**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

**КУРСОВАЯ РАБОТА**

**по дисциплине «Программирование»**

Тема: Линейные списки

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6303 |  | Горбунова А.П. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студентка Горбунова А.П. | | |
| Группа 6303 | | |
| Тема работы: Линейные списки | | |
|  | | |
| Содержание пояснительной записки:   * Содержание * Введение * Описание функций, необходимых для работы с двунаправленным линейным списком * Описание функций поиска подстроки 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 | | |
| Студентка |  | Горбунова А.П. |
| 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 и выводить элементы списка и их количество в консоль, а также описана структура элемента списка.

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 4 |
| 1. | Функции, необходимые для работы с двунаправленным линейным списком | 4 |
| 1.1. | Функция для создания элемента списка | 5 |
| 1.2. | Функция для создания списка | 5 |
| 1.3. | Функция для добавления элемента в конец списка | 6 |
| 1.4. | Функция для добавления элемента в начало списка | 6 |
| 1.5. | Функция для удаления элемента списка | 7 |
| 1.6.  1.7 | Функция для возвращения количества элементов списка  Функция для вывода названий композиций | 7  8 |
| 2. | Исходный код программы | 8 |
| 3. | Примеры работы программы | 13 |
|  | Список использованных источников | 14 |
|  | Заключение | 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.

1. Необходимые функции для работы со списком:

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

**Описание функций программы**

* 1. Функция для создания элемента списка

(тип элемента MusicalComposition)

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

{

MusicalComposition \*MusComp =(MusicalComposition\*)malloc(sizeof(MusicalComposition));  
MusComp->name = name;  
MusComp->author = author;  
MusComp->year = year;  
MusComp->prev = NULL;  
MusComp->next = NULL;  
return MusComp;

}

* 1. Функция создания списка музыкальных композиций

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

В списке:

* 1. ***n****- длина массивов****array\_names****,****array\_authors****,****array\_years****.*
  2. поле **name** первого элемента списка соответствует первому элементу списка array\_names (**array\_[0]**).
  3. поле **author** первого элемента списка соответствует первому элементу списка array\_authors (**array\_authors[0]**).
  4. 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.*

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 \*prev = head;  
MusicalComposition \*curr;  
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;  
}

* 1. Функция для добавления элемента в конец списка

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

{

MusicalComposition\* temp = head;

while(temp->next != NULL)

temp = temp->next;

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

temp->next = new;

new->prev = temp;

}

* 1. Функция для добавления элемента в начало списка

void push (MusicalComposition\*\* head, MusicalComposition\* element1, MusicalComposition\* element2, MusicalComposition\* element3)

{

MusicalComposition\* temp =\* head;

MusicalComposition\* new1 = createMusicalComposition(element1->name, element1-

>author, element1->year);  
temp->prev = new1;  
new1->next = temp;  
\*head = new1;

MusicalComposition\* new2 = createMusicalComposition(element2->name, element2->author,   
element2->year);  
new->next=new2;  
new2->prev=new;

new2->next=temp;  
temp->prev=new2;

MusicalComposition\* new3 = createMusicalComposition(element3->name, element3->author, element3->year);  
new2->next=new3;  
new3->prev=new2;  
new3->next=temp;  
temp->prev=new3;  
}

1.5 Функция для удаления элемента element списка, у которого значение name равно значению  name\_for\_remove

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

{   
MusicalComposition\* ptr = head;   
while (ptr->next != NULL)

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

{   
if (ptr->next != NULL && ptr->prev != NULL)

{   
ptr->prev->next = ptr->next;   
ptr->next->prev = ptr->prev;   
}   
else if (ptr->next == NULL)   
{   
ptr->prev->next = NULL;   
}

else if (ptr->prev == NULL)   
{   
\*head = \*ptr->next;   
head->prev = NULL;   
}

}   
ptr = ptr->next;   
}

}

1.6 Функция для возвращения количества элементов списка

int count(MusicalComposition\* head)

