \*/

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

**int i;** // счетчик структур

**MusicalComposition\* temp = createMusicalComposition(array\_names[0], array\_authors[0], array\_years[0]);** // Создание указателя и первой структуры

**MusicalComposition\* head = temp;** //Перенос указателя на первую структуру в head

**for (i=1; i<n; i++)**

**{**

**temp->next = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);** //Создание еще n структур с помощью предыдущей функции

**temp->next->prev = temp;** // Добавление к созданной структуре указателей

**temp = temp->next;** // переход к созданию следующей структуры

**}**

**return head;** // Возвращение указателя на начало списка

**}**

*Функция добавления элемента в конец списка:*

С помощью указателя на следующий элемент пройдем от головного элемента до конца и там создадим новый элемент, связав его с предыдущим.

void push(MusicalComposition\* head, MusicalComposition\* element) //объявление ф-ции

{

while(head->next){ //пока существует следующий элемент

head=head->next; //двигаемся вперед

}

head->next=element; //следующим за последним теперь является element

element->prev=head; //а предыдущий перед element-ом теперь является последний

}

*Функция удаления элемента*:

Найдем нужный элемент и изменим связь с предыдущим и следующим.

Int removeEl(MusicalComposition\* head, char\* name\_for\_remove){ //объявляем функцию

if(((head->next)==NULL)&&((head->prev)==NULL)){

if((strcmp(name\_for\_remove, head->name))==0){

printf("Со смертью этого листа нить вашей судьбы обрывается.\nЗагрузите новый лист, чтобы восстановить течение судьбы,\n или живите дальше в проклятом мире, который сами и создали.\n");

return 0; //проверяем условие на единственность элемента

} //при удовлетворении программа завершает работу

}

else{

while (head!=NULL){ //пока настоящий элемент существует

if (strcmp(name\_for\_remove, head->name) == 0) //сравниваем совпадение имен

break; //и останавливаем если совпадают

else

head = head->next; //а если не совпадают то движемся по списку дальше

}

if (head == NULL){ //если дошли до конца

printf("Element does not exists\n"); //пишем что элемента не существует

}

else if ((head->prev!=NULL)&&(head->next!=NULL)){ //если попадается элемент в середине списка

head->prev->next = head->next; //перекидываем указатели с предыдущего элемента на

head->next->prev = head->prev; //следующий и в обратную сторону

}

else if (head->next == NULL){ //если элемент последний

head->prev->next = NULL; //передаем ноль в указатель с предпоследнего на следующий

}

else if(head->prev == NULL){ //если элемент первый

while(head->next!=NULL){ //пока существует следующий

strcpy(head->name,head->next->name); //копируем данные на с элемента на

strcpy(head->author, head->next->author); //элемент к началу

head->year = head->next->year;

head=head->next; //постепенно продвигаясь по списку

}

head->prev->next = NULL; //удаляем последний элемент так как он по факту

head->prev = NULL; //равен предпоследнему, а от первого мы избавились

}

}

}

*Функция подсчета элементов списка:*

Просто движемся по списку до конца, увеличивая счетчик на один за каждый проход списка.

**int count(MusicalComposition\* head){** //объявляем

**int i = 0;** //счетчик

**while (head!=NULL)** //пока настоящий элемент существует

**{**

**head = head->next;** //продвигаясь дальше

**i++;** //увеличиваем счетчик

**}**

**return i;** //возвращаем его

**}**

*Функция печати имен:*

Данная функция делается аналогично предыдущей с пошаговым выводом данных каждого элемента

**void print\_list(MusicalComposition\* head){** //объявляем

**while (head!=NULL)** //пока настоящий элемент существует

**{**

**printf("%s %s %d\n", head->name, head->author, head->year);** //выводим данные

**head = head->next;** //и продвигаемся дальше

**}**

## *Функция добавления трех элементов после каждого четного элемента:*

Для реализации этой функции мы уже имеем функцию добавления элемента в конец списка. Будет удобно ей воспользоваться.

