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: Структуры данных, линейные списки.

Студентка гр. 1304 Нго Тхи Йен.

Преподаватель Чайка К.В.

Санкт-Петербург

2022

Цель работы.

Исучить основные структуры данных, научиться реализовывать линейные спички в программах на языке программирования Си.

Задание.

Создайте двунаправленный список музыкальных композиций MusicalComposition и api (application programming interface - в данном случае набор функций) для работы со списком.

Структура элемента списка (тип - MusicalComposition):

- name - строка неизвестной длины (гарантируется, что длина неможет 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 первомуэлементу списка array_names (array_names[0]).
  - Поле author первого элемента списка соответствует первомуэлементу списка array_authors (array_authors[0]).

- Поле year первого элемента списка соответствует первомуэлементу 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 названиякомпозиций.

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

Функцию main менять не нужно.

Выполнение работы.

Структура MusicalComposition:

Структура состоит из 5-ти полей: name – строка, длина которой не превышает 80-ти символов, содержит в себе название музыкальной композиции; author – строка, длина которой не превышает 80-ти символов, содержит в себе имя авторамузыкальной композиции; year – целое число, содержит в себе год создания композиции; next – указатель на структуру MusicalComposition, содержит в себе указатель на следующий элемент списка; previous – указатель на структуру MusicalComposition, содержит в себе указатель на предыдущий элемент списка; Функция createMusicalComposition:

Функция принимает на вход указатели на строки name и author, а также целое число year, возвращает указатель на созданную структуру. Выполнение функции начинается с выделения памяти под хранение структуры. В соответствующие поля созданой структуры копируются строки name и author, в поле year записывается содержимое одноименной переменной, поля next и previous заполняются нулевыми указателями.

Функция createMusicalCompositionList:

Функция принимает на вход 3 массива: массивы array_names и array_authors содержат в себе указатели на строки, массив array_years содержит целые числа, также функция принимает число n – число элементов в каждом массиве. Функция возвращает указатель на первый жлемент созданного списка.

Функция начинает работу с объявления 3-х указателей на структуры MusicalComposition: current, previous и out. В переменную out записывается указатель на первый элемент списка, в previous запиывается значение out. Далее запускается цикл, счетчик i которого принимает значения от 1 до n. В переменную current записывается указатей на i-ый элемент списка, в поле next структуры, находящейся по указателю previous, записывается значение указателя current. В поле previous структуры, находящейся по указателю current, записывается значение указателя previous. Выполнение итерации цикла заканчивается записью значения указателя current в переменную previous. Функция возвращает значение указателя out.

Функция push:

Функция принимает на вход указатель на первый элемент списка head и указатель на элемент element, который необходимо добавить в список. С помощью цикла while происходит нахождение последнего элемента списка. В поле next последнего элемента списка записывается указатель на элемент element, в поле previous структуры по указателю element записывается указатель на последний элемент списка.

Функция removeEl:

Функция принимает на вход указатель на первый элемент списка head, а также указатель на строку name_for_remove, в которой хранится название элемента, который необходимо удалить. В цикле while перебираются все элементы списка, пока поле next текущего элемента не равно нулевому указателю. Если поле name следующего элемента совпадает со строкой, записаной по указателю name_for_remove, то в переменной tmp сохраняется указатель из поля next следующего элемента, память, выделенная под хранение следующего элемента освобождается, в поле next текущего элемента записывается указатель из переменной next, в пооле previous следующего элемента записывается указатель на текущий элемент. Если значение поля name следующего элемента не совпадает со строкой, записаной по указателю name_for_remove, то происходит переход к следующему элементу.

Функция count:

Функция принимает на вход указатель на первый элемент списка head, функция возвращает количество элементов списка.

В цикле внутри функции происходит перебор всех элементов списка, с каждой итерацией цикла увеличивается значение счетчика, функция возвращает знаяение счетчика.

Функция print_names:

Функция принимает на вход указатель на первый элемент списка head.

В цикле внутри функции происходит перебор всех элементов списка, с каждой итерацией цикла происходит вывод в консоль поля name текущего элемента списка.

Выводы.

В ходе выполнения данной лабораторной раобты были изучены основные структуры данных, была создана реализация двунаправленного списка, а также функции для работы с данным списком.

ПРИЛОЖЕНИЕ БТЕСТИРОВАНИЕ

Данные тестирования представлены в таблице 1.

Таблица 1 – Результаты тестирования.

№ п/п	Входные данные	Выходные данные	Комментарии
1.	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	Результат корректен
2.	777 Joji 2020 Besidju shamana 2016 Down Hashir 2019 Tone Deaf Eminem	777 Joji 2020 6 7 777 Down Tone Deaf No Chances Bullet In A Gun 5	Результат корректен

	2020 Besidju shamana 2016 No Chances Twenty One Pilots 2021 Bullet In A Gun Imagine Dragons 2018 Besidju		
3.	5 Blessing It Nujabes 2003 Dream On Aerosmith 1973 Killer Queen Queen 1974 My Name Is Eminem 1999 By Myself Linkin Park 2000 Riders on the Storm The Doors 1971 Riders on the Storm	Blessing It Nujabes 2003 5 6 Blessing It Dream On Killer Queen My Name Is By Myself 5	Результат корректен

Ход работы

Название файла main.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>


typedef struct MusicalComposition{

   char* name;
   char* author;
   int year;
   struct MusicalComposition* next;
   struct MusicalComposition* prev;
}MusicalComposition;



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

   MusicalComposition* cmp  = malloc(sizeof(struct MusicalComposition));
   cmp->name = name;
   cmp->author = author;
   cmp->year = year;
   cmp->next = NULL;
   cmp->prev = NULL;
   return cmp;
}


MusicalComposition* createMusicalCompositionList(char** array_names, char** array_authors,
int* array_years, int n){

   int i;
   MusicalComposition* cmp = createMusicalComposition(array_names[0], array_authors[0],
array_years[0]);
   MusicalComposition* now;
   MusicalComposition* tmp = cmp;
   for(i = 1; i < n; i++){
      now = createMusicalComposition(array_names[i], array_authors[i], array_years[i]);
      now->prev = tmp;
      tmp->next = now;
      tmp = now;
   }
   return cmp;
}
```

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

   while (head->next != NULL){
      head = head->next;
   }
   head->next = element;
   element->prev = head;
}

void removeEl(MusicalComposition* head, char* name_for_remove){

   while(head != NULL){
      if (strcmp(head->name, name_for_remove) == 0){
         head->prev->next = head->next;
         head->next->prev = head->prev;
         free(head);
      }
      head = head->next;
   }

}

int count(MusicalComposition* head){

   int n = 0;
   while (head != NULL){
      head = head->next;
      n++;
   }
   return n;
}

void print_names(MusicalComposition* head){

   while (head != NULL){
      printf("%s\n", head->name);
      head = head->next;
   }
}


int main(){
   int length;
   char c;
   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", &years[i]);
        fscanf(stdin, "%c", &c);//
        (*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", &year_for_push);
    fscanf(stdin, "%c", &c);
    (*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;

}
```