**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

Курсовая РАБОТА

**по дисциплине «Программирование»**

Тема: Линейные связанные списки

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Малышенко Ю.И. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

**ЗАДАНИЕ**

**на курсовую работу**

|  |
| --- |
| Студент Малышенко Ю.И. |
| Группа 6303 |
| Тема работы: Линейные списки |
|  |
| Содержание пояснительной записки:   * Содержание * Введение * Описание функций, необходимых для работы с двунаправленным линейным списком * Описание функций удаления элементов списка * Примеры работы программы * Заключение * Список использованных источников * Приложение А. Исходный код программы |
| Предполагаемый объем пояснительной записки:  Не менее 10 страниц.  Дата выдачи задания: 02.12.2016  Дата сдачи реферата: 23.12.2016  Дата защиты реферата: 23.12.2016   |  |  |  | | --- | --- | --- | | Студент гр. 6303 |  | Малышенко Ю.И. | | Преподаватель |  | Берленко Т.А. | |

**Аннотация**

В данной работе была разработана программа на языке программирования C, которая позволяет работать с набором функций, отвечающих засписок музыкальных композиций. Для функционирования списка были созданы и описаны необходимые функции, позволяющие добавлять, удалять(удаление элемента (или элементов) из списка с помощью подстроки) и выводить элементы списка и их количество в консоль, а также описана структура элемента списка.

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 5 |
| 1. | Описание функций, необходимых для работы с двунаправленным линейным списком | 5 |
| 1.1. | Созданиеэлемента | 5 |
| 1.2. | Создание двунаправленного списка связанных элементов | 6 |
| 1.3. | Добавление элемента в конец списка | 7 |
| 1.4. | Удаление элементов из списка | 7 |
| 1.5. | Подсчет количества элементов в списке | 9 |
| 1.6. | Вывод элементов списка в консоль | 10 |
| 1.7. | Примеры работы программы | 11 |
| 1.8. | Список использованных источников | 13 |
| 1.9. | Вывод | 13 |
| 2. | Исходный код программы | 14 |

**Введение**

Необходимо создать линейный связанный список музыкальных композиций и интерфейс для работы с ним, в этой работе это набор функций.  
Элемент списка представляет собой структуру вида struct MusicalComposition,  
Которая хранит данные о названии музыкального произведения (name), об авторе (author), о годе выпуска (year) и об указателях на следующий (next) и предыдущий(prev) элемент списка.

**Цель работы:**

Научиться использовать линейные списки, создать **api (application programming interface)** для работы со списком.

**Необходимые функции для работы со списком:**

* Создание элемента типа MusicalComposition
* Создание двунаправленного списка связанных элементов типа MusicalComposition
* Добавление элемента в конец списка
* Удаление определенного элемента (или элементов) из списка
* Удаление элемента (или элементов) из списка с помощью подстроки
* Подсчет количества элементов в списке
* Вывод элементов списка в консоль

**Описание функций программы**

* Функция CreateMusicalComposition создаёт элемент структуры и сохраняет в него данные пользователя.

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

*// функция создания элемента списка*

{

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

MusComp->name = name;

MusComp->author = author;

MusComp->year = year;

MusComp->status = 1;

MusComp->prev = NULL;

MusComp->next = NULL;

return MusComp;

}

* Функция CreateMusicalCompositionList вызывает функцию CreateMusicalComposition для каждой композиции, которая подаётся на входе, а затем связывает элементы между собой, формируя связанный список.

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

*// Функции для работы со списком MusicalComposition*

*{*

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

MusicalComposition \*prev = head;

MusicalComposition \*curr; //текущая

int i;

for (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;

}

* Функция push позволяет добавлять новый элемент в конец связанного списка

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

*// добавляет element в конец списка musical\_composition\_list*

{

MusicalComposition\* temp = head;

MusicalComposition\* new = createMusicalComposition(element->name, element->author, element->year);

if (head-> status == 0)

\*head = \*new;

else

{

while(temp->next != NULL)

temp = temp->next;

temp->next = new;

new->prev = temp;

}

}

* Функция removeEl позволяет найти и удалить любой элемент по его имени и удалить его из списка.

