

**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

КУРСОВА ЗАДАЧА

ПО ПИК II

*Разработил: Петър Михайлов*

Ръководител: Момчил Петков

Факултет: ФКСТ

Поток: 9

Фак. №: 121217035

Дата: 22.05.2018

**1.Текст на заданиетo**

*Задание No*  **3 дисциплина: ПИК II – курсова задача**

***Студент:***

…………………………………………………………………………

**Да се разработи програма тип „меню” за поддържане на информацията за видеотека със следните изисквания:**

1. Информацията за филмите във видеотеката се съхраняват в двоични файлове със следните данни:

* Номенклатурен номер - уникално 12 цифрено число;
* Име на филма - до 50 символен низ;
* Режисьор - до 50 символен низ;
* Продължителност - реално число;
* Година - цяло число;

1. Да се извършват следните обработки:   
   а) Добавяне на нов филм във видеотеката;  
   б) Изтриване на филмите на даден режисьор;   
   в) Извеждане на информация за всички филми, подредени в намаляващ ред на годината;  
   г) Намиране на най-продължителният филм.
2. Данните да се поддържат в динамична структура - едносвързан списък в оперативната памет на ПК.

## ИЗИСКВАНИЯ КЪМ ОФОРМЛЕНИЕТО

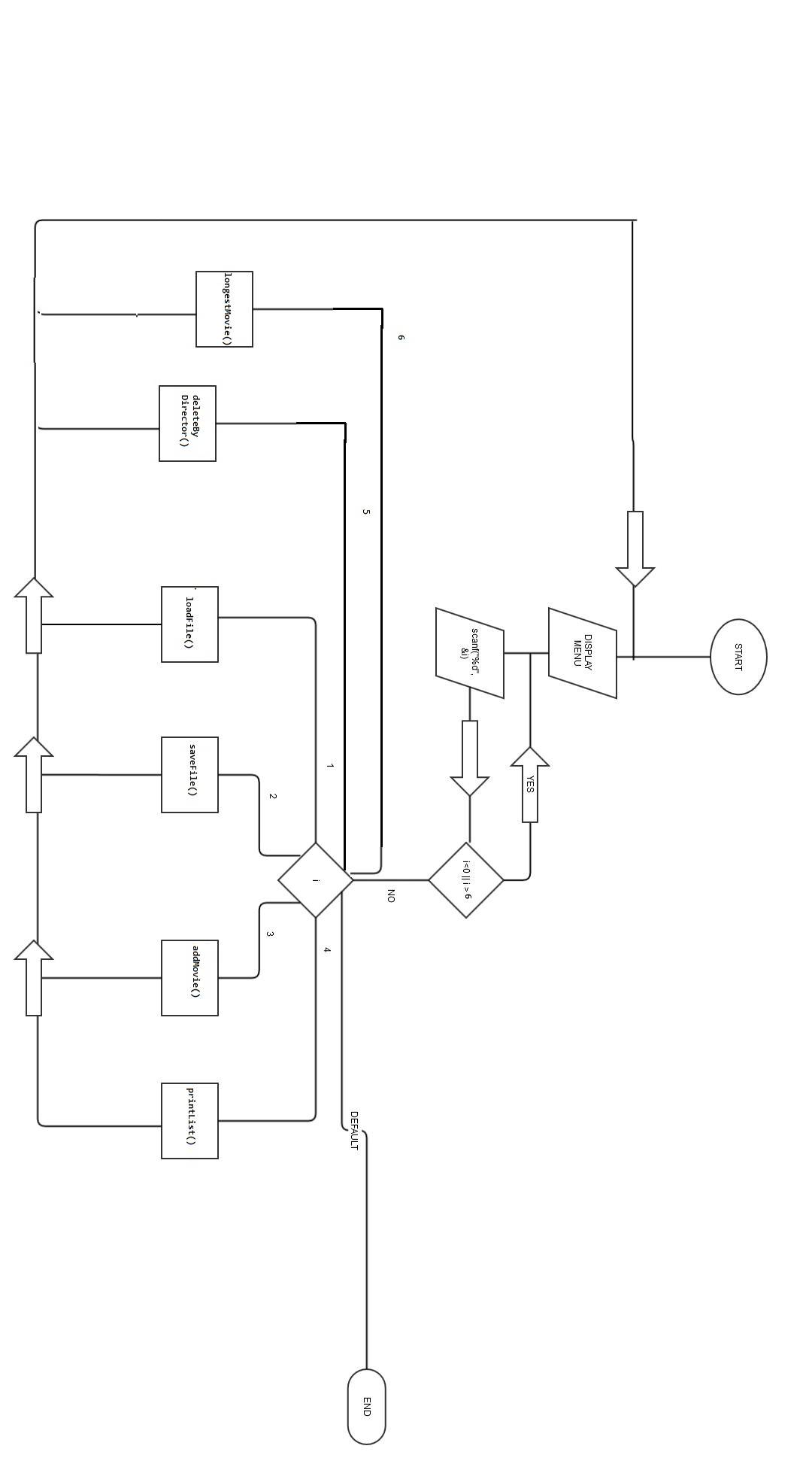
Задачата да се оформи като задача, съдържаща:

