

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** 3

з дисципліни “ Основи програмування ”

тема “Робота з одномірними та багатомірними масивами. Рядки.”

|  |  |  |
| --- | --- | --- |
| Виконав(ла)  студент(ка) I курсу  групи КП-61  Щербина Вадим Олегович  (*прізвище, ім’я, по батькові*)  варіант №\_\_\_\_\_\_\_ |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Штрафні бали:   |  |  | | --- | --- | | **Термін здачі** | **Оформлення звіту** | |  |  | | Нараховані бали:   |  |  |  | | --- | --- | --- | | **Корект. програм (2 бала)** | **Відп. на теор. питання (1 бал)** | **Відп. на прогр. питання (2 бала)** | |  |  |  | | Сумарний бал:   |  | | --- | |  | |

Київ 2015

**Мета роботи**

Навчитися працювати зі структурами, файлами, динамічно виділяти пам’ять, а також вдосконалити та закріпити вміння роботи з вказівниками.

**Постановка завдання**

### **Завдання №1. Одномірний масив**

1. Описати нову струкутуру даних типу **"Професія"**. Визначити у структурі не менше 3-х полів різного типу, серед яких обов'язковими є: строка, ціле число, дробове число, **інша структура**.
2. Виконати декомпозицію завдання на окремі функції і описати їх прототипи у окремому header-файлі:
   1. Описати функції, одна з яких виконує запис даних із одної структури (через вказівник) у строку, та функцію, що виконує зворотнє перетворення.
   2. Описати функцію, що із тексту створює масив структур та зворотню функцію.
   3. Описати функцію, що видаляє структуру із масиву структур по індексу, функцію, що виконує перезапис структури по індексу даними із іншої структури.
   4. Описати функцію, що отримує вказівник на структуру, строку, що позначає поле у структурі та строку, що позначає нове значення вказаного поля у структурі і виконує перезапис поля заданої структури новим значенням.
   5. Описати функцію, що виконує дію над масивом вказівників на структури: **"Знайти всі професії із середньою зарплатою, більшою з Х"**.

### **Завдання №2. Робота із файловими потоками**

1. Реалізувати функцію для зчитування даних із текстового файлу і перетворення їх у масив структур.
2. Реалізувати функцію для запису масиву структур у текстовий файл.

### **Завдання №3. Тестування функцій**

1. Протестувати функції, що піддаються тестуванню, за допомогою assert() у окремій функції(-ях). Кожна функція має тестуватись мінімум на 5 тестових випадках.
2. Винести прототипи тестових функцій у окремий header-файл.
3. Винести тестові функції у окремий файл із кодом.

### **Завдання №4. Розробка CUI-інтерфейсу**

1. Створити CUI - консольний інтерфейс користувача (*просто взаємодія з користувачем через консоль*) для використання описаного функціоналу.
2. Всі прототипи відповідних функцій CUI описати у окремому header-файлі, а їх реалізацію винести у окремий .c файл із кодом.
3. Тестові функції запускати тільки при заданні певного аргумента командного рядка.
4. У файлі із кодом функції main() не повинно бути жодних інших функцій.
5. Весь ввід користувача потрбно перевіряти на валідність щоби не допускати збоїв у роботі програми.