**void** **pushExt(MusicalComposition\* head)** //объявляем

**{**

**int k=0;** //два счетчика

**int i=0;**

**while(head->next){** //идем пока существует следующий элемент

**head=head->next;**

**k++;** //увеличиваем счетчик

**if(k%2!=0){** //цикл получился сдвинутым, поэтому ищем «нечетные» элементы

**if(head->next!=NULL){** //если существует следующий элемент

**MusicalComposition \*nexttmp=head->next;** //записываем указатель на него в переменную

**head->next=NULL;** //после чего делаем настоящий элемент «последним»

**for(i=0;i<3;i++){**

**push(head, createMusicalComposition("ne ochen`", "ez", 322));** //и добавляем 3 элемента

**}** //в «конец»

**while(head->next){** //после чего перемещаемся по ним до конца

**head=head->next;**

**}**

**head->next=nexttmp;** //и передаем адрес из переменной в крайний добавленный элемент

**}**

**else{** //а если мы и так дошли до последнего элемента

**for(i=0;i<3;i++){**

**push(head, createMusicalComposition("2", "ez", 322));** //то просто добавляем еще три

**}**

**while(head->next){**

**head=head->next;**

**}**

**}**

**}**

**}**

**}**

## *Функция удаления элементов по подстроке ,содержащейся в имени:*

Здесь мы тоже имеем некоторые наработки – функцию удаления одного элемента. Нужно только реализовать поиск подстроки в строке. С эти нам поможет функция strstr библиотеки string.h.

**void removeExt(MusicalComposition\* head)** //объявляем функцию

**{**

**char\* str=(char\*)malloc(80\*sizeof(char));** //выделяем память под подстроку

**scanf("%s", str);** //считываем ее с ввода

**int k=0;** //счетчик содержания подстроки

**MusicalComposition\* bgn=head;** //сохраняем указатель на первый элемент

**while(head){**

**if((strstr(head->author, str))==NULL)** //проводим проверку на наличие не содержащих

**k=1;**  //подстроку элементов

**head=head->next;**

**}**

**if(k==1){** //если есть не содержащие

**head=bgn;** //восстанавливаем начало

**while(head!=NULL){** //пока существует настоящий элемент

**if((strstr(head->author, str))!=NULL){** //производим поиск подстроки в строке

**removeEl(head, head->name);** //и если удовлетворяет то удаляем элемент

removeEl(head, head->name);

**}**

**head=head->next;** //переходим на следующий

**}**

**}**

**else{** //если нет

**printf("Со смертью этого листа нить вашей судьбы обрывается.\nЗагрузите новый лист, чтобы восстановить течение судьбы,\n или живите дальше в проклятом мире, который сами и создали.\n");**

**return 0;** //завершаем работу программы

**}**

**}**

*Меню*

Функцию меню реализуем в main. Необходимо ее зациклить. Поэтому просто заключим весь код функции в цикл While(1). Из этого цикла можно выйти только принудительно по соответствующей команде от пользователя. Варианты действий реализуем через switch. Учтем формирование нового списка при его отсутствии

/\* Данная функция выводит пользователю список доступных операций, считывает его ответ и передает его в главную функцию \*/

**int menu()** //объявляем ф-цию

**{**

**char ch;** //объявляем переменную

**printf("What do you wanе to do?\n1)Print the names of the songs\n2)Add one element\n3)Add some elements\n4)Remove an element\n5)Remove some elements\n6)Add 3 el. after even el.\n7)Exit\n");** //выводим варианты на экран

