

**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

КУРСОВА ЗАДАЧА

ПО ПИК II

*Разработил: Kонстантин Kостов*

Ръководител:

Факултет: ФКСУ

Фак. №: 121217148

Дата:24.05.2017г.

**1.Текст на заданието**



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

* **Описание на използуваните модули (функции) - прототип, входно изходни параметри и предназначение**

**void insertFirm()**-функция, която дава възможност на потребителя да въведе информацията за нова фирма и записва новата информация накрая на списъка

**void printListStdNumber()-** функцията извежда на екран информацията за дадена фирма по въведен от потребителя номер

**void printList- ()-** функцията извежда на екран информацията за всички фирми

**void check()- Справка на всички фирми имащи средна печалба в зададен интервал**

**void update\_info()- ъпдейтва дадена информация за дадена фирма**

**retrieve() -взима всички запазени фирми от бинарен файл и създава нов лист**

**removeStdnumber() -изтрива фирма по зададен номер**

**void SaveInFile()-**функцията записва всички фирми, които сме записали в бинарен файл

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

Програмата изкарва на екран меню с 9 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.

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

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<stdbool.h>

#define NAMELEN 30 //Maximum characters for each name

#define MAXNUMGRADES 5

// LINKED LIST ----------------------------------------------------------------------------------------------------------------------

/\* A structure that contains a string that would be used for the firms's name,

an integer which would be used for the firms's =number

and an array of floating points which would be used for the student's marks,

an integer for the number of grades introduced for each firm,

and a pointer to the next node in the linked list

\*/

struct node

{

char name[NAMELEN];

int firmNumber;

int salary[MAXNUMGRADES];

int date\_reg;

struct node \*next;

float average\_salary;

};

// Function to insert new firm in Linked list in alphabetical order

void insertFirm(struct node\*\* head)

{

char wantedName[NAMELEN]; //String to store the name of the new firm when a new firm is introduced

printf("Add firm name: \n");

scanf("%s",wantedName);

int wantedfirmNumber; // Integer to store the firm number of the new firm when a new firm is introduced

printf ("Add firm number: \n");

scanf("%d",&wantedfirmNumber);

int wanted\_salary[5];

//Allocating memory for a new firm node

struct node\* new\_node = (struct node\*) malloc(sizeof(struct node));

//We copy the name and firm number parameters into the name and firm number of the firm

strcpy(new\_node->name, wantedName);

new\_node->firmNumber = wantedfirmNumber;

printf("Salary for the first year\n");

scanf("%d",&wanted\_salary[0]);

printf("Salary for the second year\n");

scanf("%d",&wanted\_salary[1]);

printf("Salary for the third year\n");

scanf("%d",&wanted\_salary[2]);

printf("Salary for the fourth year\n");

scanf("%d",&wanted\_salary[3]);

printf("Salary for the fifth year\n");

scanf("%d",&wanted\_salary[4]);

new\_node->salary[0]=wanted\_salary[0];

new\_node->salary[1]=wanted\_salary[1];

new\_node->salary[2]=wanted\_salary[2];

new\_node->salary[3]=wanted\_salary[3];

new\_node->salary[4]=wanted\_salary[4];

new\_node->average\_salary = (new\_node->salary[0] + new\_node->salary[1] + new\_node->salary[2] + new\_node->salary[3] + new\_node->salary[4])/5;

int dd,mm,yy;

int date;

printf("Enter date (yy.dd.mm) format: ");

scanf("%d.%d.%d",&yy,&dd,&mm);

/\*adding dd,mm,yy into date\*/

/\*an integer has 4 bytes and dd range is 1 to 31 , mm range is 1 to 12 which

\*can be stored in 1 byte, 1 byte and in rest of 2 bytes

\*we can store year.\*/

new\_node->date\_reg=0;

new\_node->date\_reg |= (dd&0xff); /\*dd storing in byte 0\*/

new\_node->date\_reg |= (mm&0xff)<<8; /\*mm storing in byte 1\*/

new\_node->date\_reg |= (yy&0xffff)<<16; /\*yy storing in byte 2 and 3\*/

//For traversing through the list, use two nodes next to each other moving them forward simultaneously