**Тексти коду програм**

main.c

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <ctype.h>  #include <progbase.h>  #include <pbconsole.h>  #include "menu.h"  #include "assert.h"  int main(int argc, char \*argv[])  {  conHideCursor();  int number = 1000;  int i = 0;  int k = 0;  struct profecional \* profs[1000] = {NULL};  for(int j = 0; j < argc; j++)  {  if (strcmp("test",argv[j]) == 0)  {  validtesting();  validtestfloatin();  check(profs);  filetest();  checkdelete();  checkstruct2();  checkcopy();  return 0;  }  }  line();  printfirstmenu();  printdots(0);  while(1)  {  k = conGetChar();  conReset();  if(k == 'q')  {  cleanall();  cat();  break;  }  switch(k)  {  case '1':  printdots(1);  clearmainmenu();  printsecondmenu();  while(1)  {  k = conGetChar();  conReset();  if(k == 'q')  {  printdots(0);  clearstringtostruct();  clearmainmenu();  printfirstmenu();  break;  }  switch(k)  {  case '1':  clearstringtostruct();  stringtostruct(profs);  break;  case '2':  clearstringtostruct();  printsecondmenu();  structtostring(profs);  break;  case '3':  clearstringtostruct();  clear();  texttostruct(profs);  break;  case '4':  clearstringtostruct();  deletestruct(profs);  break;  case '5':  clearstringtostruct();  structtotext(profs);  break;  case '6':  clearstringtostruct();  copystruct(profs);  break;  case '7':  clearstringtostruct();  rewritefiealdinstruct(profs);  break;  case '8':  clearstringtostruct();  highestsalary(profs);  break;  }  }  break;  case '2':  printdots(1);  clearmainmenu();  printthirdmenu();  while(1)  {  k = conGetChar();  conReset();  if(k == 'q')  {  printdots(0);  clearstringtostruct();  clearmainmenu();  printfirstmenu();  break;  }  switch (k)  {  case '1':  clearstringtostruct();  fileread("input.txt", profs);  break;  case '2':  clearstringtostruct();  filewrite(profs);  break;  }  }  break;  }  }  for(i=0; i < number; i++)  {  if(profs[i]!= NULL){  free( profs[i]);  }  }  return 0;  } |

