**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. 3341 |  | Самокрутов А.Р. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## Цель работы

Целью работы является освоение работы с линейными списками и их практическое применение. Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Ознакомиться со структурой данных «список».
2. Ознакомиться с функциями, используемыми для работы со списками.
3. Изучить способы реализации этих функций на языке программирования C.
4. Разработать программу, реализующую двусвязный линейный список и API для работы с ним.

## 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.

В функции main написана некоторая последовательность вызова команд для проверки работы вашего списка.

*Функцию main менять не нужно.*

## Выполнение работы

Создаётся структура данных *MusicalComposition* — узел двусвязного списка. Он содержит информацию о музыкальной композиции (строку *char\* name* — название композиции, строку *char\* author* — имя автора, целое число *int year* — год создания) и два указателя *struct MusicalComposition\** — на следующий и предыдущий узлы списка.

Далее описан API для работы со списком:

* 1. Функция для создания элемента списка *MusicalComposition\* createMusicalComposition(char\* name, char\* author, int year)*. Функция динамически выделяет память для элемента структуры *MusicalComposition*, после чего заполняет поля *name, author* и *year* переданными в функцию аргументами. Поля *next* и *prev* инициализируются значением *NULL*. Функция возвращает указатель на созданный узел.
  2. Функция *MusicalComposition\* createMusicalCompositionList(char\*\* array\_names, char\*\* array\_authors, int\* array\_years, int n)*. Функция с помощью *createMusicalComposition()* создаёт первый узел списка — *MusicalComposition\* head*. Далее функция с помощью цикла *for* (*n-1*) раз создаёт последующий узел *curr->next*, присваивает полю *prev* нового узла значение указателя на последний старый узел. Функция возвращает указатель на первый элемент списка.
  3. Функция *void push(MusicalComposition\* head, MusicalComposition\* element)*. С помощью цикла *while* функция доходит последнего элемента списка, после чего присваивает полю *next* значение указателя на элемент *MusicalComposition\* element*, который необходимо добавить. Полю *prev* этого элемента в свою очередь присваивается значение указателя на последний (после выполнения функции предпоследний) узел списка.
  4. Функция *removeEl(MusicalComposition\* head, char\* name\_for\_remove).* Функция проходит по узлам списка, пока не найдёт с помощью функции *strcmp()* композицию, чьё название совпадает со строкой *char\* name\_to\_remove*. После этого значение поля *next* предыдущей композиции заменяется на указатель на следующую, а значение поля *prev* следующей композиции — на указатель на предыдущую. Память, динамически выделенная для удалённого узла освобождается функцией *free()*.
  5. Функция *int* *count(MusicalComposition\* head)*. Счётчик *int* *cnt* инициализируется значением *0*, после чего функция циклом *while* проходит по всем узлам списка, инкрементируя счётчик на каждой итерации цикла. Функция возвращает значение счётчика *int* *cnt*.
  6. Функция *void* *print\_names(MusicalComposition\* head)*. Функция циклом *while* проходит по всем узлам списка, выводя на экран значение поля name текущего узла при каждой итерации.

Разработанный программный код см. в приложении А.

Результаты тестирования см. в приложении Б.

## Выводы

В ходе выполнения работы были изучены структура данных «список», операции, применяемые к этой структуре, способы реализации этих операций в языке C.

Разработана программа, реализующая двусвязный линейный список и API для работы с ним. В данном случае это набор функций, выполняющих следующие действия:

* 1. Создание элемента списка.
  2. Создание самого списка.
  3. Добавление элемента в конец списка.
  4. Удаления элемента с определённым значением.
  5. Подсчёт количества элементов списка.
  6. Вывод определённых значений элементов списка.

# Приложение А Исходный код программы

Название файла: main.c

#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\* autor,int year);

// Функции для работы со списком MusicalComposition

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

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);

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

free(names[i]);

free(authors[i]);

}

free(names);

free(authors);

free(years);

return 0;

}

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

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

comp->name = name;

comp->author = author;

comp->year = year;

comp->next = NULL;

comp->prev = NULL;

return comp;

}

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]);

MusicalComposition\* curr = head;

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

curr->next = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);

curr->next->prev = curr;

curr = curr->next;

}

return head;

}

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

MusicalComposition\* curr = head;

while (curr->next != NULL) {

curr = curr->next;

}

curr->next = element;

curr->next->prev = curr;

}

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

MusicalComposition\* curr = head;

while (strcmp(curr->name, name\_for\_remove) != 0 && curr != NULL) {

curr = curr->next;

}

if (strcmp(curr->name, name\_for\_remove) == 0) {

curr->prev->next = curr->next;

curr->next->prev = curr->prev;

free(curr);

}

}

int count(MusicalComposition\* head) {

int cnt = 0;

MusicalComposition\* curr = head;

while (curr != NULL) {

curr = curr->next;

cnt++;

}

return cnt;

}

void print\_names(MusicalComposition\* head) {

MusicalComposition\* curr = head;

while (curr != NULL) {

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

curr = curr->next;

}

}

# Приложение Б Тестирование

Таблица Б.1 - Примеры тестовых случаев

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | 7  Fields of Gold  Sting  1993  In the Army Now  Status Quo  1986  Mixed Emotions  The Rolling Stones  1989  Billie Jean  Michael Jackson  1983  Seek and Destroy  Metallica  1982  Wicked Game  Chris Isaak  1989  Points of Authority  Linkin Park  2000  Sonne  Rammstein  2001  Points of Authority | Fields of Gold Sting  1993  7  8  Fields of Gold  In the Army Now  Mixed Emotions  Billie Jean  Seek and Destroy  Wicked Game  Sonne  7 | Пример теста из задания. |
|  | 3  Rakbladsvalsen I  Hypothermia  2006  M/s salmonella  Lifelover  2006  Mountains Made Of Steam  Silver Mt. Zion  2012  So Special  FFF  2021  Mountains Made Of Steam | Rakbladsvalsen I Hypothermia 2006  3  4  Rakbladsvalsen I  M/s salmonella  So Special  3 | Удаление последнего элемента. |