* титулна страница с данни за студента, ръководителя на курсовата задача;
* текст на заданието;
* обобщен блоков алгоритъм на разработеното програмно осигуряване;
* описание на използуваните модули (функции) - прототип, входно изходни параметри и предназначение;
* общо описание за функциониране на програмата (вход/изход);
* листинг на source (изходния) код на програмата;
* резултати от изпълнението на програмата (контролен пример);
* проектът да се реализира в програмната среда като проект с разделна компилация.

**Дата на задаване: Преподавател:**

/…………………………..

2. **Обобщен блоков алгоритъм на разработеното програмно осигуряване**



1. **Описание на използуваните модули (функции)**

int menu(); - функция, която отваря диалогов прозорец с опции

int isUnique(int ID, node\* head); - функция, която проверява за припокриващи се ID-та

movie initMovie(node\* head); - функция, която въвежда информация за филма

void printMovie(movie data); - функция, която извежда на екрана информация да филм

void swapNode(node\*\* A, node\*\* B); - функция, която разменя 2 филма в структурата

void bubbleSortByYear(node\*\* head); - функция, която подрежда филмите по година

void printLongestMovie(node\* head); - функция, която извежда на екрана филма, с най-голяма продължителност

void deleteByDirector(node\*\* head); - функция, която трие филм по зададен режисьор

int isEmpty(node\* head); - функция, която проверява дали списъкът е празен

int nodesCounter(node\* head); - функция, която брои елементите в списъка

void printNode(node\*\* head); - функция, която извежда целия списък

void addNode(node\*\* head, movie data, int mode); - функция, чрез която се навързва списъка

void deleteNode(node\*\* head); - функция, която изтрива целия списък

void deleteNodeAtPoss(node\*\* head, int poss); - функция, която изтрива само зададен елемент от списъка

char\* getFileName(); - функция, която взима името за файл

void loadFile(node\*\* head); - функция, която отваря бинарен файл

void saveFile(node\* head); - функция, която запазва списъка в бинарен файл

void getstr(char\* str, int maxLen); - функция, чрез която въвеждаме низ

**4.Общо описание за функциониране на програмата (вход/изход)**

Програмата изкарва на екран меню с 6 опции. При избиране на дадена опция от потребителя, програмата изпълнява условието й и връща на екран резултата от нея. При избиране на ‚изход‘ се излиза от програмата.

**5.Листинг на source (изходния) код на програмата**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

typedef struct{

int ID;

char title[51];

char director[51];

double length;

int year;

}movie;

typedef struct Node{

movie data;

struct Node \*next;

} node;

//Init Movie

int isUnique(int ID, node\* head);

movie initMovie(node\* head);

void printMovie(movie data);

void swapNode(node\*\* A, node\*\* B);

void bubbleSortByYear(node\*\* head);

void printLongestMovie(node\* head);

void deleteByDirector(node\*\* head);

//Nodes

int isEmpty(node\* head);

int nodesCounter(node\* head);

void printNode(node\*\* head);

void addNode(node\*\* head, movie data, int mode);

void deleteNode(node\*\* head);

void deleteNodeAtPoss(node\*\* head, int poss);

