**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

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6303 |  | Горбунова А.П. |
| Преподаватель |  | Берленко Т.А |

Санкт-Петербург

2016

**Цель работы.**

Создать двунаправленный список музыкальных композиций MusicalComposition и **api** ( ***a****pplication****p****rogramming****i****nterface - в данном случае 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 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]**).
* **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

**Ход работы.**

1. Создала файл main.c

|  |
| --- |
|  |
| #include <stdlib.h> |
|  | #include <stdio.h> |
|  | #include <string.h> |
|  |  |
|  | struct MusicalComposition |
|  | { |
|  | char \*name; |
|  | char \*author; |
|  | int year; |
|  | struct MusicalComposition \*next, \*prev; |
|  | }; |
|  |  |
|  | typedef struct MusicalComposition MusicalComposition; |
|  | // Описание структуры MusicalComposition |
|  |  |
|  |  |
|  | // Создание структуры MusicalComposition |
|  | // функция создания элемента списка |
|  | MusicalComposition\* createMusicalComposition(char\* name, char\* author, int year) |
|  | { |
|  | MusicalComposition \*MusComp = (MusicalComposition\*)malloc(sizeof(MusicalComposition)); |
|  | MusComp->name = name; |
|  | MusComp->author = author; |
|  | MusComp->year = year; |
|  | MusComp->prev = NULL; |
|  | MusComp->next = NULL; |
|  |  |
|  | return MusComp; |
|  | } |
|  |  |
|  | // Функции для работы со списком MusicalComposition |
|  |  |
|  | MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n) |
|  | //создает список музыкальных коспозиций |
|  | { |
|  | MusicalComposition \*head = createMusicalComposition(array\_names[0], array\_authors[0], array\_years[0]); |
|  | MusicalComposition \*prev = head; |
|  | MusicalComposition \*curr; |
|  |  |
|  | for (int i = 1; i < n; ++i) |
|  | { |
|  | curr = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]); |
|  | curr->prev = prev; |
|  | prev->next = curr; |
|  | prev = curr; |
|  | } |
|  |  |
|  | return head; |
|  | } |
|  | void push(MusicalComposition\* head, MusicalComposition\* element) |
|  | // добавляет element в конец списка musical\_composition\_list |
|  | { |
|  | MusicalComposition\* temp = head; |
|  | while(temp->next != NULL) |
|  | temp = temp->next; |
|  | MusicalComposition\* new = createMusicalComposition(element->name, element->author, element->year); |
|  | temp->next = new; |
|  | new->prev = temp; |
|  | } |
|  |  |
|  | void removeEl(MusicalComposition\* head, char\* name\_for\_remove) |
|  | { |
|  | MusicalComposition\* ptr = head; |
|  | while (ptr->next != NULL) |
|  | { |
|  | if (strcmp(ptr->name, name\_for\_remove) == 0) |
|  | { |
|  | if (ptr->next != NULL && ptr->prev != NULL) |
|  | { |
|  | ptr->prev->next = ptr->next; |
|  | ptr->next->prev = ptr->prev; |
|  | } |
|  | else if (ptr->next == NULL) |
|  | { |
|  | ptr->prev->next = NULL; |
|  | } |
|  | else if (ptr->prev == NULL) |
|  | { |
|  | \*head = \*ptr->next; |
|  | head->prev = NULL; |
|  | } |
|  | } |
|  | ptr = ptr->next; |
|  | } |
|  | } |
|  |  |
|  | int count(MusicalComposition\* head) //возвращает количество элементов списка |
|  | { |
|  | MusicalComposition\* temp = head; |
|  | int count = 0; |
|  | while(temp != NULL) { |
|  | count++; |
|  | temp = temp->next; |
|  | } |
|  | return count; |
|  | } |
|  |  |
|  | void print\_names(MusicalComposition\* head) |
|  | //вывести названия композиций |
|  | { |
|  | MusicalComposition\* temp = head; |
|  | while(temp != NULL) { |
|  | printf("%s\n", temp->name); |
|  | temp = temp->next; |
|  | } |
|  | } |
|  |  |
|  | int main(){ |
|  | int length; //длина |
|  | 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); //выделение памяти под массивы |
|  |  |
|  | for (int i=0;i<length;i++) |
|  | { |
|  | char name[80]; |
|  | char author[80]; |
|  |  |
|  | fgets(name, 80, stdin); //считывание имени автора и года |
|  | fgets(author, 80, stdin); |
|  | fscanf(stdin, "%d\n", &years[i]); |
|  |  |
|  | (\*strstr(name,"\n"))=0; //функция стрстр-поиск первого вхождения подстроки в строку |
|  | (\*strstr(author,"\n"))=0; //находит символ подстроки в строки заменяет на ноль |
|  |  |
|  | names[i] = (char\*)malloc(sizeof(char\*) \* (strlen(name)+1)); |
|  | authors[i] = (char\*)malloc(sizeof(char\*) \* (strlen(author)+1)); |
|  |  |
|  | strcpy(names[i], name); //копируем name в names[i] |
|  | 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; |
|  |  |
|  | } |
|  |  |

1. Отправила в репозитрий Git

* Создана новая ветка <gorbunova\_lr4>
* В ней создана папка <gorbunovalab4>
* Исходные файлы добавлены в локальный репозиторий git командой

<git add>. Далее командой <git commit> были зафиксированы изменения в проекте

* Командой <git push> последний commit был отправлен на удаленный репозиторий

**Выводы.**

Данная работа позволила опробовать различный действия с линейными