**scanf("%c", &ch);** //считываем переменную с ввода

**return ch;** //возвращаем ее

**}**

**int main(){**

/\*Выделяем память под головной элемент и обычный элемент списка

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

**MusicalComposition\*element\_for\_push = (MusicalComposition\*)malloc(sizeof(MusicalComposition));**

**head=NULL;** //записываем нуль в головной элемент

**int length, length2, pmenu, pushy;** //объявляем переменный

**char pushn[80];** //выделяем память под добавляемые элементы

**char pusha[80];**

**int i=0;**

**int f=1;**

**while(f==1){ {** //зацикливаем меню

**if(head==NULL){** //проверка на заполненность списка

/\*Заполнение списка\*/

**printf("\033[2J");** //очищаем экран

**printf("\033[0;0f");** //перемещаем каретку в верхний левый угол

**printf("You need to fill the list up\nEnter a number of elements first\n");**

**scanf("%d\n", &length);**  //сканирование начального кол-ва элементов

**if(length==0){** //если нет элементов то завершаем программу

**goto exit;**

**}**

**printf("For now you have to write down the name, author and year of out for each composition\n");**

**char\*\* names = (char\*\*)malloc(sizeof(char\*)\*length);** //выделение памяти под

**char\*\* authors = (char\*\*)malloc(sizeof(char\*)\*length);** //два массива строк

**int\* years = (int\*)malloc(sizeof(int)\*length);** //и один чисел

**for (i=0;i<length;i++)**

**{**

**char name[80];** //цикл считывания данных элементов с ввода

**char author[80];**

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

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

**strcpy(names[i], name);** //копирование из массива в структуру

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

**strcpy(authors[i], author);** //копирование из массива в структуру

**}**

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

**}** //создание списка

**int n=count(head);** //переменная подсчета элементов

**pmenu=menu();** //переменная для возвращенного значения ф-ции меню

**switch(pmenu){** //switch-конструкция

**case '1':**

**printf("\033[2J");**

**printf("\033[0;0f");**

**print\_list(head);** //вывод данных списка на экран

**getchar();** //здесь и далее для удобства работы

**break;**

**case '2':**

**printf("\033[2J");**

**printf("\033[0;0f");**

**printf("Enter the name, author and year for this element\n");**

**scanf("%s %s %d", pushn, pusha, &pushy);**

**element\_for\_push = createMusicalComposition(pushn,pusha, pushy);**

**push(head, element\_for\_push);** //добавление элемента в конец

**getchar();**

**break;**

**case '3':**

**printf("\033[2J");**

**printf("\033[0;0f");**

**printf("Enter the amount of elements you want to push\n");**

**scanf("%d", &length2);**

**for(i=0;i<length2;i++){**

**printf("Enter the name, author and year for this element\n");**

**scanf("%s %s %d", pushn, pusha, &pushy);**

**element\_for\_push = createMusicalComposition(pushn, pusha, pushy);**

**push(head, element\_for\_push);** /\*добавление нескольких элементов в конец\*/

**}**

**getchar();**

**break;**

**case '4':**

**printf("\033[2J");**

**printf("\033[0;0f");**

**printf("Enter the name of composition you want to remove\n");**

**scanf("%s", pushn);**

**f=removeEl(head, pushn); getchar();** //удаление одного элемента

**break;**

**case '5':**

**printf("\033[2J");**

**printf("\033[0;0f");**

**printf("Enter the author`s name you want to remove\n");**

**f=removeExt(head);** //удаление нескольких элементов

**getchar();**

**break;**

**case '6':**

**printf("\033[2J");**

**printf("\033[0;0f");**

**printf("Enter the data for each of three elements\n");**

**pushExt(head);** //добавление трех элементов после каждого четного

**getchar();**

**break;**

**case '7':**

**printf("\033[2J");**

**printf("\033[0;0f");**

**goto exit;** //выход из программы

**break;**

**default:**

**printf("You, probably, miss the right button. Try again\n");**

**break;** //объявление границ верного ввода

**}**

**}**

**exit:** //точка выхода из цикла

**return 0;**

**}**

*Makefile:*

Создадим поочерёдно объектные файлы функций и главной функции, после скомпилируем их вместе, также предусмотрена функция очистки после сборки проекта. В отдельные файлы вынесем функции для создания структур(createfunc.c), функции для удаления по подстроке(remove.c) и функции для добавления трех элементов(addel.c). Для каждого из этих файлов создадим заголовочные файлы \*.h, куда поместим прототипы используемых в файле функций. В каждом из файлов не забудем включить type.h, описанный выше.

