**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 |  | Шевченко Д.В. |
| 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 *shevchenko\_lr4*

git checkout -b shevchenko\_lr4

1. В папку ветки помещен отчёт и код лабораторной работы

3.В файле *main.c* записан программный код по заданию к лабораторной работе.

4. Ниже приведен код программы:

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

typedef struct MusicalComposition {

char name[80];

char author[80];

int year; //ОПИСАНИЕ ПРОТОТИПА СТРУКТУРЫ

struct MusicalComposition \*next; // УКАЗАТЕЛЬ НА СЛЕДУЮЩИЙ ЭЛЕМЕНТ СПИСКА

struct MusicalComposition \*prev; // УКАЗАТЕЛЬ НА ПРЕДЫДУЩИЙ ЭЛЕМЕНТ СПИСКА

} MusicalComposition;

//ПРОТОТИПЫ ФУНКЦИЙ

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

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

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

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

int count(MusicalComposition\* head);

void print\_names(MusicalComposition\* head);

void removeElchetniy(MusicalComposition\* head);

int main() {

int length;

scanf("%d\n", &length); // СЧИТЫВАЕТСЯ ДЛИННА СПИСКА

char\*\* names = (char\*\*)malloc(sizeof(char\*)\*length); // ДВУМЕРНЫЙ МАССИВ

char\*\* authors = (char\*\*)malloc(sizeof(char\*)\*length); // ДЛЯ ИМЕН И АВТОРОВ

int\* years = (int\*)malloc(sizeof(int)\*length); // МАССИВ ДЛЯ ХРАНЕНИЯ ЛЕТ

for (int i = 0;i<length;i++) // ЗАПОЛНЕНИЕ МАССИВОВ ДАННЫХ ДЛЯ СПИСКА

{

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); // СОЗДАНИЕ СПИСКА И ЗАПИСЬ ЕГО КОРНЕВОГО ЭЛЕМЕНТА В УКАЗАТЕЛЬ head

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); //СЧИТЫВАНИЕ СТРОКИ НАЗВАНИЯ ПЕСНИ,

fgets(author\_for\_push, 80, stdin); // АВТОРА, А ТАК-ЖЕ СЧИТЫВАНИЕ ГОДА

fscanf(stdin, "%d\n", &year\_for\_push); // ДЛЯ СОЗДАНИЯ НОВОГО ЭЛЕМ. СПИСКА

(\*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);

// СОЗДАНИЕ НОВОГО ЭЛЕМЕНТА СПИСКА

fgets(name\_for\_remove, 80, stdin); // СЧИТЫВАНИЕ НАЗВАНИЯ ПЕСНИ, ЭЛЕМЕНТ СПИСКА ВКЛЮЧАЮЩИЙ ЕГО БУДЕТ УДАЛЕН ВПОСЛЕДСТВИИ

(\*strstr(name\_for\_remove, "\n")) = 0;

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

// ВЫВОД ПОЛЕЙ КОРНЕВОГО ЭЛЕМЕНТА СПИСКА

int k = count(head); // ПОДСЧЕТ ЭЛЕМЕНТОВ СПИСКА ИХ ВЫВОД

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

push(head, element\_for\_push); // ПОМЕЩЕНИЕ РАНЕЕ СОЗДАННОГО ЭЛЕМЕНТА

// В КОНЕЦ СПИСКА

k = count(head); // ПЕРЕПОДСЧЕТ ЭЛЕМЕНТОВ СПИСКА И ВЫВОД

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

removeEl(head, name\_for\_remove); // УДАЛЕНИЕ ЭЛЕМЕНТА СПИСКА С НАЗВАНИЕМ

// ТРЕКА СОВПАДАЮЩИМ С name\_for\_remove

print\_names(head); // ВЫВОД ВСЕХ НАЗВАНИЙ ТРЕКОВ НА КОНСОЛЬ

k = count(head); // ПЕРЕПОДСЧЕТ ЭЛЕМЕНТОВ СПИСКА И ВЫВОД

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

removeElchetniy(head); // УДАЛЕНИЕ ВСЕХ ЧЁТНЫХ ЭЛЕМЕНТОВ СПИСКА

print\_names(head); // ВЫВОД НАЗВАНИЙ ТРЕКОВ ПОСЛЕ УДАЛЕНИЯ ЭЛЕМЕНТОВ СПИСКА

return 0;

}

// ФУНКЦИЯ СОЗДАЮЩАЯ СПИСОК НА ОСНОВЕ ПОЛУЧЕННЫХ НАЗВАНИЙ ТРЕКОВ, АВТОРОВ, И ГОДОВ // ПРОИЗВОДСТВА

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

{

MusicalComposition \*ptmp = (MusicalComposition\*)malloc(sizeof(MusicalComposition)); // БУФЕРНЫЙ УКАЗАТЕЛЬ

strcpy(ptmp->name, array\_names[0]);

strcpy(ptmp->author, array\_authors[0]);

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

ptmp->next = NULL;

ptmp->prev = NULL;

MusicalComposition \*head = ptmp;

for (int i = 1; i < n; i++) // В ЦИКЛЕ ЗАПОЛНЯЮТСЯ ПОЛЯ И РАСШИРЯЕТСЯ СПИСОК

// ПОКА НЕ БУДЕТ ДОСТИГНУТ ПОСЛЕДНИЙ ЭЛЕМЕНТ

{

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

ptmp->next = tmp;