{  
MusicalComposition\* temp = head;  
int count = 0;  
while(temp != NULL)

{

count++;  
temp = temp->next;  
}

return count;  
}

1.7 Функция для вывода названий композиций

void print\_names(MusicalComposition\* head)

{  
MusicalComposition\* temp = head;  
while(temp != NULL)

{  
printf("%s\n", temp->name);  
temp = temp->next;  
}

}

1. **Исходный код программы.**

#include <stdlib.h>  
#include <stdio.h>  
#include <string.h>  
  
struct MusicalComposition

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

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

{  
MusicalComposition \*MusComp =(MusicalComposition\*)malloc(sizeof(MusicalComposition));  
MusComp->name = name;  
MusComp->author = author;  
MusComp->year = year;  
MusComp->prev = NULL;  
MusComp->next = NULL;  
return MusComp;  
}

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 \*prev = head;  
MusicalComposition \*curr;  
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 pushend(MusicalComposition\* head, MusicalComposition\* element)

{

MusicalComposition\* temp = head;

while(temp->next != NULL)

temp = temp->next;

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

temp->next = new;

new->prev = temp;

}

void push (MusicalComposition\*\* head, MusicalComposition\* element1, MusicalComposition\* element2, MusicalComposition\* element3)

{  
MusicalComposition\* temp =\* head;  
MusicalComposition\* new1 = createMusicalComposition(element1->name, element1->author, element1->year);  
temp->prev = new1;  
new1->next = temp;  
\*head = new1;

MusicalComposition\* new2 = createMusicalComposition(element2->name, element2->author, element2->year);  
new1->next=new2;  
new2->prev=new1;

new2->next=temp;  
temp->prev=new2;

MusicalComposition\* new3 = createMusicalComposition(element3->name, element3->author, element3->year);  
new2->next=new3;  
new3->prev=new2;  
new3->next=temp;  
temp->prev=new3;  
}

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

{   
MusicalComposition\* ptr = head;   
while (ptr->next != NULL)

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

{   
if (ptr->next != NULL && ptr->prev != NULL)

{   
ptr->prev->next = ptr->next;   
ptr->next->prev = ptr->prev;   
}   
else if (ptr->next == NULL)   
{   
ptr->prev->next = NULL;   
}

else if (ptr->prev == NULL)   
{   
\*head = \*ptr->next;   
head->prev = NULL;   
}

}   
ptr = ptr->next;   
}

}

int count(MusicalComposition\* head)   
{

MusicalComposition\* temp = head;  
int count = 0;  
while(temp != NULL)

{

count++;  
temp = temp->next;  
}

return count;  
}  
  
void print\_names(MusicalComposition\* head)  
{

MusicalComposition\* temp = head;  
while(temp != NULL)

{  
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\_push1[80];

char author\_for\_push1[80];

int year\_for\_push1;  
fgets(name\_for\_push1, 80, stdin);  
fgets(author\_for\_push1, 80, stdin);  
fscanf(stdin, "%d\n", &year\_for\_push1);  
(\*strstr(name\_for\_push1,"\n"))=0;  
(\*strstr(author\_for\_push1,"\n"))=0;

MusicalComposition\* element\_for\_push1 = createMusicalComposition(name\_for\_push1, author\_for\_push1, year\_for\_push1);

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);

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);

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;

int k = count(head);  
printf("Before: \n %d\n", k);

removeEl(head, name\_for\_remove);

pushend(head, element\_for\_push);

