**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

Лабораторная работа №4

**по дисциплине «Программирование»**

Тема: Линейные списки

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6304 |  | Смотрова А.А. |
| Преподаватель |  | Кринкин К. В. |

Санкт-Петербург

2016

Цель работы:

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

Задание:

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]**).

*Аналогично для второго, третьего, ...****n-1****-го элемента массива.*

*! длина массивов****array\_names, array\_authors, array\_years****одинаковая и равна n, это проверять не требуется.*

*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

Ход работы:

1. Создадим описание структуры MusicalComposition:

**struct** MusicalComposition { //объявили структуру

**char** name[**80**]; //строка не более 80 символов

**char** author[**80**];

**int** year;

**struct** MusicalComposition \* next;//следующий элемент списка

**struct** MusicalComposition \* previous; //предыдущий элемент списка

};

1. Определим новый тип MusicalComposition, чтобы впоследствии не использовать struct.

**typedef** **struct** MusicalComposition MusicalComposition;

1. Создадим саму структуру MusicalComposition

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

{

MusicalComposition \*mc = MusicalComposition\*) malloc (**sizeof**(MusicalComposition));//выделяем память по размеру структуры

strcpy(mc->name, name); //скопируем в поле «Имя» введенное имя

strcpy(mc->author, author); //в поле автор – введенного автора

mc->year = year;

mc->previous = NULL; //нет ссылок на предыдущий

mc->next = NULL; //и следующий элемент

**return** mc;

};

1. Создадим функцию для создания списка

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

{

MusicalComposition \*head = createMusicalComposition(array\_names[**0**], array\_authors[**0**], array\_years[**0**]); //вводим шапку таблицы

MusicalComposition \*previous = head; //все поля previous ==полям head

MusicalComposition \*current;

**for** (**int** i = **1**; i < n; ++i) //начинаем заполнять текущий узел

{

current = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);

current->previous = previous; //для текущего узла предыдущий узел – это previous

previous->next = current; //для предыдущего узла следующий – это текущий

previous = current; //передвигаем узел на один

}

**return** head;

}

1. Напишем функцию для добавления элемента в конец списка:

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

{

**while** (head->next) //пока следующий узел существует

head = head->next; //двигаемся по списку

head->next = element; //для последнего эл-та делаем ссылку на новый эл-т

element->previous = head; //связываем новый элемент с последним

element->next = NULL; //у последнего элемента нет следующего

}

1. Теперь – функция для удаления элемента по названию песни.

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

{

MusicalComposition \*current = head;

**while** (current->next) //пока есть следующий элемент

{

**if** (strcmp(current->name, name\_for\_remove) == **0**) //если имя и строка для удаления совпали

**if** ((current->previous) == **0**) //если это первый элемент

{

MusicalComposition \*newcurrent = current->next; //снимаем связь со следующим элементом (перенаправляем на newcurrent)

strcpy(current->name, newcurrent->name);//копируем

strcpy(current->author, newcurrent->author); //инф-цию

current->year = newcurrent->year;//в newcurrent

newcurrent->next->previous = current;

current->next = newcurrent->next;

free(newcurrent);//сбросили newcurrent

}

Else //если это не первый элемент

{

current->previous->next = current->next;

current->next->previous = current->previous;

free(current);

}

current = current->next;//сдвигаемся на один узел дальше

}

**if** ((strcmp(current->name, name\_for\_remove) == **0**) && (current->next == **0**)) //если это последний элемент

{

current->previous->next = NULL; //указатель на след. Эл-т предыдущего делаем NULL

free(current);

}

}

1. Следующая функция считает количество элементов в списке.

**int** **count**(MusicalComposition \*head) { //возвращает число элементов списка

**int** c=**0**;

**while** (head) {

c++;

head=head->next;

}

**return** c;

}

1. Последняя функция реализует печать всех названий песен из списка.

**void** **print\_names**(MusicalComposition \*head){

MusicalComposition \*current = head;

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

printf("%s**\n**", current->name); //печатай название

current = current->next;//двигаемся к след. узлу

}

printf("%s**\n**", current->name); //печатаем последний узел

}

Вывод:

В процессе работы были получены навыки создания и управления одно- и двунаправленными линейными списками. В результате был получен двунаправленный список музыкальных композиций MusicalComposition и  набор функций для работы со списком.

приложение

**int** **main**(){

**int** length;

scanf("%d**\n**", &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++)

{

**char** name[**80**];

**char** author[**80**];

fgets(name, **80**, stdin);

fgets(author, **80**, stdin);

fscanf(stdin, "%d**\n**", &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);

}

MusicalComposition\* 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**];

fgets(name\_for\_push, **80**, stdin);

fgets(author\_for\_push, **80**, stdin);

fscanf(stdin, "%d**\n**", &year\_for\_push);

(\*strstr(name\_for\_push,"**\n**"))=**0**;

(\*strstr(author\_for\_push,"**\n**"))=**0**;

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

fgets(name\_for\_remove, **80**, stdin);

(\*strstr(name\_for\_remove,"**\n**"))=**0**;

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

**int** k = count(head);

printf("%d**\n**", k);

push(head, element\_for\_push);

k = count(head);

printf("%d**\n**", k);

removeEl(head, name\_for\_remove);

print\_names(head);

k = count(head);

printf("%d**\n**", k);

**return** **0**;

}