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

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

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

tmp->prev = ptmp;

ptmp = tmp;

}

ptmp->next = NULL; // УКАЗАТЕЛЬ НА ПОСЛЕДНИЙ ЭЛЕМЕНТ ПОЛУЧАЕТ ЗНАЧЕНИЕ NULL

return head; // ВОЗВРАЩАЕТ УКАЗАТЕЛЬ НА КОРЕНЬ СПИСКА

}

// ЭТА ФУНКЦИЯ ПРОСТО ПОЛУЧАЕТ ЗНАЧЕНИЕ И ПОДГОТАВЛИВАЕТ ЭЛЕМЕНТ СПИСКА НА ИХ ОСНОВ

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

{

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

strcpy(track->name, name);

strcpy(track->author, author);

track->year = year; // ЗАПОЛНЕНИЕ ПОЛЕЙ

track->next = NULL;

track->prev = NULL;

return track;

}

// ФУНКЦИЯ ПОМЕЩАЮЩАЯ НОВЫЙ ЭЛЕМЕНТ СПИСКА СОЗДАННЫЙ ФУНКЦИЕЙ ВЫШЕ В КОНЕЦ СПИСКА

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

{

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

tmp = head;

while (tmp->next != NULL) // ФУНКЦИЯ ИЩЕТ УКАЗАТЕЛЬ НА ПОСЛЕДНИЙ ЭЛЕМЕНТ

{

tmp = tmp->next;

}

tmp->next = element; // ФУНКЦИЯ ПРИВЯЗЫВАЕТ РАНЕЕ СОЗДАННЫЙ ЭЛЕМЕНТ К КОНЦУ

// СПИСКА

element->prev = tmp;

}

// ФУНКЦИЯ УДАЛЯЮЩАЯ ЭЛЕМЕНТ СПИСКА НАЗВАНИЕ КОТОРОГО СОВПАДАЕТ С name\_for\_remove

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

{

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

tmp = head;

while (strcmp(tmp->name, name\_for\_remove)) // ПОИСК ЭЛЕМЕНТА СОДЕРЖАЩЕГО

// name\_for\_remove

{

tmp = tmp->next;

}

if (tmp->prev == NULL) // ЕСЛИ УДАЛЯЕТСЯ ПЕРВЫЙ ЭЛЕМЕНТ ТО КОРЕНЬ СПИСКА

// ПЕРЕДВИГАЕТСЯ НА СЛЕДУЮЩИЙ ЭЛЕМЕНТ

// СДЕЛАНО ДЛЯ ТОГО ЧТО-БЫ ИЗБЕЖАТЬ ССЫЛКИ НА NULL PTR

{

tmp = tmp->next;

tmp->prev = NULL;

\*head = \*tmp;

}

else if (tmp->next == NULL) // АНАЛОГИЧНО ДЛЯ ПОСЛЕДНЕГО ЭЛЕМЕНТА, ТОЛЬКО

// ПОСЛЕДНИЙ ЭЛЕМЕНТ ПРОСТО УДАЛЯЕТСЯ БЕЗ

// ПЕРЕМЕЩЕНИЯ КОРНЯ СПИСКА

{

tmp = tmp->prev;

tmp->next = NULL;

}

else // ЕСЛИ НЕ ПЕРВЫЙ И НЕ ПОСЛЕДНИЙ ТО СВЯЗЬ ПРЕДЫДУЩЕГО СО СЛЕДУЮЩИМ ЭЛ.

{

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

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

}

}

// ФУНКЦИЯ ДЛЯ ПОДСЧЕТА ЭЛЕМЕНТОВ СПИСКА

int count(MusicalComposition\* head)

{

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

tmp = head;

int counter = 1;

while (tmp->next != NULL) // ЦИКЛ ПРОСТО…СЧИТАЕТ ЭЛЕМЕНТЫ! ДА-ДА! ИМЕННО ТАК

{

tmp = tmp->next;

counter++;

}

return counter;

}

// ВЫВОД НАЗВАНИЙ ВСЕХ ТРЕКОВ НА КОНСОЛЬ

void print\_names(MusicalComposition \* head)

{

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

tmp = head;

while (tmp->next != NULL) // ЦИКЛ ПРОСТО ВЫВОДИТ НАЗВАНИЯ ВСЕХ ТРЕКОВ СПИСКА

{

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

tmp = tmp->next;

}

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

}

// ФУНКЦИЯ УДАЛЯЮЩАЯ ВСЕ ЧЁТНЫЕ ЭЛЕМЕНТЫ СПИСКА

void removeElchetniy(MusicalComposition\* head)

{

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

tmp = head;

int counter = 1;

while (tmp != NULL) // ЦИКЛ ПЕРЕДВИГАЕТСЯ ПО СПИСКУ И ЕСЛИ ЭЛЕМЕНТ ЧЕТНЫЙ

{ // УДАЛЯЕТ ЕГО

if (counter % 2 == 0) // АНАЛОГИЧНО С ФУНКЦИЕЙ remove\_El

{ // (Серьезно, почти весь код копипаста из неё)

if (tmp->prev == NULL)

{

tmp = tmp->next;

tmp->prev = NULL;

\*head = \*tmp;

}

else if (tmp->next == NULL)

{

tmp = tmp->prev;

tmp->next = NULL;

}

else

{

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

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

}

}

tmp = tmp->next;

counter++;

}

printf("NYA PoKa\n"); // NYA PoKa

}

**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.