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

*// удаляет элемент element списка, у которого значение name равно значению name\_for\_remove*

{

MusicalComposition\* temp = head;

while(temp->next)

if (strcmp(temp->name, name\_for\_remove) == 0*)*

*//strcmp сравнивает name и name\_for\_remove*

{

if(temp->next != NULL && temp->prev != NULL)

{

temp->prev->next = temp->next;

temp->next->prev = temp->prev;

temp = temp->next;

}

else if(temp->prev == NULL)

{

\*temp = \*(temp->next);

temp->prev = NULL;

}

}

else

temp = temp->next;

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

{

if(temp->prev == NULL)

head->status = 0;

else

temp->prev->next = NULL;

}

}

* Функция removeElPOD позволяет найти и удалить любой элемент по его подстрокой в имени и удалить его из списка.

void removeElPOD(MusicalComposition\* head, char\* name\_for\_removePOD)

{

MusicalComposition\* temp = head;

while(temp->next)

if (strstr(temp->name, name\_for\_removePOD) != NULL)

{

if(temp->next != NULL && temp->prev != NULL)

{

temp->prev->next = temp->next;

temp->next->prev = temp->prev;

temp = temp->next;

}

else if(temp->prev == NULL)

{

\*temp = \*(temp->next);

temp->prev = NULL;

}

}

else

temp = temp->next;

if (strstr(temp->name, name\_for\_removePOD) != NULL)

{

if(temp->prev == NULL)

head->status = 0;

else

temp->prev->next = NULL;

}

}

* Функция count считает количество элементов, которые находятся в списке

int count(MusicalComposition\* head)

*//считает количество структур в списке*

*{*

MusicalComposition\* temp = head;

int count = 0;

while(temp)

if (temp->status == 0)

return 0;

else

{

count++;

temp = temp->next;

}

return count;

}

* Функция print\_names выводит в консоль текущее состояние списка

void print\_names(MusicalComposition\* head)

*//вывести названия композиций*

{

MusicalComposition\* temp = head;

if (temp->status == 1)

while(temp)

{

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

temp = temp->next;

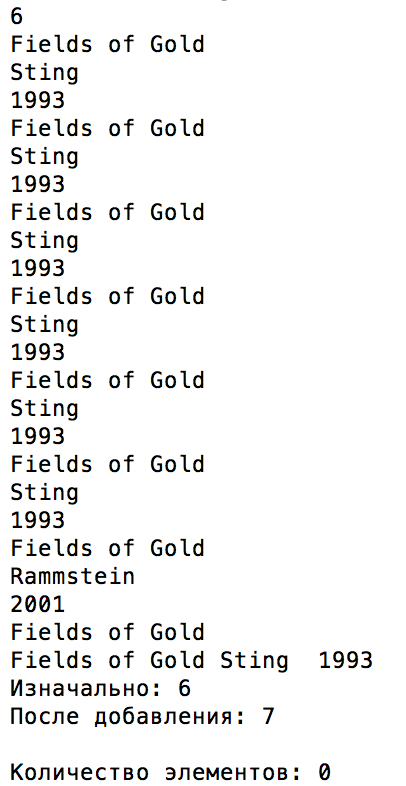
}

}

**Примеры работы программы:**

**Удаление элемента с середины списка**  
****

**Удаление всех элементов из списка**

****

**Удаление всех элементов с помощью подстроки:**

****

**список использованных источников**

1. Язык программирования СИ / Керниган Б., Ритчи Д. СПб.: Издательство "Невский Диалект", 2001. 352 с.
2. UNIX.  Программное окружение / Керниган Б., Пайк Р. СПб.: Символ Плюс, 2003. 416 с.
3. "Язык программирования C. Лекции и упражнения", Стивен Прата

**Вывод :** В ходе курсовой работы была написана работающая программа, которая полностью решает поставленную задачу.

**Приложение А.**

**Полный код программы:**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

struct MusicalComposition

{

char \*name;

char \*author;

int year;

int status;

struct MusicalComposition \*next, \*prev;

};

typedef struct MusicalComposition MusicalComposition;

*// Описание структуры MusicalComposition*

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

*// функция создания элемента списка*

*{*

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

MusComp->name = name;

MusComp->author = author;

MusComp->year = year;

MusComp->status = 1;

MusComp->prev = NULL;