**mclist: mclist.o** //Сбор всех объектных файлов воедино

**gcc -o mclist mclist.c addel.c createfunc.c remove.c**

**rm \*.o** //удаление объектных файлов

**mclist.o: mclist.c** // Создание объектного файла mclist.o

**gcc -c mclist.c**

**createfunc.o: createfunc.c createfunc.h type.h** // Создание объектного файла createfunc.o

**gcc -c createfunc.c**

**addel.o: addel.c addel.h createfunc.h** // Создание объектного файла addel.o

**gcc -c addel.c**

**remove.o: remove.c remove.h createfunc.h** // Создание объектного файла remove.o

**gcc -c remove.c**

*Работа с GitHub:*

Создадим новую ветку git checkout –b “Vinogradov\_kurswork”

Создадим новую папку mkdir Vinogradov\_kurswork

Добавим файлы проекта в данную папку

Проиндексируем изменения git add Vinogradov\_kurswork

Создадим коммит git commit –m “#Kurswork added”

Отправим изменения на удаленный репозиторий git push origin

**Примеры работы программы**

Приведем примеры работы функций программы. Результат работы некоторых функций можно увидеть только при выводе списка на экран.

|  |  |  |
| --- | --- | --- |
| **Проверяемая**  **функция** | **Ввод** | **Результат** |
| Создание списка | 4  InisMona Eluveitie 2008  TwilightTavern Enseiferum 2009  ThePursuitOfVikings AmonAmarath 2004  BalkanicHeart Niburta 2012 | - |
| Добавление элемента | 2  TheCrimsonWineOfBattle  Folkearth 2014 | InisMona Eluveitie 2008  TwilightTavern Enseiferum 2009  ThePursuitOfVikings AmonAmarath 2004  BalkanicHeart Niburta 2012  TheCrimsonWineOfBattle  Folkearth 2014 |
| Добавление двух элементов | 3  TriumphAndLoss [Noumena](https://vk.com/search?c%5bq%5d=Noumena&c%5bsection%5d=audio&c%5bperformer%5d=1)  2006  草原牧歌 [SongOfChu](https://vk.com/search?c%5bq%5d=Song%20Of%20Chu&c%5bsection%5d=audio&c%5bperformer%5d=1)  2016 | InisMona Eluveitie 2008  TwilightTavern Enseiferum 2009  ThePursuitOfVikings AmonAmarath 2004  BalkanicHeart Niburta 2012  TheCrimsonWineOfBattle  Folkearth 2014  TriumphAndLoss [Noumena](https://vk.com/search?c%5bq%5d=Noumena&c%5bsection%5d=audio&c%5bperformer%5d=1)  2006  草原牧歌 [SongOfChu](https://vk.com/search?c%5bq%5d=Song%20Of%20Chu&c%5bsection%5d=audio&c%5bperformer%5d=1)  2016 |
| Удаление элемента | 4  ThePursuitOfVikings | InisMona Eluveitie 2008  TwilightTavern Enseiferum 2009  BalkanicHeart Niburta 2012  TheCrimsonWineOfBattle  Folkearth 2014  TriumphAndLoss [Noumena](https://vk.com/search?c%5bq%5d=Noumena&c%5bsection%5d=audio&c%5bperformer%5d=1)  2006  草原牧歌 [SongOfChu](https://vk.com/search?c%5bq%5d=Song%20Of%20Chu&c%5bsection%5d=audio&c%5bperformer%5d=1)  2016 |
| Удаление нескольких элементов | 5  o | InisMona Eluveitie 2008  TwilightTavern Enseiferum 2009  BalkanicHeart Niburta 2012 |
| Добавление неск. элементов | 6  Hell Disturbed 2005  WordUp! Korn 2004  ForestOfMist Ocelon 2016 | InisMona Eluveitie 2008  TwilightTavern Enseiferum 2009  Hell Disturbed 2005  WordUp! Korn 2004  ForestOfMist Ocelon 2016  BalkanicHeart Niburta 2012 |
| Удаление элемента | 4  InisMona | TwilightTavern Enseiferum 2009  Hell Disturbed 2005  WordUp! Korn 2004  ForestOfMist Ocelon 2016  BalkanicHeart Niburta 2012 |
| Добавление элемента | 2  AGunshotToTheHeadOfTrepidation [Trivium](https://vk.com/search?c%5bq%5d=Trivium&c%5bsection%5d=audio&c%5bperformer%5d=1) 2005 | TwilightTavern Enseiferum 2009  Hell Disturbed 2005  WordUp! Korn 2004  ForestOfMist Ocelon 2016  BalkanicHeart Niburta 2012  AGunshotToTheHeadOfTrepidation [Trivium](https://vk.com/search?c%5bq%5d=Trivium&c%5bsection%5d=audio&c%5bperformer%5d=1) 2005 |
| Удаление нескольких элементов | 5  um | Hell Disturbed 2005  WordUp! Korn 2004  ForestOfMist Ocelon 2016  BalkanicHeart Niburta 2012 |
| Удаление элемента | 4  Hell | WordUp! Korn 2004  ForestOfMist Ocelon 2016  BalkanicHeart Niburta 2012 |
| Завершение работы | 7 | Программа завершена |