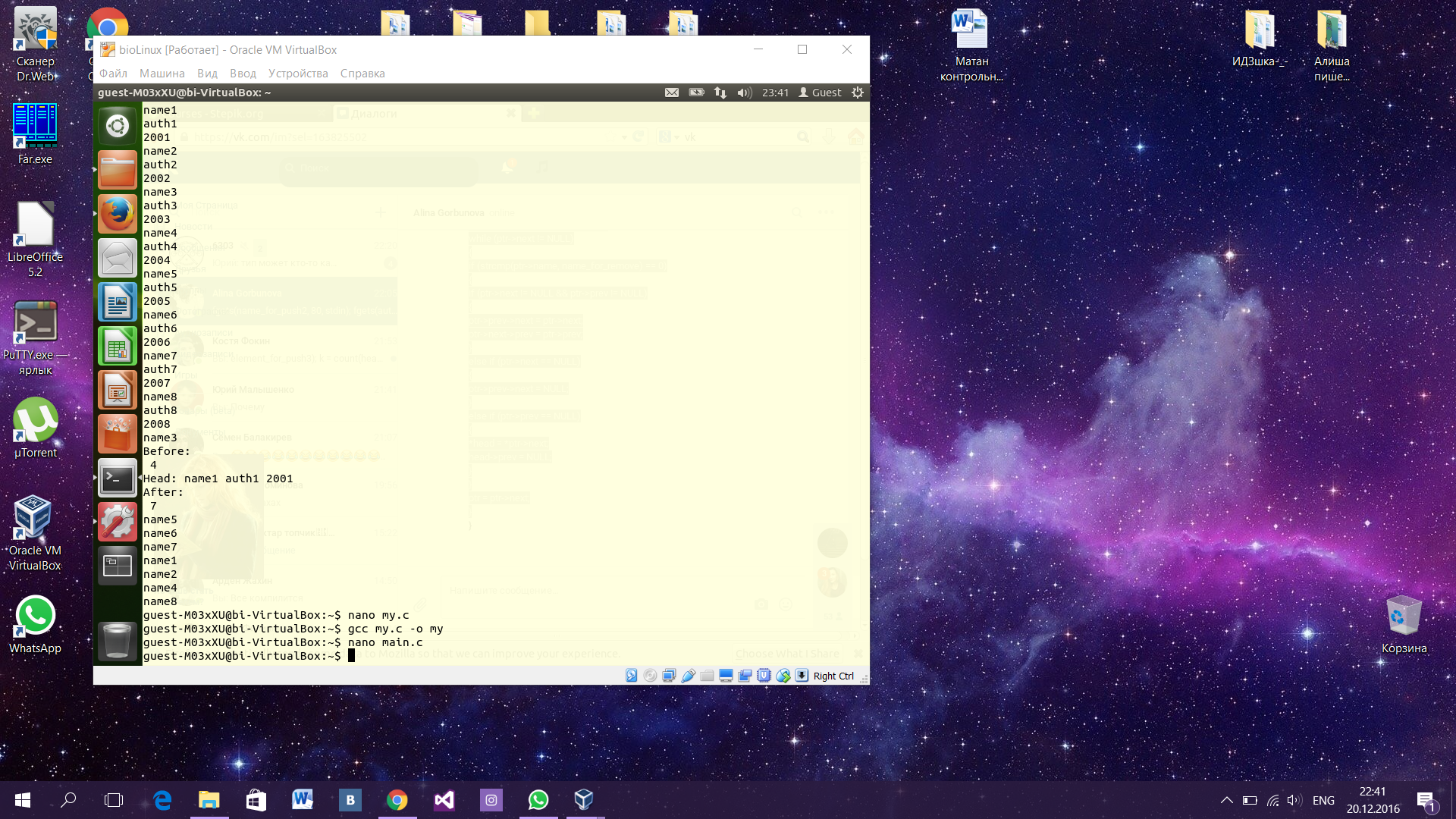
push(&head, element\_for\_push1, element\_for\_push2, element\_for\_push3);

k = count(head);  
printf("After: \n %d\n", k);

printf("Head: %s %s %d\n", head->name, head->author, head->year);

print\_names(head);  
  
free(names);  
free(authors);  
free(years);  
return 0;  
  
}

1. **Примеры работы программы**



**список использованных источников**

1. Язык программирования СИ / Керниган Б., Ритчи Д. СПб.: Издательство "Невский Диалект", 2001. 352 с.

2. <http://cppstudio.com/> (дата обращения:17.12.2016)

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