//Files

char\* getFileName();

void loadFile(node\*\* head);

void saveFile(node\* head);

//General

void getstr(char\* str, int maxLen);

int strCmpIgnoreCase(char\* str1, char\* str2);

int menu(){

int i;

printf("Main Menu:\n\n");

printf("1. Load Data\n");

printf("2. Save Data\n");

printf("3. Add New Movie\n");

printf("4. Print All Movies\n");

printf("5. Delete Movies By Director\n");

printf("6. Print Longest Movie\n\n");

printf("0. Exit\n\n");

do{

printf("SELECT OPTION: ");

fflush(stdin);

scanf("%d", &i);

} while (i < 0 || i > 6);

return i;

}

int main(){

node\* head = NULL;

while (1){

system("cls");

switch (menu()){

case 1: loadFile(&head);

break;

case 2: saveFile(head);

break;

case 3: addNode(&head, initMovie(head), 1);

break;

case 4: printNode(&head);

break;

case 5: deleteByDirector(&head);

break;

case 6: printLongestMovie(head);

break;

default:

return 0;

}

system("pause");

}

}

//Delete By Director

void deleteByDirector(node\*\* head){

char director[51];

int poss = 1, count = 0;

node\* temp = (\*head);

char c;

if (!(\*head)){

printf("List is Empty!\n");

return;

}

printf("Enter Director: ");

getstr(director, 51);

printf("Movies By %s\n", director);

while (temp){

if (!strCmpIgnoreCase(director, temp->data.director)){

printf("\nMovie %d\n", ++count);

printMovie(temp->data);

}

temp = temp->next;

}

if (!count){

printf("No movies Found!");

return;

}

printf("Do you want do delete data<Y/N>");

do{

fflush(stdin);

c = toupper(getchar());

if (c != 'Y' && c != 'N'){

printf("Select <Y/N>: ");

}

} while (c != 'Y' && c != 'N');

if (c == 'N') {

return;

}

temp = (\*head);

while (temp){

if (!strCmpIgnoreCase(director, temp->data.director)){

temp = temp->next;

deleteNodeAtPoss(head, poss);

}

else{

temp = temp->next;

++poss;

}

}

printf("All Movies by %s Deleted!\n", director);

}

//Longest Movie

void printLongestMovie(node\* head){

node\* temp = head;

double maxLen = 0;

if (!head){

printf("No movies!\n");

return;

}

while (temp){

if (temp->data.length > maxLen){

maxLen = temp->data.length;

}

temp = temp->next;

}

while (head){

if (head->data.length == maxLen){

printf("\n");

printMovie(head->data);

}

head = head->next;

}

}

//Init Movie

int isUnique(int ID, node\* head){

while (head){

if (head->data.ID == ID){

return 0;

}

head = head->next;

}

return 1;

}

movie initMovie(node\* head){

movie newMovie;

while(1){

printf("Enter ID: ");

fflush(stdin);

scanf("%d", &newMovie.ID);

if (newMovie.ID < 1 || newMovie.ID > 999999999999){

fprintf(stderr, "Incorrect ID!\n");

continue;

}

if (!isUnique(newMovie.ID, head)){

fprintf(stderr, "ID Already Exist!\n");

continue;

}

break;

}

printf("Enter Title: ");

getstr(newMovie.title, 51);

printf("Enter Director: ");

getstr(newMovie.director, 51);

do{

printf("Enter Length: ");

fflush(stdin);

scanf("%lf", &newMovie.length);

} while (newMovie.length <= 0);

do{

printf("Enter Year: ");

fflush(stdin);

scanf("%d", &newMovie.year);

} while (newMovie.year < 1900 || newMovie.year > 2020);

return newMovie;

}

void printMovie(movie data){

printf("ID: %.12d\n", data.ID);

printf("Title: %s\n", data.title);

printf("Director: %s\n", data.director);

printf("Length: %.2lf\n", data.length);

printf("Year: %d\n", data.year);

}

//Sort

void swapNode(node\*\* A, node\*\* B){

movie temp = (\*(\*A)).data;