# **Заключение**

Выполнив данную курсовую работу, мы освоили и закрепили на практике написание программы с применением структур в качестве элементов сложных типов представления данных таких, как одно- и двунаправленные линейные списки, а также API для работы с ними, в частности, функции вставки, удаления, подсчета и вывода элементов списка. Повторили работу с проектом в целом: разбиение его на заголовки (header files) и файлы кода (source files), написание makefile для сборки программы, так же темы указатели, динамическая память, массивы и функции стандартных библиотек.

# **Исходный код программы**

***Файл mclist.c***

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <stddef.h>

#include "type.h"

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

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

{

while(head->next){

head=head->next;

}

head->next=element;

element->prev=head;

}

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

if(((head->next)==NULL)&&((head->prev)==NULL)){

if((strcmp(name\_for\_remove, head->name))==0){

printf("Со смертью этого листа нить вашей судьбы обрывается.\nЗагрузите новый лист, чтобы восстановить течение судьбы,\n или живите дальше в проклятом мире, который сами и создали.\n");

return 0;

}

}

else{

while (head!=NULL){

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

break;

else

head = head->next;

}

if (head == NULL){

printf("Element does not exists\n");

}

else if ((head->prev!=NULL)&&(head->next!=NULL)){

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

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

}

else if (head->next == NULL){

head->prev->next = NULL;

}

else if(head->prev == NULL){

while(head->next!=NULL){

strcpy(head->name,head->next->name);

strcpy(head->author, head->next->author);

head->year = head->next->year;

head=head->next;

}

head->prev->next = NULL;

head->prev = NULL;

}

}

}

int count(MusicalComposition\* head)

{

int i=0;

while(head!=NULL){

head=head->next;

i++;

}

return i;

}

void print\_list(MusicalComposition\* head)

{

while(head!=NULL){

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

head=head->next;

}

}

int menu()

{

char ch;

// printf("\033[2J"); // Clear the entire screen.

// printf("\033[0;0f"); // Move cursor to the top left hand corner

printf("What do you want to do?\n1)Print the names of the songs\n2)Add one element\n3)Add some elements\n4)Remove an element\n5)Remove some elements\n6)Add 3 el. after even el.\n7)Exit\n");

scanf("%c", &ch);

return ch;

}

int main(){

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

MusicalComposition\* element\_for\_push = (MusicalComposition\*)malloc(sizeof(MusicalComposition));

head=NULL;

int length, length2, pmenu, pushy;

char pushn[80];

char pusha[80];

int i=0;

int f=1;

while(f==1){

if(head==NULL){

printf("\033[2J");

printf("\033[0;0f");

printf("You need to fill the list up\nEnter a number of elements first\n");

scanf("%d\n", &length);

if(length==0){

goto exit;

}

printf("For now you have to write down the name, author and year of out for each composition\n");

char\*\* names = (char\*\*)malloc(sizeof(char\*)\*length);

char\*\* authors = (char\*\*)malloc(sizeof(char\*)\*length);

int\* years = (int\*)malloc(sizeof(int)\*length);

// printf("For now you have to write down the name, author and year of out for each composition\n");

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

{

char name[80];

char author[80];

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

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

strcpy(names[i], name);

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

strcpy(authors[i], author);

}

head = createMusicalCompositionList(names, authors, years, length);