//pointer to the previous firm in the linked list

struct node\* previous = (struct node\*)malloc(sizeof(struct node));

previous = NULL;

//pointer to the current firm in the linked list

struct node\* current = (struct node\*)malloc(sizeof(struct node));

current = \*head;

//Looping through the firms until the name of the current is alphabetically 'higher' then the one we want to insert

while (NULL != current && strcmp(wantedName, current->name) > 0)

{

previous = current;

current = current->next;

}

if (NULL == previous)

{ //Insert at beginning of linked list

new\_node->next = current;

\*head = new\_node;

}

else

{ //Insert between previous and current node

previous->next = new\_node;

new\_node->next = current;

}

}

//Function to delete firm from the list using the firm's number

void removeStdNumber(struct node \*\*head){

int wantedStudentNumber; //Integer to store the firm number of the firm that the user wished to delete

printf("Please enter Firm number: \n");

scanf("%d",&wantedStudentNumber);

// For traversing through the list, we use two nodes next to each other moving them forward simultaneously

// Use a third node as an auxiliary node when deleting the node in order to fix the links

struct node \*current;

current=\*head;

struct node \*previous;

previous=NULL;

struct node \*temp;

temp=NULL;

// if list is empty

if(current == NULL){

printf("List empty, no items to delete\n\n");

return;

}

while(current != NULL){

// If match found

if(wantedStudentNumber == current->firmNumber){

// Delete at the start of the list

if (current == \*head){

temp=current;

\*head=current->next;

free(temp);

printf("Firm has been removed successfully!\n\n");

}

else{

temp=current;

previous->next=current->next;

free(temp);

printf("Firm has been removed successfully!\n\n");

}

return;

}

// If match not found move on to next

previous=current;

current = current->next;

}

//No match found, so error

printf("Student has not been introduced yet! \n\n");

}

// Function to update infor of a firm

void update\_info(struct node \*\*head){

char wantedName[NAMELEN]; //Integer to store the name of the firm whose report we wish to print

printf("Please enter firm name: \n");

scanf("%s",wantedName);

char wantedName1[NAMELEN]; //String to store the name of the new firm when a new student is introduced

// Node used for traversing the list

struct node \*temp;

temp=\*head;

while(temp !=NULL){

// If match found

if (strcmp(wantedName,temp->name)==0){

char wantedName1[NAMELEN]; //String to store the name of the new firm when a new student is introduced

printf("Add firm name: \n");

scanf("%s",wantedName1);

int wantedfirmNumber; // Integer to store the firm number of the new firm when a new firm is introduced

printf ("Add firm number: \n");

scanf("%d",&wantedfirmNumber);

strcpy(temp->name, wantedName1);

temp->firmNumber = wantedfirmNumber;

int wanted\_salary[5];

printf("Salary for the first year\n");

scanf("%d",&wanted\_salary[0]);

printf("Salary for the second year\n");

scanf("%d",&wanted\_salary[1]);

printf("Salary for the third year\n");

scanf("%d",&wanted\_salary[2]);

printf("Salary for the fourth year\n");

scanf("%d",&wanted\_salary[3]);

printf("Salary for the fifth year\n");

scanf("%d",&wanted\_salary[4]);

temp->salary[0]=wanted\_salary[0];

temp->salary[1]=wanted\_salary[1];

temp->salary[2]=wanted\_salary[2];

temp->salary[3]=wanted\_salary[3];

temp->salary[4]=wanted\_salary[4];

int dd,mm,yy;

int date;

printf("Enter date (yy.dd.mm) format: ");

scanf("%d.%d.%d",&yy,&dd,&mm);

/\*adding dd,mm,yy into date\*/

/\*an integer has 4 bytes and dd range is 1 to 31 , mm range is 1 to 12 which

\*can be stored in 1 byte, 1 byte and in rest of 2 bytes

\*we can store year.\*/

temp->date\_reg=0;

temp->date\_reg |= (dd&0xff); /\*dd storing in byte 0\*/

temp->date\_reg |= (mm&0xff)<<8; /\*mm storing in byte 1\*/

temp->date\_reg |= (yy&0xffff)<<16; /\*yy storing in byte 2 and 3\*/

}

// If not a match, move to next item

temp = temp->next;