menu.c

|  |
| --- |
| #include <progbase.h>  #include <pbconsole.h>  #include "function.h"  #include "menu.h"  void line()  {  int i=0;  conClear();  conSetAttr (BG\_WHITE);  for(i=0; i < 121; i++)  {  printf(" ");  conMove(1,i);  }  for(i = 0; i < 121; i++)  {  printf(" ");  conMove(30,i);  }  for(i = 0; i < 121; i++)  {  printf(" ");  conMove(15,i);  }  for(i = 0; i < 30; i++)  {  printf(" ");  conMove(i,120);  }  for(i = 0; i < 31; i++)  {  printf(" ");  conMove(i,1);  }  printf("\n");  conReset();  }  void printfirstmenu()  {  for(int j=2; j < 5; j++)  {  for(int i = 2; i < 120; i++)  {  conMove(j,i);  conSetAttr(BG\_YELLOW);  printf(" ");  }  }  conReset();  conMove(3,55);  conSetAttr(BG\_YELLOW);  conSetAttr(FG\_RED);  printf("LAB 3");  conReset();  conSetAttr(FG\_GREEN);  conMove(8,45);  printf("1.Створити новий масив даних");  conMove(9,45);  printf("2.Зчитати масив даних із файлу або записати дані на файл");  conMove(10,45);  printf("q-вихід");  conReset();  conMove(122,1);  }  void clearmainmenu()  {  for(int j=2; j < 15; j++)  {  for(int i = 2; i < 120; i++)  {  conMove(j,i);  printf(" ");  }  }  }  void printsecondmenu()  {  conSetAttr(FG\_CYAN);  conMove(4,35);  printf("Виберіть дію");  conReset();  conSetAttr(FG\_GREEN);  conMove(5,35);  printf("1. Заповнити структуру через строку");  conMove(6,35);  printf("2. Вивести структуру по заданому індексу");  conMove(7,35);  printf("3. Заповнити структуру з тексту");  conMove(8,35);  printf("4. Видалити структуру по заданому індексу");  conMove(9,35);  printf("5. Вивести всі заповненні структури");  conMove(10,35);  printf("6. Скопіювати дані з одної структури в іншу");  conMove(11,35);  printf("7. Змінити поле структури по вказаному індексу");  conMove(12,35);  printf("8. Знайти назву професій в яких заробітня плата більша за введену");  conMove(13,35);  printf("q - вихід");  for(int j=2; j < 4 ; j++)  {  for(int i = 2; i < 120; i++)  {  conMove(j,i);  conSetAttr(BG\_YELLOW);  printf(" ");  }  }  conSetAttr(FG\_RED);  conSetAttr(BG\_YELLOW);  conMove(2, 45);  printf("Створeння нового масиву даних");  conMove(41,3);  conReset();  }  void clearstringtostruct()  {  for(int j = 16; j < 30; j++)  {  for(int i = 3; i < 118; i++)  {  printf(" ");  conMove(j,i);  }  }  conMove(28,120);  conSetAttr(BG\_WHITE);  printf(" ");  conReset();  }  void clear()  {  int i = 0;  int j = 0;  for(i = 16; i < 29; i ++)  {  for(j = 2; j < 120; j++)  {  conMove(i, j );  printf(" ");  }  }  }  void clearfield()  {  for(int i = 2; i < 120; i++)  {  conMove(16,i);  printf(" ");  conReset();  }  for(int i = 16; i < 30; i++)  {  conSetAttr(BG\_WHITE);  conMove(i,40);  printf(" ");  conReset();  }  for(int i = 17; i < 30 ; i++)  {  for(int j = 41; j < 120; j++)  {  conMove(i,j);  printf(" ");  }  }  }  void printthirdmenu()  {  for(int j=2; j < 5; j++)  {  for(int i = 2; i < 120; i++)  {  conMove(j,i);  conSetAttr(BG\_YELLOW);  printf(" ");  }  }  conReset();  conMove(3,55);  conSetAttr(BG\_YELLOW);  conSetAttr(FG\_RED);  printf("Menu 2");  conReset();  conSetAttr(FG\_GREEN);  conMove(8,45);  printf("1.Зчитати з файлу");  conMove(9,45);  printf("2.Записати на файл структури");  conReset();  conMove(10,45);  conSetAttr(FG\_GREEN);  printf("q - вихід");  conReset();  conMove(122,1);  }  void cleanall()  {  for(int i = 0; i < 31; i++)  {  for(int j = 0; j < 120; j++)  {  conMove(i,j);  printf(" ");  }  }  }  void cleanstr(int i, int j)  {  for(; j < 118; j++)  {  conMove(i,j);  printf(" ");  }  }  void printdots(int k){  const unsigned long MILLIS = 5;  if(k==0){  for(int i = 0;i<=120;i++){  conMove(1,i);  conSetAttr(BG\_CYAN);  puts(" ");  conMove(30,120-i);  conSetAttr(BG\_CYAN);  puts(" ");  if(i <=30){  conMove(i,1);  conSetAttr(BG\_GREEN);  puts(" ");  conMove(i,120);  conSetAttr(BG\_GREEN);  puts(" ");  }  sleepMillis(MILLIS);  }  }  else{  for(int i= 120;i > 0;i--){  conMove(1,i);  conSetAttr(BG\_MAGENTA);  puts(" ");  conMove(30,120-i);  conSetAttr(BG\_MAGENTA);  puts(" ");  if(i <=30){  conMove(i,1);  conSetAttr(BG\_WHITE);  puts(" ");  conMove(i,120);  conSetAttr(BG\_WHITE);  puts(" ");  }  sleepMillis(MILLIS);  }  }  conReset();  } |

