**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

**КУРСОВАЯ РАБОТА**

**по дисциплине «Программирование»**

Тема: Линейные списки

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Ваганов Н.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

Цель работы:создать двунаправленный список музыкальных композиций MusicalComposition и api (набор функций) для работы со списком.

Ход работы:

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

Дополнительное задание: создать функцию, создающую срез списка: на вход функции подается указатель на начало структуры и два индекса (i, j), i - номер первого элемента, j - последнего.

**Код:**

1. Структура элемента списка (тип – MusicalComposition)

*typedef struct MusicalComposition*

*{*

*char\* name;*

*char\* author;*

*int year;*

*struct MusicalComposition \*next;*

*struct MusicalComposition \*prev;*

*}MusicalComposition;*

1. Функция для создания элемента списка (тип элемента MusicalComposition).

Данной функции передают три параметра: название, имя автора и год создания композиции; которые она записывает в новый элемент списка, для которого резервирует память с помощью функции malloc. Функция возвращает указатель на созданный элемент списка.

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

*{*

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

*next->name = name;*

*next->author = author;*

*next->year = year;*

*next->next=NULL;*

*next->prev=NULL;*

*return next;*

*}*

1. Функция для создания списка музыкальных композиций MusicalCompositionList.

Функция получает количество существующих композиций и три массива: массив названий, массив авторов и массив годов созданий; и формирует линейный двунаправленный список.

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

*{*

*MusicalComposition\* head;*

*int i=1;*

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

*MusicalComposition\* ptr=head;*

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

*{*

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

*ptr->next->prev = ptr;*

*ptr->next->next=NULL;*

*ptr=ptr->next;*

*}*

*return head;*

*}*

1. Функция для добавления нового элемента в конец списка.

Данной функции передается указатель на начало линейного списка и элемент, который необходимо добавить. Функция перемещается по списку, пока не доходит до последнего элемента, а затем соединяет последний и новый элементы списка посредством указателей.

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

*{*

*while (head->next != NULL)*

*{*

*head=head->next;*

*}*

*head->next=element;*

*element->next=NULL;*

*}*

1. Функция для удаления элемент списка, с определённым названием композиции.

Функция получает указатель на начало списка и название композиции, которую нужно удалить. Перемещаясь по списку, она сверяет название композиции с переданной строкой. При совпадении значений функция связывает предыдущий и следующий элементы посредством указателей, тем самым удаляя найденный элемент из списка.

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

*{*

*while (head->next != NULL)*

*{*

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

*{*

*if (head->prev == NULL)*

*{*

*\*head = \*head->next;*

*head->prev=NULL;*

*}*

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

*{*

*head->prev->next = NULL;*

*}*

*else*

*{*

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

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

*}*

*}*

*head=head->next;*

*}*

*}*

1. Функция для подсчёта количества элементов списка.

Функция получает указатель на первый элемент списка, и пробегая весь список, увеличивает счётчик counter на каждом элементе списка.

*int count(MusicalComposition\* head)*

*{*

*int counter = 0;*

*while(head)*

*{*

*counter++;*

*head=head->next;*

*}*

*return counter;*

*}*

1. Функция для выведения названия композиций.

Данной функции передается указатель на первый элемент списка. Перебирая все элементы, функция печатает названия каждой композиции.

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

*{*

*while(head)*

*{*

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

*head=head->next;*

*}*

*}*

1. Функция для сортировки списка по году создания по убыванию.

*int cut(MusicalComposition\* head, int i, int j)*

*{*

*int count = 1;*

*if (i < j && i != 0 && j != 0 && head != NULL)*

*{*

*for (count;count <= j;count++)*

*{*

*if (count != i && count < i)*

*{*

*\*head=\*head->next;*

*head->prev=NULL;*

*}*

*if (count > i && count < j)*

*{*

*head=head->next;*

*}*

*else if (count == j) head->next=NULL;*

*}*

*return 0;*

*}*

*else return 1;*

*}*

**Вывод:** в ходе работы получены навыки работы с двунаправленными списками, создан набор функций для работы с линейными списками.