}

//No match found, so error

printf("Firm has not been introduced yet, please introduce student first in order to print their report\n");

}

// function to : printf(" 9) Spravka na vsichki firmi imashti sredna pechalba v zadaden interval\n\n");

void check(struct node \*\*head){

printf("Spravka na vsichki firmi imashti sredna pechalba v zadaden interval\n");

printf("print the first digit\n");

float digit1;

float digit2;

scanf("%f",&digit1);

printf("print the second digit\n");

scanf("%f",&digit2);

struct node \*temp;

temp=\*head;

int dd;

int mm;

int yy;

if (\*head == NULL){

printf("No Firms have been introduced!\n");

}

while(temp != NULL) {

if(temp->average\_salary >digit1 && temp->average\_salary<digit2){

printf("Name: %s\n",temp->name);

}

temp = temp->next;

}

}

// Function to print report for one firm using firm number

void printListStdNumber(struct node \*\*head){

int wantedFirmNumber; // Integer to store firm number of firm whose report the user wihes to print

printf("Please enter firm number: \n");

scanf("%d",&wantedFirmNumber);

struct node \*temp;

temp=\*head;

while(temp !=NULL){

// If match found

if (wantedFirmNumber==temp->firmNumber){

printf("Name: %s\n",temp->name);

printf("Firm number: %d\n",temp->firmNumber);

printf("Salaries: \n");

for(int i=0; i<5;i++){

printf("\t Salary no.%d: %.2d\n",i+1,temp->salary[i]);

}

return;

}

// If not a amatch, move to next item

temp = temp->next;

}

//No match found, so error

printf("Firm has not been introduced yet, please introduce student first in order to print their report\n");

}

// Function to print report of all firms

void printList(struct node \*\*head) {

struct node \*temp;

temp=\*head;

int dd;

int mm;

int yy;

if (\*head == NULL){

printf("No Firms have been introduced!\n");

}

while(temp != NULL) {

printf("Name: %s\n",temp->name);

printf("Firm number: %d\n",temp->firmNumber);

printf("Salaries: \n");

for (int i=0; i<5; i++){

printf("\t Salary no.%d: %.2d\n",i+1,temp->salary[i]);

}

dd = (temp->date\_reg &0xff); /\*dd from byte 0\*/

mm = ((temp->date\_reg>>8)& 0xff); /\*mm from byte 1\*/

yy = ((temp->date\_reg>>16)&0xffff); /\*yy from byte 2 and 3\*/

printf(" Date: %04d.%02d.%02d\n",yy,dd,mm);

printf("\n\n");

temp = temp->next;

}

}

// Function to save data to file

void saveToFile(struct node \*\*head){

FILE \*outfile;

outfile = fopen ("Firms", "wb");

struct node \*temp;

temp=\*head;

if (outfile == NULL){

printf("Error opening file\n");

return;

}

while(temp != NULL){

fwrite(temp,sizeof(struct node),1,outfile);

temp=temp->next;

}

printf("Data saved successfully!\n\n");

fclose(outfile);

}

// Function to retrieve data from file

void retrieve(struct node \*\*head){

struct node \*tempObject = (struct node\*)malloc(sizeof(struct node));

struct node \*ptr;

struct node \*previous;

FILE \*file=fopen("Firms", "rb");

\*head=tempObject;

if (file != NULL){

do{

fread(tempObject,sizeof(struct node),1, file);

ptr=\*head;

previous=\*head;

while(previous->next != NULL){

ptr= (struct node \*)malloc(sizeof(struct node));

fread(ptr, sizeof(struct node), 1,file);

previous->next=ptr;

previous=ptr;

}

}while(fread(tempObject, sizeof(struct node), 1,file) == 1);

}

else{

printf("Error opening file\n");

return;

}

printf("Data retrieved successfully!\n\n");