(\*A)->data = (\*B)->data;

(\*(\*B)).data = temp;

}

void bubbleSortByYear(node\*\* head){

node\* temp = \*head;

int i, j;

int n = nodesCounter(\*head);

for (i = 0; i < n - 1; i++, temp = (\*head)){

for (j = 0; j < n - i - 1; j++){

if (temp->data.year < temp->next->data.year){

swapNode(&temp, &temp->next);

}

temp = temp->next;

}

}

}

//Nodes

int isEmpty(node\* head){

if (head)

return 0;

printf("List is empty!\n");

return 1;

}

int nodesCounter(node\* head){

int n = 0;

while (head){

++n;

head = head->next;

}

return n;

}

void printNode(node\*\* head) {

int i = 0;

node\* temp = (\*head);

if (isEmpty(\*head))

return;

bubbleSortByYear(head);

while (temp){

printf("\nMovie %d\n", ++i);

printMovie(temp->data);

temp = temp->next;

}

}

void addNode(node\*\* head, movie data, int mode){

node\* temp = (node\*)malloc(sizeof(node));

node\* temp2 = \*head;

temp->data = data;

temp->next = NULL;

if (!(\*head)){

\*head = temp;

if (mode == 1)

printf("First movie added successfully!\n");

return;

}

while (temp2->next){

temp2 = temp2->next;

}

temp2->next = temp;

if (mode == 1)

printf("New movie added successfully!\n");

}

void deleteNode(node\*\* head){

node\* temp;

while (\*head){

temp = \*head;

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

free(temp);

}

}

void deleteNodeAtPoss(node\*\* head, int poss){

int i;

node \*temp1 = (\*head), \*temp2;

if (poss == 1){

(\*head) = temp1->next;

free(temp1);

return;

}

for (i = 0; i < poss - 2; i++){

temp1 = temp1->next;

//temp1 pointst to (n-1)th possition;

}

temp2 = temp1->next;//temp2 points to n-th poss Node;

temp1->next = temp2->next;

free(temp2);

}

//Files

char\* getFileName(){

char \*fileName = (char\*)malloc(51 \* sizeof(char));

printf("\nEnter file name: ");

getstr(fileName, 51);

if (!strchr(fileName, '.')){

strcat(fileName, ".dat");

}

return fileName;

}

void loadFile(node\*\* head){

FILE\* fp;

movie data;

char \*fileName;

fileName = getFileName();

if (!(fp = fopen(fileName, "rb"))){

fprintf(stderr, "Error loading file!\a\n");

free(fileName);

return;

}

if (\*head)

deleteNode(head);

while (1){

if (!fread(&data, sizeof(data), 1, fp))

break;

addNode(head, data, 2);

}

fclose(fp);

if (!(\*head))

printf("No data loaded from \"%s\"\n", fileName);

else

printf("Info loaded successfully from \"%s\"\n", fileName);

free(fileName);

}

void saveFile(node\* head){

FILE\* fp;

char \*fileName;

if (isEmpty(head)){

printf("Add some movies first!\n");

return;

}

fileName = getFileName();

if (!(fp = fopen(fileName, "wb"))){

exit(1);

}

while (head){

if (fwrite(&head->data, sizeof(movie), 1, fp) != 1){

printf("Error occured while writing on file\n");

exit(1);

}

head = head->next;

}

fclose(fp);

printf("Info saved successfully to \"%s\"\n", fileName);

free(fileName);

}

//General

void getstr(char\* str, int maxLen){

fflush(stdin);

fgets(str, maxLen \* sizeof(char), stdin);

str[strlen(str) - 1] = '\0';

}

int strCmpIgnoreCase(char\* str1, char\* str2){

char str1upper[51];

char str2upper[51];

unsigned i;

strcpy(str1upper, str1);

strcpy(str2upper, str2);

for (i = 0; i < strlen(str1); i++){

str1upper[i] = toupper(str1upper[i]);

}

for (i = 0; i < strlen(str2); i++){

str2upper[i] = toupper(str2upper[i]);

}

if (!strcmp(str1upper, str2upper)){

return 0;

}

return -1;

}

**6. Примери**