}

int n=count(head);

pmenu=menu();

switch(pmenu){

case '1':

printf("\033[2J");

printf("\033[0;0f");

print\_list(head);

getchar();

break;

case '2':

printf("\033[2J");

printf("\033[0;0f");

printf("Enter the name, author and year for this element\n");

scanf("%s %s %d", pushn, pusha, &pushy);

element\_for\_push = createMusicalComposition(pushn,pusha, pushy); push(head, element\_for\_push);

getchar();

break;

case '3':

printf("\033[2J");

printf("\033[0;0f");

printf("Enter the amount of elements you want to push\n");

scanf("%d", &length2);

for(i=0;i<length2;i++){

printf("Enter the name, author and year for this element\n");

scanf("%s %s %d", pushn, pusha, &pushy);

element\_for\_push = createMusicalComposition(pushn, pusha, pushy);

push(head, element\_for\_push);

}

getchar();

break;

case '4':

printf("\033[2J");

printf("\033[0;0f");

printf("Enter the name of composition you want to remove\n");

scanf("%s", pushn);

f=removeEl(head, pushn);

getchar();

break;

case '5':

printf("\033[2J");

printf("\033[0;0f");

printf("Enter the author`s name you want to remove\n");

f=removeExt(head);

getchar();

break;

case '6':

printf("\033[2J");

printf("\033[0;0f");

printf("Enter the data for each of three elements\n");

pushExt(head);

getchar();

break;

case '7':

printf("\033[2J");

printf("\033[0;0f");

goto exit;

break;

default:

printf("You, probably, miss the right button. Try again\n");

getchar();

}

}

exit:

return 0;

***Файл createfunc.c***

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <stddef.h>

#include "type.h"

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

MusicalComposition\* song=(MusicalComposition\*)malloc(sizeof(MusicalComposition));

strcpy(song->name, name);

strcpy(song->author, author);

song->year=year;

MusicalComposition\* next=NULL;

MusicalComposition\* prev=NULL;

return song;

}

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

int i;

MusicalComposition\* temp = createMusicalComposition(array\_names[0], array\_authors[0], array\_years[0]);

MusicalComposition\* head = temp;

for (i=1; i<n; i++)

{

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

temp->next->prev = temp;

temp = temp->next;

}

return head;

}

***Файл addel.c***

#include <stdio.h>

#include <stdlib.h>

#include <stddef.h>

#include <string.h>

#include "type.h"

void pushExt(MusicalComposition\* head)

{

int k=0;

int i=0;

while(head->next){

head=head->next;

k++;

if(k%2!=0){

if(head->next!=NULL){

MusicalComposition \*nexttmp=head->next;

head->next=NULL;

for(i=0;i<3;i++){

push(head, createMusicalComposition("ne ochen`", "ez", 322));

}

while(head->next){

head=head->next;

}

head->next=nexttmp;

}

else{

for(i=0;i<3;i++){

push(head, createMusicalComposition("2", "ez", 322));

}

while(head->next){

head=head->next;

}

}

}

}

}

***Файл remove.c***

#include <stdio.h>

#include <stdlib.h>

#include <stddef.h>

#include <string.h>

#include "type.h"

int removeExt(MusicalComposition\* head)

{

char\* str=(char\*)malloc(80\*sizeof(char));

scanf("%s", str);

int k=0;

MusicalComposition\* bgn=head;

while(head){

if((strstr(head->author, str))==NULL)

k=1;

head=head->next;

}

if(k==1){

head=bgn;

while(head!=NULL){

if((strstr(head->author, str))!=NULL){

removeEl(head, head->name);

removeEl(head, head->name);

}

head=head->next;

}

}

else{

printf("Со смертью этого листа нить вашей судьбы обрывается.\nЗагрузите новый лист, чтобы восстановить течение судьбы,\n или живите дальше в проклятом мире, который сами и создали.\n");

return 0;

}

}

***Файлы прототипов***

* **createfunc.h**

#include "type.h"

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

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

* **addel.h**

#include "td.h"

void pushExt(MusicalComposition\* head);

* **remove.h**

#include "type.h"

void removeExt(MusicalComposition\* head);

* **type.h**

typedef struct MusicalComposition{

char name[80];

char author[80];

int year;

struct MusicalComposition\* next;

struct MusicalComposition\* prev;

}MusicalComposition;