}

//---------------------------------------------------------------------------------------------------------------------------------

int main(int argc, char const \*argv[]){

// Boundary condition so that the program does not crash

int menu\_choice;

int menu\_choice1;

int menu\_choice2;

int menu\_choice3;

int menu\_choice4;

int menu\_choice5;

int menu\_choice6;

int menu\_choice7;

int menu\_choice8;

int menu\_choice9;

struct node \*head=NULL;

do{

// Print menu on screen

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

printf (" 1) Introduce Firm \n");

printf (" 2) Remove firm \n");

printf (" 3) edit information \n");

printf (" 4) Print report for a firm\n");

printf (" 5) Print report for all firms\n");

printf (" 6) Save to File\n");

printf (" 7) Retrive from FIle\n");

printf (" 8) Exit\n");

printf(" 9) Spravka na vsichki firmi imashti sredna pechalba v zadaden interval\n\n");

printf (" Please select an option from the menu: \n");

scanf ("%d" , &menu\_choice); // Read user input for menu choice

switch (menu\_choice)

{

case 1:

printf("Press 1 if you wish to continue:\n");

printf("Press 0 to go back to Menu :\n");

scanf("%d",&menu\_choice1);

if (menu\_choice1==1){

insertFirm(&head);

}

else if(menu\_choice1==0){

break;

}

else{

printf("Invalid choice, please select your option from the menu once again:\n");

}

break;

case 2:

printf("Press 1 to remove Firm using their number:\n");

printf("Press 0 to go back to Menu:\n");

scanf("%d",&menu\_choice2);

if (menu\_choice2==1){

removeStdNumber(&head);

}

else if(menu\_choice2==0){

break;

}

else{

printf("Invalid choice, please select your option from the menu once again:\n");

}

break;

case 3:

printf("Press 1 to update information about the firm:\n");

printf("Press 0 to go back to Menu:\n");

scanf("%d",&menu\_choice3);

if (menu\_choice3==1){

update\_info(&head);

}

else if(menu\_choice3==0){

break;

}

else{

printf("Invalid choice, please select your option from the menu once again:\n");

}

break;

case 4:

printf("Press 1 to print report for an individual firm using the firms's number:\n");

printf("Press 0 to go back to Menu:\n");

scanf("%d",&menu\_choice4);

if (menu\_choice4==1){

printListStdNumber(&head);

}

else if(menu\_choice4==0){

break;

}

else{

printf("Invalid choice, please select your option from the menu once again:\n");

}

break;

case 5:

printf("Press 1 if you wish to continue:\n");

printf("Press 0 to go back to Menu :\n");

scanf("%d",&menu\_choice5);

if (menu\_choice5==1){

printList(&head);

}

else if(menu\_choice5==0){

break;

}

else{

printf("Invalid choice, please select your option from the menu once again:\n");

}

break;

case 6:

printf("Press 1 if you wish to continue:\n");

printf("Press 0 to go back to Menu :\n");

scanf("%d",&menu\_choice6);

if (menu\_choice6==1){

saveToFile(&head);

}

else if(menu\_choice6==0){

break;

}

else{

printf("Invalid choice, please select your option from the menu once again:\n");

}

break;

case 7:

printf("Press 1 if you wish to continue:\n");

printf("Press 0 to go back to Menu :\n");

scanf("%d",&menu\_choice7);

if (menu\_choice7==1){

retrieve(&head);

}

else if(menu\_choice7==0){

break;

}

else{

printf("Invalid choice, please select your option from the menu once again:\n");

}

break;

case 8:

printf("Press 1 if you wish to continue:\n");

printf("Press 0 to go back to Menu :\n");

scanf("%d",&menu\_choice8);

if (menu\_choice8==1){

printf ("You have selected to exit the menu \n");

}

else if(menu\_choice8==0){

break;

}

else{

printf("Invalid choice, please select your option from the menu once again:\n");

}

break;

case 9:

printf("Press 1 if you wish to continue:\n");

printf("Press 0 to go back to Menu :\n");

scanf("%d",&menu\_choice8);

if (menu\_choice9==1){

check(&head);

}

else if(menu\_choice9==0){

break;

}

else{

printf("Invalid choice, please select your option from the menu once again:\n");

}

break;

default:

printf("Invalid choice, please select your option from the menu once again\n");

break;

}

}while (menu\_choice!=8 || menu\_choice8!=1);

}