**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.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Рыбин А.С. |
| Преподаватель |  | Кринкин К.В. |

Санкт-Петербург

2016

Оглавление

[Цель: 3](#_Toc468537147)

[Задание: 3](#_Toc468537148)

[Структура элемента списка (тип – struct MusicalComposition) 3](#_Toc468537149)

[Функция для создания элемента списка (тип элемента struct MusicalComposition) 3](#_Toc468537150)

[Функции для работы со списком: 3](#_Toc468537151)

[Содержание: 4](#_Toc468537152)

[Структура "Музыкальная композиция 4](#_Toc468537153)

[Функция создания и инициализации нового экземпляра структуры "MusicalComposition" 4](#_Toc468537154)

[Функция создания списка 4](#_Toc468537155)

[Добавление элемента в конец списка 5](#_Toc468537156)

[Функция удаления элемента 5](#_Toc468537157)

[Подсчет элементов списка 6](#_Toc468537158)

[Печать имен 6](#_Toc468537159)

[Вывод: 6](#_Toc468537160)

Цель:

Написание программы с линейных списков и работа с ними (вставка, удаление элементов)

Задание:

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 – struct 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 struct MusicalComposition)

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

Функции для работы со списком:

* Struct 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]**).

Функция возвращает указатель на первый элемент списка.

* void push (struct MusicalComposition\* head, struct MusicalComposition\* element); добавляет **element** в конец списка **musical\_composition\_list**
* void removeEl (struct MusicalComposition\* head, char\* name\_for\_remove); // удаляет элемент **element** списка, у которого значение **name** равно значению **name\_for\_remove**
* int count(struct MusicalComposition\* head); //возвращает количество элементов списка
* void print\_names(struct MusicalComposition\* head); //Выводит названия композиций

Содержание:

Структура "Музыкальная композиция

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

};

Функция создания и инициализации нового экземпляра структуры "MusicalComposition"

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

/\* Выделим память для нового экземпляра структуры "MusicalComposition" \*/

struct MusicalComposition\* new\_element = (struct MusicalComposition\*)malloc(sizeof(struct MusicalComposition));

/\* Проинициализируем поля параметрами, переданными в функцию \*/

strcpy(new\_element->name, name);

strcpy(new\_element->author, author);

new\_element->year = year;

new\_element->next = NULL;

new\_element->prev = NULL;

/\* Вернем указатель на новый экземпляр структуры "MusicalComposition" \*/

return new\_element;

}

Функция создания списка

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

/\* Указатель на первый элемент, который вернет функция \*/

struct MusicalComposition\*head =NULL;

/\* Указатель для движения по списку \*/

struct MusicalComposition\*ptr = NULL;

/\* Фиктивный указатель для работы со списком \*/

struct MusicalComposition\*f\_ptr = NULL;

/\* Заполнение списка \*/

for (int i = 0; i < n; i++)

{

/\* Выделяем память для нового элемента \*/

ptr = (struct MusicalComposition\*)malloc(sizeof(struct MusicalComposition));

/\* Создадим первый элемент отдельно чтобы вернуть head \*/

if (i == 0)

{

head = ptr;

strcpy(ptr->name, array\_names[i]);

strcpy(ptr->author, array\_authors[i]);

ptr->year = array\_years[i];

ptr->next = NULL;

ptr->prev = NULL;

f\_ptr = ptr;

}

else

{

/\* Указатель на следующий элемент у фиктивного указателя равен вновь созданному \*/

f\_ptr->next = ptr;

strcpy(ptr->name, array\_names[i]);

strcpy(ptr->author, array\_authors[i]);

ptr->year = array\_years[i];

ptr->next = NULL;

/\* Указатель на предыдущий элемент равен фиктивному указателю \*/

ptr->prev = f\_ptr;

/\* Двигаем фиктивный указатель вместе с новым \*/

f\_ptr = ptr;

}

}

/\* Вернем указатель на первый элемент списка \*/

return head;

}

Добавление элемента в конец списка

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

{

/\* Двигаем указатель до последнего элемента \*/

while (head->next)

head = head->next;

/\* Указатель на следующий равен переданному в функцию параметру \*/

head->next = element;

/\* Указатель на предыдущий равен текущему \*/

element->prev = head;

element->next = NULL;

}

Функция удаления элемента

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

{

/\* Пока не конец списка двигаем указатель и сравниваем строки \*/

while (head)

{

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

{

/\* Если элемент, который надо удалить последний то разрываем связь с предыдущим \*/

if (head->next == NULL)

{

head->prev->next = NULL;

return;

}

/\* Если элемент, который надо удалить первый то разрываем связь со следующим

Указатель на первый элемент меняем на второй элемент\*/

else if (head->prev == NULL)

{

head->next->prev = NULL;

\*head = \*head->next;

return;

}

/\* Меняем значения поля next у предыдущего элемента \*/

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

/\* Меняем значение поля prev у следующего элемента \*/

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

break;

}

head = head->next;

}

}

Подсчет элементов списка

int count(struct MusicalComposition\*head)

{

int count = 0;

while (head)

{

count++;

head = head->next;

}

return count;

}

Печать имен

void print\_names(struct MusicalComposition\* head)

{

while (head)

{

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

head = head->next;

}

}

**Вывод:**

Выполнив данную лабораторную работу, мы освоили и закрепили на практике написание программы с применением структур в качестве элементов сложных типов представления данных таких, как одно- и дву- направленные линейные списки, а также API для работы с ними, в частности, функции вставки, удаления, подсчета и вывода элементов списка.