**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 |  | Матюшина М.Е. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

Цель работы: Создать двунаправленный список музыкальных композиций MusicalComposition и **api** (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 композицийMusicalCompositionList, 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
* intcount (MusicalComposition\*  head); //возвращает количество элементов списка
* void print\_names(MusicalComposition\*  head); //Выводит названия композиций

**Ход работы.**

**Описание кода:**

* + Создание структуры MusicalComposition.

typedef struct MusicalComposition {

char\* name;

char\* author;

int year;

struct MusicalComposition \*next;

struct MusicalComposition \*prev;

} MusicalComposition;

MusicalComposition \* head; //создаем глобальную переменную head – начало списка

* + Функция для создания элемента списка (тип элемента MusicalComposition).

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

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

music->name = name;

music->author = author;

music->year = year;

music->next = NULL;

music->prev = NULL;

return music;

}

Функции для работы со списком MusicalComposition.

1. Функция для создания списка музыкальных композиций MusicalCompositionList.

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

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

MusicalComposition \*tmp = head;

int i;

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

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

tmp->next = pp;

pp->prev = tmp;

tmp = tmp->next;

}

return;

}

Функция для добавления нового элемента в конец списка.

MusicalComposition \* push( MusicalComposition\* element) {

MusicalComposition \* tmp = head;

if (tmp->next!=NULL){

while ( tmp->next != NULL) {

tmp = tmp->next;

}

}

tmp->next = element;

element->prev= tmp;

element->next=NULL;

return head;

}

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

void removeEl( char\* name\_for\_remove){

MusicalComposition \*pp;

pp=head;

MusicalComposition \*tmp;

tmp=head;

MusicalComposition\* next;

MusicalComposition\* prev;

while ( pp!=NULL){

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

tmp=head;

while (\*tmp->name!=\*name\_for\_remove) {

tmp=tmp->next;

}

if (tmp==head) {

head=head->next;

}

else if(tmp->next==NULL) {

prev=tmp->prev;

prev->next=NULL;

}

else{

next=tmp->next;

prev=tmp->prev;

next->prev=prev;

prev->next=next;

}

}

pp=pp->next;

}

}

1. Функция для подсчёта количества элементов списка.

int count() {

MusicalComposition \* tmp = head;

int count = 0;

while (tmp != NULL) {

++count;

tmp = tmp->next;

}

return count;

}

1. Функция для выведения названия композиций.

void print\_names() {

MusicalComposition \* tmp = head;

while (tmp != NULL) {

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

tmp = tmp->next;

}

if(tmp==NULL) { printf("" ); }

}

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