**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №4**

**по дисциплине «Программирование»**

Тема: Линейные списки

|  |  |  |
| --- | --- | --- |
| Студенка гр. 6304 |  | Блинникова Ю. И. |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2016

**Цель:**

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

**Задание:**

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 (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

В функции main написана некоторая последовательность вызова команд для проверки работы вашего списка.

**Содержание:**

**Код программы и его комментарий:**

#include<stdlib.h> // заголовочный файл динамического распределения памяти

#include <stdio.h> // заголовочный файл ввода-вывода

#include <string.h> // заголовочный файл для функции для работы со строками

struct MusicalComposition { // описание структуры MusicalComposition

char name[80]; // имя песни

char author[80];//автор песни

int year;//год выпуска

struct MusicalComposition \*next;//указательна следующий элемент

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

};

typedef struct MusicalComposition MusicalComposition;//присваивание структуре имя MusicalComposition

MusicalComposition\* createMusicalComposition(char \*name, char \*author, int year)//функция создания музыкальной композиции

{

MusicalComposition\* Composition = (MusicalComposition\*)malloc(sizeof(MusicalComposition));//создание структуры с именем Composition и выделение памяти под неё

strcpy(Composition->name, name);//присвоит песне название

strcpy(Composition->author, author);//присвоить песне автора

Composition->year = year;//присвоить песне год

Composition->next = NULL;//присвоить песне указатель на следующую структуру, которая ни на что не указывает

Composition->prev = NULL;//присвоить песне указатель на предыдущую структуру, которая ни на что не указывает

return Composition;//возврат значения этой структуры

}

MusicalComposition \*createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n)// функция для работы со списком MusicalComposition(создание плейлиста)

{

MusicalComposition \*head = createMusicalComposition(array\_names[0], array\_authors[0], array\_years[0]);//создание указателя на первую структуру

MusicalComposition \*prev = head;//указатель на предыдущий элемент, который указывает на head

MusicalComposition \*current;// создаём указатель, который используется только в этой функции для хождения по циклу и создания структур

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

{

current = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);//указатель на созданный плейлист

current->prev = prev;//указатель структуры указывает на предыдущую структуру

prev->next = current;//указатель next предыдущей структуры указывает на настоящую структуру

prev = current;//присвоить указателю на предыдущую структуру указатель на настоящую структуру

}

return head;// возвращает указатель на настоящую структуру

}

void push(MusicalComposition\* head, MusicalComposition\* element) // добавить в плейлист композицию

{

while (head->next) //пока существует следующая структура

head = head->next; // идем по плейлисту

head->next = element; //дойдя до последней структуры, присваиваем ее указателю next указатель на новый элемент.

element->prev = head; // указатель структуры element указывает на последнюю структуру плейлиста

element->next = NULL; // указатель next структуры element ни на что не указывает

}

void removeEl(MusicalComposition\*head, char\*name\_for\_remove) // функция удаления элемента с именем name\_for\_remove

{

while (head) // пока существует структура

{

if (strcmp(head->name, name\_for\_remove) == 0)// если имена совпали, то смотрим, где именно

{

if (head->next == NULL) // если в конце,то

{

head->prev->next = NULL;// присвоить указателю next предпоследней структуры значение NULL

return; // закончить

}

else if (head->prev == NULL) // если в начале, то

{

head->next->prev = NULL; // присвоить значению указателя prev второй структуры NULL

return;// закончить

}

head->prev->next = head->next; //если в середине, то присваиваем указателю next предудущей структуры адрес следующей структуры

head->next->prev = head->prev; // присваиваем указателю prev следующей структуры адрес предыдущей структуры

break;//закончить

}

head = head->next;// идем по плейлисту

}

}

int count(MusicalComposition\*head)// подсчет количества композиций в плейлисте

{

int count = 0;

while (head) // пока существует структура

{

count++; // количество увеличиваем на 1

head = head->next;//идем по структуре

}

return count;// возвращаем количество композиций

};

void print\_names(MusicalComposition \*head) // вывод названий композиций

{

while (head) // пока существует структура

{

printf("%s\n", head->name); // вывод на экран название

head = head->next; //идем по структуре

}

}

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;

}

**Вывод:**

Создали двунаправленный список музыкальных композиций MusicalComposition. Научились работать со структурами: создавать, описывыть и применять их.