МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине “Программирование”

Тема: Структуры данных, линейные списки

Студент гр. 7381 _____ Ильясов А. В.
Преподаватель _____ Берленко Т. А.

Санкт-Петербург
2017

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Ильясов А. В.

Группа 7381

Тема работы: Структуры данных, линейные списки

Исходные данные:

Создать двунаправленный список музыкальных композиций MusicalComposition и api (application programming interface – в данном случае набор функций) для работы со списком. Создать функцию которая удаляет все чётные элементы списка.

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 23.11.2017

Дата сдачи реферата:

Дата защиты реферата:

Студент гр. 7381 _____ Ильясов А. В.

Преподаватель _____ Берленко Т. А.

АННОТАЦИЯ

В результате выполнения курсовой работы был создан двунаправленный линейный список. Созданы функции для работы с данным списком, а именно: добавление в конец нового элемента списка, удаление элемента списка, подсчёт количества элементов списка, вывода названия элементов списка, функция удаления чётных элементов списка. Создан удобный интерфейс для работы с программой.

SUMMARY

As a result of performing the course work was created by a bidirectional linear list. Created functions for working with data list, namely: add to end of new list item, delete list item, counting the number of elements in the list, output the names of the list items, delete function of the even list elements. Created user-friendly interface for working with the program.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной курсовой работы является закрепление знаний по созданию структур данных в языке Си. Завладеть знаниями в создании и работе с двунаправленным линейным списком. Консолидировать знания в работе с динамической памятью.

1. ОПИСАНИЕ ТЕЛА ФУНКЦИИ MAIN.C

1.1. При помощи функции ввода общего значения scanf вводится количество количество композиций элементов, составляющих список.

1.2. Динамически выделяется память под названия композиций, имена авторов, и год создания композиции.

1.3. Заполняются данные списка.

1.4. При помощи функции createMusicalComposition создаётся двунаправленный список по заполненным данным.

1.5. Далее пользователю предлагается набор функций для работы со списком:

1 – добавление новой композиции в конец списка;

2 – удаление элемента списка;

3 – удаление всех элементов списка, стоящих на нечетных местах;

4 – вывод количества композиций;

5 – вывод названий композиций;

6 – окончание работы со списком;

1.6. Освобождается динамическая память, выделенная для работы со списком.

2.ОПИСАНИЕ ФУНКЦИЙ ДЛЯ РАБОТЫ СО СПИСКОМ

2.1. Была создана структура MusicalComposition хранящая в себе 2 переменных-указателя char*, int и 2 указателя на структуру, содержащие адресса следующего и предыдущего элементов.

2.2. Создана функция для создания элемента списка-createMusicalComposition. Данная функция получает на вход три параметра: имя, автор, и год издания композиции. В ней выделяется память malloc`ом под новый элемент списка и используется функция strncpy для копирования данных о композиции в новый элемент списка. Функция возвращает первый элемент.

2.3. Создаётся функция push, которая добавляет элемент в конец списка музыкальных композиций. На вход функции подаётся 2 параметра: указатель на элемент, который надо добавить, и указатель на первый элемент. Функция ничего не возвращает.

2.4. Функция removeEl удаляет элемент списка, у которого имя композиции совпадает с именем композиции для удаления. На вход программе подаются: указатель на первый элемент и указатель на имя элемента для удаления. При помощи функции strcmp проверяется список на совпадение имя композиции с именем композиции для удаления. Функция ничего не возвращает.

2.5. Функция print_number возвращает количество элементов списка. На вход подаётся указатель на первый эл-т списка. При помощи цикла for подсчитывается количество элементов списка.

2.6. Создана функция remove_odd которая все элементы списка, стоящие на нечетных местах. На вход функции подаётся указатель на первый элемент списка. При помощи циклических операторов происходит удаление нечетных элементов списка, с учётом того что в списке может быть один и более элементов. Функция ничего не возвращает.

2.7. Функция print_names выводит названия композиций. На вход программе подаётся указатель на первый элемент списка. При помощи цикла for и функции printf выводятся названия композиций. Функция ничего не возвращает.