MusComp->next = NULL;

return MusComp;

}

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

*// Функции для работы со списком MusicalComposition*

{

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

MusicalComposition \*prev = head;

MusicalComposition \*curr; *//текущая*

int i;

for (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;

MusicalComposition\* new = createMusicalComposition(element->name, element->author, element->year);

if (head-> status == 0)

\*head = \*new;

else

{

while(temp->next != NULL)

temp = temp->next;

temp->next = new;

new->prev = temp;

}

}

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

*// удаляет элемент element списка, у которого значение name равно значению name\_for\_remove*

{

MusicalComposition\* temp = head;

while(temp->next)

if (strcmp(temp->name, name\_for\_remove) == 0)*//strcmp сравнивает name и name\_for\_remove*

{

if(temp->next != NULL && temp->prev != NULL)

{

temp->prev->next = temp->next;

temp->next->prev = temp->prev;

temp = temp->next;

}

else if(temp->prev == NULL)

{

\*temp = \*(temp->next);

temp->prev = NULL;

}

}

else

temp = temp->next;

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

{

if(temp->prev == NULL)

head->status = 0;

else

temp->prev->next = NULL;

}

}

void removeElPOD(MusicalComposition\* head, char\* name\_for\_removePOD)

*// удаляет элемент element списка, у которого значение name равно значению name\_for\_remove*

{

MusicalComposition\* temp = head;

while(temp->next)

if (strstr(temp->name, name\_for\_removePOD) != NULL)

{

if(temp->next != NULL && temp->prev != NULL)

{

temp->prev->next = temp->next;

temp->next->prev = temp->prev;

temp = temp->next;

}

else if(temp->prev == NULL)

{

\*temp = \*(temp->next);

temp->prev = NULL;

}

}

else

temp = temp->next;

if (strstr(temp->name, name\_for\_removePOD) != NULL)

{

if(temp->prev == NULL)

head->status = 0;

else

temp->prev->next = NULL;

}

}

int count(MusicalComposition\* head) *//считает количество структур в списке*

{

MusicalComposition\* temp = head;

int count = 0;

while(temp)

if (temp->status == 0)

return 0;

else

{

count++;

temp = temp->next;

}

return count;

}

void print\_names(MusicalComposition\* head)

*//вывести названия композиций*

{

MusicalComposition\* temp = head;

if (temp->status == 1)

while(temp)

{

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

temp = temp->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);

int i;

for (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];

char name\_for\_removePOD[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("Изначально: %d\n", k);

push(head, element\_for\_push);

k = count(head);

printf("После добавления: %d\n", k);

removeEl(head, name\_for\_remove);

print\_names(head);

fgets(name\_for\_removePOD, 80, stdin);

(\*strstr(name\_for\_removePOD,"\n"))=0;

removeElPOD(head, name\_for\_removePOD);

print\_names(head);

k = count(head);

printf("Количество после добавления: %d\n", k);

char name\_for\_push2[80];

char author\_for\_push2[80];

int year\_for\_push2;

fgets(name\_for\_push2, 80, stdin);

fgets(author\_for\_push2, 80, stdin);

fscanf(stdin, "%d\n", &year\_for\_push2);

(\*strstr(name\_for\_push2,"\n"))=0;

(\*strstr(author\_for\_push2,"\n"))=0;

MusicalComposition\* element\_for\_push2 = createMusicalComposition(name\_for\_push2, author\_for\_push2, year\_for\_push2);

push(head, element\_for\_push2);

print\_names(head);

k = count(head);

printf("Количество после добавления: %d\n", k);

char name\_for\_push3[80];

char author\_for\_push3[80];

int year\_for\_push3;

fgets(name\_for\_push3, 80, stdin);

fgets(author\_for\_push3, 80, stdin);

fscanf(stdin, "%d\n", &year\_for\_push3);

(\*strstr(name\_for\_push3,"\n"))=0;

(\*strstr(author\_for\_push3,"\n"))=0;

MusicalComposition\* element\_for\_push3 = createMusicalComposition(name\_for\_push3, author\_for\_push3, year\_for\_push3);

push(head, element\_for\_push3);

print\_names(head);

k = count(head);

printf("Количество после добавления: %d\n", k);

free (names);

free(authors);

free(years);

return 0;

}