**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. 6303 |  | Ваганов Н.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

**Цель работы**

Научиться использовать линейные списки, создать api (application programming interface) для работы со списком.

**Ход работы**

1. В локальном репозитории создана новая ветка *vaganov\_lr4*

git checkout -b vaganov\_lr4

1. В папку ветки помещен отчёт и код лабораторной работы

3.В файле *lr\_4.c* записан программный код по заданию к лабораторной работе.

4. Ниже приведен код программы:

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

// Описание структуры MusicalComposition

typedef struct MusicalComposition

{

char\* name;

char\* author;

int year;

struct MusicalComposition \*next;

struct MusicalComposition \*prev;

}MusicalComposition;

// Создание структуры MusicalComposition

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;

}

// Функции для работы со списком MusicalComposition

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;

}

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

{

while (head->next != NULL)

{

head=head->next;

}

head->next=element;

element->next=NULL;

}

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;

}

}

int count(MusicalComposition\* head)

{

int counter = 0;

while(head)

{

counter++;

head=head->next;

}

return counter;

}

void print\_names(MusicalComposition\* head)

{

while(head)

{

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

head=head->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);

for (int 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("%d\n", k);

push(head, element\_for\_push);

k = count(head);

printf("%d\n", k);

removeEl(head, name\_for\_remove);

print\_names(head);

k = count(head);

printf("%d\n", k);

return 0;

}

**Вывод**

В ходе лабораторной работы получены навыки работы с линейными списками, создан api(application programming interface) для работы со списком, с помощью которого реализована программа для работы со списком музыкальных композиций.