ЗАКЛЮЧЕНИЕ

В ходе данной курсовой работы были закреплены знания по работе со структурами данных в языке Си. Получены знания по созданию двунаправленного линейного списка, знания по работе и созданию функций для работы со списком. Были закреплены знания по выделению и очищению динамической памяти. Реализована функция, удаляющая все чётные элементы списка, а также пользователю представлен удобный интерфейс для работы со списком.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Б. Керниган, Д. Риччи «Язык программирования Си».

2. https://stepik.org/course/1096/syllabus

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

1. main.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stddef.h>
#include "opt.h"

int main() {
	int len;
	printf("Введите количество композиций:\n");
	scanf("%d", &len);
	char m = getchar();

	char** names = (char**)malloc(len*sizeof(char*));
	char** authors = (char**)malloc(len*sizeof(char*));
	int* years = (int*)malloc(len*sizeof(int));

	MusicalComposition* head = NULL;
	MusicalComposition* element_for_push;

	if (len == 0) {
		head == NULL;
		printf("Список пуст!\n");
		exit(0);
	}
	else {
		for (int i = 0; i < len; i++) {
			char name[80];
			char author[80];
			printf("Введите название композиции:\n");
			fgets(name, 80, stdin);
			printf("Введите автора композиции:\n");
			fgets(author, 80, stdin);
			printf("Введите год создания:\n");
			fscanf(stdin, "%d", &years[i]);
```

```
                        char n = getchar();

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

                        names[i] = (char*)malloc((strlen(name)+1)*sizeof(char*));
                        authors[i] = (char*)malloc((strlen(author)
+1)*sizeof(char*));

                        strcpy(names[i], name);
                        strcpy(authors[i], author);

                }
        }

        head = createMusicalCompositionList(names, authors, years, len);

        char name_for_push[81];
        char author_for_push[81];
        int year_for_push;

        char name_for_remove[81];

        printf("Действия со списком:\n");
        printf("1 - Добавление новой композиции в конец списка.\n");
        printf("2 - Удаление композиции из списка.\n");
        printf("3 - Удаление композиций, стоящих на нечетных
позициях.\n");
        printf("4 - Вывод количества композиций в списке.\n");
        printf("5 - Вывод названий композиций в списке.\n");
        printf("6 - завершение работы со списком.\n");

        int opt;
        int flag = 1;

        while (flag) {
                printf("Введите номер возможного действия со списком:\n");
                scanf("%d", &opt);
                char t = getchar();
                switch (opt) {
```

```
                case 1:
                        printf("Введите название композиции:\n");
                        fgets(name_for_push, 81, stdin);
                        (*strstr(name_for_push,"\n"))=0;
                        printf("Введите автора композиции:\n");
                        fgets(author_for_push, 81, stdin);
                        (*strstr(author_for_push,"\n"))=0;
                        printf("Введите год создания:\n");
                        fscanf(stdin, "%d", &year_for_push);
                        char k = getchar();
                        MusicalComposition* element_for_push =
createMusicalComposition(name_for_push, author_for_push, year_for_push);
                        if (head == NULL) {
                                head = element_for_push;
                        }
                        else {
                                push(head, element_for_push);
                        }
                        break;
                case 2:
                        if (head == NULL) {
                                printf("Список пуст!\n");
                        }
                        else {
                                printf("Введите название композиции, которую
вы хотите удалить:\n");
                                fgets(name_for_remove, 81, stdin);
                                (*strstr(name_for_remove,"\n"))=0;
                                removeEl(&head, name_for_remove);
                        }
                        break;
                case 3:
                        if (head == NULL) {
                                printf("Список пуст!\n");
                        }
                        else {
                                printf("Композиции, стоящие на нечетных местах
были удалены.\n");
                                remove_odd(head);
```

```
                        printf("оставшиеся композиции в списке:\n");
                        print_names(head);
                    }
                    break;
                case 4:
                    if (head == NULL) {
                        printf("Список пуст!\n");
                    }
                    else {
                        printf("Количество композиций: %d\n",
print_number(head));
                    }
                    break;
                case 5:
                    if (head == NULL) {
                        printf("Список пуст!\n");
                    }
                    else {
                        printf("Названия композиций в списке:\n");
                        print_names(head);
                    }
                    break;
                case 6:
                    printf("До свидания!\n");
                    flag = 0;
                    break;
                default:
                    printf("Необходимо ввести число от 1 до 6!\n");
                }