function.c

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <pbconsole.h>  #include <time.h>  #include <ctype.h>  #include <progbase.h>  #include "function.h"  #include "menu.h"  #define NUMBEROFSTRUCTS 1000  void stringtostruct( struct profecional \* profs[])  {  int i =0;  char str[200];  conMove(16,3);  conMove(17,3);  conMove(41,3);  while(i != NUMBEROFSTRUCTS)  {  if(profs[i] == NULL)  {  profs[i] = malloc(sizeof(struct profecional));  printstructs(profs ,i, str);  break;  }  else  {  i++;  }  }  conReset();  conMove(51,45);  }  void printstructs(struct profecional \* profs[] ,int i, char \*str)  {  conMove(16,3);  conSetAttr(FG\_BLUE);  printf("Hазва професії: ");  conReset();  scanf("%s",str);  strcpy(profs[i]->profname,str);  while(1)  {  str[0] = '\0';  conMove(17,3);  conSetAttr(FG\_BLUE);  printf("Зарплата:");  conReset();  scanf("%s",str);  if(validtest(str)== 0 )  {  profs[i]->salary = atoi(str);  break;  }  else  {  cleanstr(17,11);  conMove(18,3);  conSetAttr(FG\_RED);  printf("Не правильний ввід введіть ще раз");  conReset();  }  }  cleanstr(18,3);  while(1)  {  str[0] = '\0';  conMove(18,3);  conSetAttr(FG\_BLUE);  printf("Pейтинг професії: ");  conReset();  scanf("%s",str);  if(validtestfloat(str)== 0 )  {  profs[i]->score = atof(str);  break;  }  else  {  cleanstr(18,19);  conMove(19,3);  conSetAttr(FG\_RED);  printf("Не правильний ввід введіть ще раз");  conReset();  }  }  cleanstr(19,3);  str[0] = '\0';  conMove(19,3);  conSetAttr(FG\_BLUE);  printf("Імя робітника: ");  conReset();  scanf("%s",str);  strcpy(profs[i]->information.name,str);  str[0] = '\0';  conMove(20,3);  conSetAttr(FG\_BLUE);  printf("Прізвище робітника: ");  conReset();  scanf("%s",str);  strcpy(profs[i]->information.surname,str);  while(1)  {  str[0] = '\0';  conMove(21,3);  conSetAttr(FG\_BLUE);  printf("Рік народження: ");  conReset();  scanf("%s",str);  if(validtest(str)== 0 )  {  cleanstr(22,3);  profs[i]->information.year = atoi(str);  break;  }  else  {  cleanstr(21,16);  conMove(22,3);  conSetAttr(FG\_RED);  printf("Не правильний ввід введіть ще раз");  conReset();  }  }  }  int validtest(char \*str)  {  int flag = 0;  for(int i = 0; i < strlen(str); i++)  {  if(isalpha(str[i]) || isspace(str[i]) || ispunct(str[i]) )  {  flag = 1;  }  }  if(flag == 0)  {  return 0;  }  else  {  return 1;  }  }  int validtestfloat(char \*str)  {  int flag = 2;  int i =0;  for(i = 0; i < strlen(str); i++)  {  if(str[i]=='.' && isdigit(str[i+1]))  {  flag = 0;  }  }  for(i = 0; i < strlen(str); i++)  {  if(isalpha(str[i]) || isspace(str[i]))  {  flag = 1;  }  }  if(flag >= 1)  {  //profs[i]->salary = atoi(str);  return 1;  }  else  {  return 0;  }  }  void structtostring( struct profecional \* profs[])  {  conReset();  int i = enterstuct();  if(profs[i] == NULL)  {  conMove(18,3);  conSetAttr(FG\_CYAN);  printf("Задана структура не завпонена");  conReset();  }  else  {  conMove(17,3);  printf("Назва професії: %s", profs[i]->profname);  conMove(18,3);  printf("Зарплата: %i", profs[i]->salary);  conMove(19,3);  printf("Рейтинг професії: %.1f", profs[i]->score);  conMove(20,3);  printf("Імя робітника: %s", profs[i]->information.name);  conMove(21,3);  printf("Прізвище робітника: %s", profs[i]->information.surname);  