**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №4**

**по дисциплине «Программирование»**

Тема: *Линейные списки*

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Филиппенко Д.Р. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2016

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

Ход работы:

1. Описали структуру MusicalComposition и создали структуру MusicalComposition:

struct MusicalComposition

{

char\* name;

char\* author;

int year;

struct MusicalComposition \*prev, \*next;

};

typedef struct MusicalComposition MusicalComposition;

1. Создали функцию для работы со списком 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. Создали функцию для создания списка музыкальных композиций MusicalCompositionList:

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 \*actual;

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

{

actual = createMusicalComposition(array\_names[i], array\_authors[i], array\_years[i]);

actual->prev = prev;

prev->next = actual;

prev = actual;

}

return head;

}

1. Также создали некоторые функции для работы со списком:

4.1) Функция void push добавляет элемент в конец списка:

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

{

MusicalComposition\* actual = head;

while(actual->next)

actual = actual->next;

actual->next = element;

element->prev = actual;

}

4.2) Функция removeEl удаляет элемент по заданному имени:

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

{

MusicalComposition\* actual = head;

while(actual->next != NULL)

{

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

{

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

{

actual->prev->next = actual->next;

actual->next->prev = actual->prev;

}

else if(actual->next == NULL)

{

actual->prev->next = NULL;

}

else if(actual->prev == NULL)

{

actual->prev = NULL;

\*head = \*actual;

}

}

actual = actual->next;

}

}

4.3) Функция int count возвращает количество элементов списка:

int count(MusicalComposition\* head)

{

MusicalComposition\* actual = head;

int count = 0;

while(actual->next)

{

++count;

actual = actual->next;

}

++count;

return count;

}

4.4) Функция print\_names выводит названия композиций:

void print\_names(MusicalComposition\* head)

{

MusicalComposition\* actual = head;

while(actual)

{

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

actual = actual->next;

}

}

1. Также по условию была дана функция main, в которой была написана последовательность команд, проверяющих работу нашего списка:

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);

return 0;

}

Вывод: в ходе выполнения лабораторной работы получены навыки работы с линейными списками и структурами в языке Си. При помощи полученных знаний успешно создана программа для работы с двусвязными списками.