            }

            if (head != NULL) {
                for (int i = 0; i < len; i++) {
                    free(names[i]);
                    free(authors[i]);
                }

                free(names);
                free(authors);
```

```
                free(years);

                MusicalComposition* tmp = head->next;

                while (tmp != NULL) {
                        free(tmp->prev->name);
                        free(tmp->prev->author);
                        free(tmp->prev);
                        tmp = tmp->next;
                }

                if (tmp = NULL) {
                        free(head->name);
                        free(head->author);
                        free(head);
                }

                free(tmp);
        }
    return 0;
    }
```

2. opt.c

```
    #include <stdlib.h>
    #include <stdio.h>
    #include <string.h>
    #include <stddef.h>
    #include "opt.h"

    MusicalComposition* createMusicalComposition(char* name, char*
author,int year) {
            MusicalComposition*
newMC=(MusicalComposition*)malloc(sizeof(MusicalComposition));
            newMC->name = (char*)malloc(80*sizeof(char));
            strncpy(newMC->name, name, 81);
            newMC->author = (char*)malloc(80*sizeof(char));
            strncpy(newMC->author, author, 81);
            newMC->year = year;
            newMC->next = NULL;
            newMC->prev = NULL;
```

```
	return newMC;
	}

	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 = head;

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

	void push(MusicalComposition* head, MusicalComposition* element) {
		MusicalComposition* tmp = head;

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

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

		for (tmp = *head; tmp != NULL; tmp = tmp->next) {
			if (strcmp(tmp->name,name_for_remove) == 0) {
				if (tmp->prev == NULL) {
					if (tmp->next == NULL) {
						*head = NULL;
					}
					else {
						tmp->next->prev=NULL;
```

```
                        *head = tmp->next;
                    }
                }
                else {
                    if (tmp->next == NULL) {
                        tmp->prev->next = NULL;
                    }
                    else {
                        tmp->next->prev = tmp->prev;
                        tmp->prev->next = tmp->next;
                    }
                }
                break;
            }
        }
}

int print_number(MusicalComposition* head) {
    MusicalComposition* tmp = head;
    int size = 0;

    for (size = 0; tmp != NULL; size++) {
        tmp = tmp->next;
    }
return size;
}

void print_names(MusicalComposition* head) {
    MusicalComposition* tmp;
    if (head == NULL)
        printf("Список пуст!\n");
    for (tmp = head; tmp != NULL; tmp = tmp->next) {
        printf("%s\n",tmp->name);
    }
}

void remove_odd(MusicalComposition* head) {

    MusicalComposition* tmp;
    int flag = 1;
```

```
        if (head->next == NULL) {
                free(head->name);
                free(head->author);
                free(head);
                head = NULL;
                flag = 0;
        }
        else {
                tmp = head->next;
                free(head->name);
                free(head->author);
                free(head);
                *head = *tmp;
                head->prev = NULL;
        }

        while (flag) {
                if (head->next == NULL) {
                        break;
                }
                else if (head->next->next == NULL) {
                        free(head->next->name);
                        free(head->next->author);
                        free(head->next);
                        head->next = NULL;
                        break;
                }
                else {
                        tmp = head->next;
                        head = head->next->next;
                        tmp->prev->next = tmp->next;
                        tmp->next->prev = tmp->prev;
                        free(tmp->name);
                        free(tmp->author);
                        free(tmp);
                }
        }
}
```

3. opt.h

```
#pragma once

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* createMusicalCompositionList(char** array_names, char** array_authors, int* array_years, int n);
void push(MusicalComposition* head, MusicalComposition* element);
void removeEl(MusicalComposition** head, char* name_for_remove);
int count(MusicalComposition* head);
void deleting(MusicalComposition* head);
```

4. Makefile

```
all:main.o opt.o
	gcc main.o opt.o
main.o:main.c opt.h
	gcc -c main.c
opt.o: opt.c opt.h
	gcc -c opt.c
```