**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по лабораторной работе №4**

**по дисциплине «Программирование»**

Тема: Линейные списки

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Эквас В.И. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

Цель работы: **с**оздать 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

**Описание кода:**

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 выделяет память с помощью функции malloc. Функция возвращает указатель на созданный элемент списка.

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

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

*composition->name = name;*

*composition->author = author;*

*composition->year = year;*

*composition->next = NULL;*

*composition->prev = NULL;*

*return composition;*

*}*

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

*MusicalComposition \*prev = head;*

*MusicalComposition \*composition;*

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

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

*composition->prev = prev;*

*prev->next = composition;*

*prev = composition;*

*}*

*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) {*

*MusicalComposition \*composition = head;*

*while (composition->next) {*

*composition = composition->next;*

*}*

*composition->next = element;*

*element->prev = composition;*

*}*

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, если это возможно, в иных случаях результат зависит от наличия соседних элементов, тем самым удаляя найденный элемент из списка.

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

*{*

*MusicalComposition\* current = head;*

*while (current)*

*{*

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

*{*

*if (current->next != NULL && current->prev != NULL)*

*{*

*current->prev->next = current->next;*

*current->next->prev = current->prev;*

*current = current->next;*

*}*

*else if (current->next == NULL && current->prev!=NULL)*

*{*

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

*current = NULL;*

*}*

*else if (current->next!=NULL && current->prev==NULL)*

*{*

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

*head->prev = NULL;*

*current = head;*

*}*

*else if (current->next == NULL && current->prev == NULL)*

*{*

*current->year=-1;*

*current = NULL;*

*}*

*}else{*

*current = current->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)*

*{*

*MusicalComposition\* composition = head;*

*int count = 0;*

*while (composition && composition->year != -1)*

*{*

*count++;*

*composition = composition->next;*

*}*

*return count;*

*}*

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)*

*{*

*MusicalComposition\* composition = head;*

*while (composition && composition->year!=-1)*

*{*

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

*composition = composition->next;*

*}*

*}*

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