**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №4**

**по дисциплине «Программирование»**

Тема: *Линейные списки*

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Малышенко Ю.И. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

**Цель работы:** научиться использовать линейные списки, создать **api (application programming interface)** для работы со списком.

**Ход работы:**

В ветке malushenko\_lr4 репозитория pr-2016-6303 создал папку Lr4 с проектом, состоящим из 1 файла ***lr4.c***

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

Ниже приведен код программы:

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

struct MusicalComposition

{

char \*name;

char \*author;

int year;

int vivod;

struct MusicalComposition\* next;

struct MusicalComposition\* prev;

};

typedef struct MusicalComposition MusicalComposition;

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

{

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

MusComp->name = name;

MusComp->author = author;

MusComp->year = year;

MusComp->vivod = 1;

MusComp->prev = NULL;

MusComp->next = NULL;

return MusComp;

}

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

{

MusicalComposition \*head;

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

MusicalComposition \*tmp = head;

MusicalComposition\* next;

int i;

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

{

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

tmp->next = next;

next->prev = tmp;

tmp = tmp->next;

}

return head;

}

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

{

MusicalComposition\* tmp = head;

MusicalComposition\* push = createMusicalComposition(element->name, element->author, element->year);

if (head->vivod == 0)

\*head = \*push;

else

{

while(tmp->next != NULL)

tmp = tmp->next;

tmp->next = push;

push->prev = tmp;

}

}

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

{

MusicalComposition\* tmp = head;

while(tmp->next)

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

{

if(tmp->next != NULL && tmp->prev != NULL)

{

tmp->prev->next = tmp->next;

tmp->next->prev = tmp->prev;

tmp = tmp->next;

}

else if(tmp->prev == NULL)

{

\*tmp = \*(tmp->next);

tmp->prev = NULL;

}

}

else

tmp = tmp->next;

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

{

if(tmp->prev == NULL)

head->vivod = 0;

else

tmp->prev->next = NULL;

}

}

int count(MusicalComposition\* head)

{

MusicalComposition\* tmp = head;

int count = 0;

while(tmp)

if(tmp->vivod == 0)

return 0;

else

{

count++;

tmp = tmp->next;

}

return count;

}

void print\_names(MusicalComposition\* head)

{

MusicalComposition\* tmp = head;

if(tmp->vivod ==1)

while(tmp)

{

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

tmp = tmp->next;

}

}

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);

int i;

for (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("изначально:%d\n", k);

push(head, element\_for\_push);

k = count(head);

printf("после добавления:%d\n", k);

removeEl(head, name\_for\_remove);

k = count(head);

printf("после удаления структур с именем name for remove:%d\n", k);

free(names);

free(authors);

free(years);

return 0;

}

Созданная папка с файлом загружена в репозиторий на Github с помощью следующих команд:

* git add Lab4
* git commit –m Lab4
* git push origin malyshenko\_lr4

**Вывод:** в ходе лабораторной работы получены навыки работы со структурами и списками в языке Си, которые успешно применены на практике при создании программы ***lr4.c***.