**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

Тема: **Линейные списки**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8381 |  | Киреев К.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2019

**Цель работы**

Научиться работать с двунаправленными списками. Выполнить реализацию такого списка на языке СИ. Закрепить использование структур. Реализовать некоторые функции для работы со списками.

**Задание**

Создайте 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]**).

*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

**Выполнение работы**

1. Сначала описывается структура MusicalComposition.
2. Далее создана функция для создания элемента списка, в котором имеются данные о музыкальной композиции (название, исполнитель и год выхода) и указатели на предыдущий и следующий узел.
3. Создана функция для создания списка, где в переменную head помещаются данные первой композиции, а далее в цикле for добавляются все остальные элементы, посредством изменения указателей на узлы у переменной tmp, в которой хранится текущий последний узел, и переменной ptr, в которой хранится новый элемент.
4. Функция push отвечает за вставку нового элемента в конец списка. Сначала переменная head принимает значение последнего элемента. Далее новый элемент и head связываются с помощью указателей. Также имеется проверка на пустой список.
5. Функция removeEl удаляет элемент. Сначала происходит поиск элемента, который нужно удалить и затем указатели предыдущего и следующего элементов принимают значения указателей head.
6. Функция count подсчитывает количество элементов списка. Осуществляется проход по списку и увеличение счетчика.
7. Функция print\_names печатает названия композиций. Осуществляется проход по списку и вывод поля name элемента. Присутствует проверка на пустой список.
8. В функции main написана некоторая последовательность вызова команд для проверки работы списка.

**Выводы**

В ходе лабораторной работы были изучены правила создания двунаправленного списка на языке СИ. А также реализованы функции для работы со списком, такие как удаление и вставка элемента, подсчет количества элементов и вывод данных списка.

**Приложение А. Исходный код программы.**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

// Описание структуры MusicalComposition

typedef struct MusicalComposition{

char name[80];

char author[80];

int year;

struct MusicalComposition \*next, \*prev;

} MusicalComposition;

// Создание структуры MusicalComposition

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

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

strcpy(tmp->name, name);

strcpy(tmp->author, author);

tmp->year = year;

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

return tmp;

}

// Функции для работы со списком MusicalComposition

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

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

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

MusicalComposition \*ptr = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);

if (i == 1){

ptr->next = NULL;

ptr->prev = head;

head->next = ptr;

}

MusicalComposition \*tmp = head;

while(tmp->next)

tmp = tmp->next;

tmp->next = ptr;

ptr->prev = tmp;

ptr->next = NULL;

}

return head;

}

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

while (head->next)

head = head->next;

element->next = NULL;

element->prev = head;

head->next = element;

if (head == NULL)

head = element;

}

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

while(strcmp(head->name, name\_for\_remove))

head = head->next;

if (head->prev)

head->prev->next = head->next;

if (head->next)

head->next->prev = head->prev;

free(head);

}

int count(MusicalComposition\* head){

int n = 1;

while (head->next){

head = head->next;

n++;

}

return n;

}

void print\_names(MusicalComposition\* head){

if (!head){

printf("Empty\n");

return;

}

while(head){

puts(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);

for (int i=0;i<length;i++){

free(names[i]);

free(authors[i]);

}

free(names);

free(authors);

free(years);

return 0;

}