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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Структуры данных, линейные списки

Студентгр. 7381 _____ Дорох С.В.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург

2017

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Дорох С.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 _____ Дорох С.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 чётных элементов списка;

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 **count** 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 **deleting** которая все четные элементы списка. На вход функции подаётся указатель на первый элемент списка. При помощи циклических операторов происходит удаление

чётных 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».

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 length;
   printf("Enter the number of compositions:\n");
   scanf("%d", &length);
   char m = getchar();

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

   MusicalComposition* head = NULL;
   MusicalComposition* element_for_push;

   if(length == 0){
      head == NULL;
      printf("List is empty!\n");
      exit(0);
   }
   else{

      for (int i=0;i<length;i++)
      {
         char name[80];
         char author[80];
         printf("Enter the name of composition:\n");
         fgets(name, 80, stdin);
         printf("Enter the name of author of the composition:\n");
         fgets(author, 80, stdin);
         printf("Enter release year:\n");
         fscanf(stdin, "%d", &years[i]);
         char n = getchar();
```

```
        (*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);

    }
}
head = createMusicalCompositionList(names, authors, years, length);

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

char name_for_remove[81];

printf("Options:\n");
printf("1 - adding new composition to the end of list.\n");
printf("2 - deleting element of list.\n");
printf("3 - deleting even elements of list.\n");
printf("4 - output the number of compositions.\n");
printf("5 - output the names of compositions.\n");
printf("6 - end of work with list.\n");


int opt = 0;
int run = 1;
while(run){
    printf("Enter the number of option:\n");
    scanf("%d", &opt);
    char t = getchar();
    switch(opt){
        case 1:
            printf("Enter the name of comp:\n");
            fgets(name_for_push, 81, stdin);
            (*strstr(name_for_push,"\n"))=0;
            printf("Enter the name of author:\n");
            fgets(author_for_push, 81, stdin);
            (*strstr(author_for_push,"\n"))=0;
            printf("Enter the release year:\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("List is empty!\n");
            else{
               printf("Enter the name of composition,which you want do delete:\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("List is empty!\n");
            else{
               printf("Deleting even composition...\n");
               deleting(head);
               print_names(head);
            }
            break;
         case 4:
            if(head == NULL)
               printf("List is empty!\n");
            else
               printf("The number of composition:%d\n",count(head));
            break;
         case 5:
            if(head == NULL)
            printf("List is empty!\n");
            else{
               printf("Names of compositions:\n");
               print_names(head);
            }
            break;
         case 6:
            printf("Bye!\n");
            run = 0;
```

```
            break;

        default: printf("The entered data is incorrect,repeat again\n");
        }

    }

    for (int i=0;i<length;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"
```

```
// Создание структуры MusicalComposition

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;
}

// Функции для работы со списком MusicalComposition

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 count(MusicalComposition* head){
   MusicalComposition* tmp = head;
   int size;
   for( size = 0; tmp != NULL ; size++)
      tmp = tmp->next;
   return size;
}

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

void deleting(MusicalComposition* head){
   MusicalComposition* tmp;
   while(1){
      if (head->next==NULL)  //Если дальше эл-тов нет прерываю цикл
```

```
            break;
        else if (head->next->next==NULL){ //Если следующий эл-т последний(он
чётный)
            free(head->next->name);
            free(head->next->author);
            free(head->next);
            head->next = NULL;
    break;
    }
        else {  // Если больше одного элемента
            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);
```