**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 |  | Сергеенков М.Ю. |
| Преподаватель |  | Кринкин К.В. |

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 *sergeenkov\_lr4*

git checkout -b sergeenkov\_lr4

1. Создана папка *sergeenkov\_lr4*

mkdir sergeenkov\_lr4

1. В вышеуказанной папке создан файл *lab4.cpp*

touch lab4.cpp

1. В файле *lab4.cpp* написан программный код по заданию к лабораторной работе.

*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*

1. Помимо указанных в задании функций для работы со списком написаны вспомогательные функции:

* int elementInList(MusicalComposition \*head, char\* element)

Возвращает 1, если в списке есть элемент, у которого поле name равно строке element, и 0 в противном случае.

* MusicalComposition \* deleteHead(MusicalComposition\* head)

Функция для удаления корня списка и назначения головой списка идущего за ним элемента. Получает на вход текущий корень списка и возвращает указатель на новый корень спсика.

1. Далее приведено краткое описание работы программы.

Считывается длина списка, затем считываются названия композиций, имена авторов и года написаний композиций, которые помещаются в соответствующие динамические массивы. Затем создается указатель на корневой элемент списка. С помощью функции createMusicalCompositionList из ранее созданных массивов создается двунаправленный список. Функция createMusicalComposition “упаковывает” элементы в структуру.

После этого считываются данные о музыкальной композиции, которую необходимо добавить в список, “упаковываются” в структуру и с помощью функции push добавляются в конец списка.

Наконец, считывается название композиции, которую необходимо удалить. С помощью функции removeEl элемент списка, в котором поле name соответствует названию композиции для удаления, удаляется из связного списка.

1. Ниже приведен код программы:

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

typedef struct MusicalComposition MusicalComposition;

struct MusicalComposition{

char \*name;

char \*author;

int year;

MusicalComposition\* next;

MusicalComposition\* prev;

};

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

// Функции для работы со списком MusicalComposition

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

MusicalComposition \* deleteHead(MusicalComposition\* head);

int elementInList(MusicalComposition \*head, char\* element);

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

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

int count(MusicalComposition\* head);

void print\_names(MusicalComposition\* head);

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;

}

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

MusicalComposition \*tmp;

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

tmp->name=name;

tmp->author=author;

tmp->year=year;

return tmp;

}

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

MusicalComposition \*head, \*ptr;

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

ptr=head;

head->name=array\_names[0];

head->author=array\_authors[0];

head->year=array\_years[0];

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

{

MusicalComposition\* temp;

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

temp->prev=ptr;

temp->next=NULL;

temp->name=array\_names[i];

temp->author=array\_authors[i];

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

ptr->next=temp;

ptr=ptr->next;

}

return head;

}

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

MusicalComposition \*ptr ,\*temp;

ptr=head;

//ищем хвостовой элемент списка

for (int i = 0; i < (count(head)-1); ++i)

{

ptr=ptr->next;

}

element->next=NULL;

element->prev=ptr;

ptr->next=element;

}

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

MusicalComposition \*ptr;

ptr=head;

MusicalComposition\* next;

MusicalComposition\* prev;

while (elementInList(head, name\_for\_remove)==1){

ptr=head;

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

ptr=ptr->next;

}

if (ptr->prev==NULL){

\*head=\*deleteHead(head);

}

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

prev=ptr->prev;

prev->next=NULL;

free(ptr);

}

else{

next=ptr->next;

prev=ptr->prev;

next->prev=prev;

prev->next=next;

free(ptr);

}

}

}

int count(MusicalComposition\* head){

MusicalComposition \*ptr;

ptr=head;

int count=0;

do{

++count;

ptr=ptr->next;

}while (ptr!=NULL);

return count;

}

void print\_names(MusicalComposition\* head){

MusicalComposition \*ptr;

ptr=head;

while (ptr!=NULL){

for (int i = 0; i < strlen(ptr->name); ++i)

{

putchar(ptr->name[i]);

}

putchar('\n');

ptr = ptr->next;

}

}

int elementInList(MusicalComposition \*head, char\* element){

MusicalComposition \*ptr;

ptr=head;

int flag=0;

while (ptr!=NULL){

if (strcmp(ptr->name, element)==0){

flag=1;

break;

}

else{

ptr=ptr->next;

}

}

if (flag==1){

return 1;

}

else{

return 0;

}

}

MusicalComposition \* deleteHead(MusicalComposition\* head) {

MusicalComposition \*temp;

temp = head->next;

temp->prev = NULL;

free(head); // освобождение памяти текущего корня

return(temp); // новый корень списка

}

1. Проиндексированы изменения в локальном репозитории.

git status

git add sergeenkov\_lr4

1. Сделан коммит в локальный репозиторий.

git commit -m “add lab4”

1. Изменения отправлены в удаленный репозиторий.

git push origin sergeenkov\_lr4

1. Сделан *pull request* в ветку master.

**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.