**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. 8383 |  | Дейнега В.Е. |
| Преподаватель |  | Берленко Т. А. |

Санкт-Петербург

2019

**Цель работы.**

Реализовать двунаправленный линейный список и api (application programming interface - в данном случае набор функций) для работы с ним на языке программирования Си.

**Основные теоретические положения.**

Объявление структуры

Синтаксис объявления структуры

struct <имя> {

<тип1> <поле1>;

<тип2> <поле2>;

...

<типN> <полеN>;

};

Например

struct point\_t {

int x;

int y;

};

Полями структуры могут быть любые объявленные типы, кроме самой структуры этого же типа, но можно хранить указатель на структуру этого типа:

struct node {

void\* value;

struct node next;

};

Нельзя, нужно

struct node {

void\* value;

struct node \*next;

};

В том случае, если несколько полей имеют один тип, то их можно перечислить через запятую:

struct Point3D {

int x, y, z;

};

После того, как мы объявили структуру, можно создавать переменную такого типа с использованием служебного слова struct. Доступ до полей структуры осуществляется с помощью операции точка

Структура, объявленная в глобальном контексте, видна всем. Структура также может быть объявлена внутри функции

Структуру можно инициализировать во время создания как массив. Поля в этом случае будут присваиваться по порядку.

struct gasket {

float weight;

unsigned height;

unsigned diameter;

} gasket obj = { 12.f, 120, 30 };

Указатель на структуру создаётся как обычно. Отличие заключается в том, что можно обращаться к полям структуры через указатель с помощью операции "стрелка" (минус + больше).

## Линейный однонаправленный список

Список - некоторый упорядоченный набор элементов любой природы.

Линейный однонаправленный (односвязный) список - список, каждый элемент которого хранит помимо значения указатель на следующий элемент. В последнем элементе указатель на следующий элемент равен NULL (константа нулевого указателя).

## Линейный двусвязный список

-базовая динамическая структура данных в информатике, состоящая из [узлов](https://ru.wikipedia.org/wiki/%D0%A3%D0%B7%D0%B5%D0%BB_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), каждый из которых содержит как собственно [данные](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), так и одну или две [ссылки](https://ru.wikipedia.org/wiki/%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) на следующий и/или предыдущий узел списка. Принципиальным преимуществом перед [массивом](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) является структурная гибкость: порядок элементов связного списка может не совпадать с порядком расположения элементов данных в памяти компьютера.

Каждый узел двунаправленного (двусвязного) линейного списка (ДЛС) содержит два поля указателей — на следующий и на предыдущий узлы. Указатель на предыдущий узел корня списка содержит нулевое значение. Указатель на следующий узел последнего узла также содержит нулевое значение.

**Реализация:**

1. Описана структура MusicalComposition, представляющая собой элемент списка, содержащая 2 строки и 2 ссылки (на следующий элемент списка и не предыдущий).

2.Описаны следующие функции:

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

Функция выделяет память под новый элемент списка, заполняет поля данными и возвращает указатель на созданный элемент.

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

Функция создает двусвязный список из элементов, используя вышеописанную функцию, возвращает ссылку на голову списка.

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

Функция добавляет элемент в конец списка.

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

Функция ищет элемент с заданным именем, а затем удаляет его из списка.

2.5 int count(MusicalComposition\* head);

Функция проходится по всему списку и возвращает количество элементов в нем.

2.6 void print\_names(MusicalComposition\* head);

Функция печатает поля name каждого элемента списка.

3. Функция main дана по условию.

**Тестирование программы:**

**Ввод:**

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

**Вывод:**

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

**Ввод**:

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

Billie Jean

**Вывод:**

Fields of Gold Sting 1993

7

8

Fields of Gold

In the Army Now

Mixed Emotions

Seek and Destroy

Wicked Game

Points of Authority

Sonne

7

**Выводы.**

В ходе выполнения задания лабораторной работы были изучены линейный и двусвязный список, а также их реализация на языке программирования си.

Приложение

Код программы: #include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct MusicalComposition{

char name[80];

char author[80];

int year;

struct MusicalComposition\* prev;

struct MusicalComposition\* next;

}MusicalComposition;

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

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

strcpy(newComp->name, name);

strcpy(newComp->author, author);

newComp->year = year;

newComp->prev = NULL;

newComp->next = NULL;

return (newComp);

}

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\* tmp = createMusicalComposition(array\_names[1], array\_authors[1], array\_years[1]);

head->next = tmp;

tmp->prev = head;

for (int i=2; i<n; i++){

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

tmp->next->prev = tmp;

tmp = tmp->next;

}

return head;

}

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

MusicalComposition\* elem = head;

while(elem->next != NULL){

elem = elem->next;

}

elem->next = element;

elem->next->prev = elem;

}

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

MusicalComposition\* elem = head;

while(strcmp(elem->name, name\_for\_remove)){

elem = elem->next;

}

elem->next->prev = elem->prev;

elem->prev->next = elem->next;

}

int count(MusicalComposition\* head){

MusicalComposition\* elem = head;

int i = 1;

while(elem->next != NULL){

elem = elem->next;

i++;

}

return i;

}

void print\_names(MusicalComposition\* head){

MusicalComposition\* elem = head;

while(elem->next != NULL){

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

elem = elem->next;

}

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

}

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;

}