conMove(22,3);  printf("Рік народження: %i", profs[i]->information.year);  }  }  int enterstuct(void)  {  int i = 0;  char str[10];  conMove(16,3);  printf("Введіть номер структури яку потрібно вивести ");  scanf("%s",str);  for(int k = 0; k < 10; k++)  {  if (isdigit(str[k]))  {  i = atoi(str);  return i;  }  }  return 1001;  }  void texttostruct(struct profecional \* profs[])  {  int k = 0;  int i=0;  char str[200];  int flag = 0;  while(1)  {  clearstringtostruct();  conMove(41,3);  if(flag == 0)  {  conMove(16,3);  printf("Якщо хочете заповнити структуру нажміть 2");  k = conGetChar();  }  if(flag == 1)  {  conMove(16,3);  printf("Якщо бажаєте продовжити нажміть 2 якщо ні нажміть 1\n");  k = conGetChar();  }  flag = 1;  if(k=='1')  {  clearstringtostruct();  break;  }  if(k == '2')  {  cleanstr(16,3);  while(1)  {  if(profs[i]== NULL)  {  profs[i] = malloc(sizeof(struct profecional));  printstructs(profs , i, str);  break;  }  else  {  i++;  }  }  }  }  }  void structtotext(struct profecional \* profs[])  {  clearstringtostruct();  int i=0;  int z = 0;  const unsigned long MILLIS = 1;  int k = 0;  for(i=0; i< NUMBEROFSTRUCTS; i++)  {  if(profs[i] != NULL)  {  if(k > 60)  {  clearstringtostruct();  k=0;  }  if(profs[i]->profname !='\0' && profs[i]->salary !=0)  {  conMove(17,3+k);  printf("Назва професії: %s", profs[i]->profname);  conMove(18,3+k);  printf("Зарплата: %i", profs[i]->salary);  conMove(19,3+k);  printf("Рейтинг професії: %.1f", profs[i]->score);  conMove(20,3+k);  printf("Імя робітника: %s", profs[i]->information.name);  conMove(21,3+k);  printf("Прізвище робітника: %s", profs[i]->information.surname);  conMove(22,3+k);  printf("Рік народження: %i", profs[i]->information.year);  conMove(45,1);  k = k + 50;  conMove(25,3);  conSetAttr(FG\_RED);  printf("Якщо бажаєте переглянути далі нажміть '2' якщо хочете зупинитись нажміть '1'");  conReset();  z = conGetChar();  if (z == '1')  {  break;  }  if (z == '2')  {  }  }  }  sleepMillis(MILLIS);  fflush(stdout);  }  if(i == 1000 || z == '1')  {  cleanstr(25,3);  conMove(25,3);  conSetAttr(FG\_RED);  printf("Всі структури виведенні");  conReset();  }  conMove(41,3);  }  int scanint(char \* str)  {  int k = 0;  int flag = 0;  while(1)  {  if(validtest(str) == 0)  {  flag = 1;  k = atoi(str);  break;  }  }  if(flag == 1)  {  return k;  }  else  {  return -1;  }  }  int deltstruct1(struct profecional \* profs[],int i)  {  if(profs[i] != NULL)  {  free(profs[i]);  profs[i] = NULL;  return 0;  }  else  {  return 1;  }  }  void deletestruct(struct profecional \* profs[])  {  clearstringtostruct();  int i = 0;  char str[100];  conMove(16,3);  printf("Введіть індекс структури яку потрібно удалити ");  while(1)  {  str[0] = '\0';  scanf ("%s",str);  if(validtest(str) == 1)  {  conMove(17,3);  printf("Неправильний ввід");  }  else  {  i = scanint(str);  break;  }  }  if(deltstruct1(profs,i) == 1 )  {  conMove(18,3);  printf("Структура під номером %i не заповнена",i);  }  else  {  conMove(18,3);  printf("Структура видалена");  }  }  void copystruct (struct profecional \* profs[])  {  clearstringtostruct();  int i =0;  char str[100];  int k =0;  int flag = 0;  conMove(16,3);  printf("Введіть індекс структури з якої потрібно скопіювати значення ");  scanf ("%s",str);  i = scanint(str);  if(i != -1 && checkstruct(profs,i) == 1)  {  conMove(17,3);  conSetAttr(FG\_RED);  printf("Дана структура не заповнена");  conReset();  flag =1;  }  if( flag == 0)  {  str[0]='\0';  conMove(18,3);  printf("Введіть індекс структури в яку потрібно скопіювати значення ");  scanf ("%s",str);  k = scanint(str);  if(profs[k] == NULL)  {  profs[k] = malloc(sizeof(struct profecional));  }  }  copstr(profs, i , k);  conMove(45,1);  }  struct profecional \* copstr(struct profecional \* profs[],int i ,int k)  {  if(profs[i] == NULL)  {  return NULL;  }  else  {  strcpy(profs[k]->profname,profs[i]->profname);  profs[k]->salary= profs[i]->salary ;  profs[k]->score = profs[i]->score;  strcpy(profs[k]->information.name,profs[i]->information.name);  strcpy(profs[k]->information.surname,profs[i]->information.surname);  profs[k]->information.year = profs[i]->information.year ;  return profs[k];  }  }  int checkstruct(struct profecional \* profs[],int i)  {  if(profs[i] == NULL)  {  return 1;  }  else  {  return 0;  }  }  void rewhelp(int i, int j)  {  conMove(i,j);  printf("Якщо хочете продовжити виберіть поле для зміни якщо ні нажміть [q]");  }  void rewritefiealdinstruct(struct profecional \* profs[])  {  int i = 0;  int k =0;  char str[100];  conMove(16,3);  printf("Введіть індекс структури в якій потрібно перезаписати поле ");  while(1){  str[0] = '\0';  cleanstr(16,61);  conMove(16,62);  scanf ("%s",str);  if(validtest(str) == 0){  i = scanint(str);  break;  }  else{  conMove(18,3);  printf("Не правильний ввід");  }  }  if (profs[i] == NULL)  {  profs[i] = malloc(sizeof(struct profecional));  }  conMove(17,3);  conSetAttr(FG\_CYAN);  printf("Введіть яке поле змінити\n");  conReset();  conMove(18,3);  printf("1 - назва професії\n");  conMove(19,3);  printf("2 - заробітня плата\n");  conMove(20,3);  printf("3 - рейтинг професії\n");  conMove(21,3);  printf("4 - імя робітника\n");  conMove(22,3);  printf("5 - прізвище робітника\n");  conMove(23,3);  printf("6 - рік народження працівника\n");  while(1)  {  k = conGetChar();  if (k == 'q')  {  clearstringtostruct();  break;  }  switch (k)  {  case 49:  clearfield();  conMove(18,45);  printf("Введіть назву професії ");  scanf("%s",str);  strcpy(profs[i]->profname,str);  rewhelp(19,45);  break;  case 50:  clearfield();  while(1)  {  str[0] = '\0';  conMove(19,45);  printf("Введіть заробітню плату ");  scanf("%s",str);  if(validtest(str)== 0 )  {  profs[i]->salary = atoi(str);  break;  }  else  {  cleanstr(19,60);  conMove(20,45);  conSetAttr(FG\_RED);  printf("Не правильний ввід введіть ще раз");  conReset();  }  }  cleanstr(20,45);  rewhelp(20,45);  break;  case 51:  clearfield();  while(1)  {  str[0] = '\0';  conMove(20,45);  printf("Введіть рейтинг професії ");  scanf("%s",str);  if(validtestfloat(str)== 0 )  {  profs[i]->score = atof(str);  break;  }  else  {  cleanstr(20,49);  conMove(21,45);  conSetAttr(FG\_RED);  printf("Не правильний ввід введіть ще раз");  conReset();  }  }  cleanstr(20,45);  rewhelp(21,45);  break;  case 52:  str[0]= '\0';  clearfield();  conMove(21,45);  printf("Введіть імя працівника ");  scanf("%s",str);  strcpy(profs[i]->information.name,str);  rewhelp(22,45);  break;  case 53:  str[0] = '\0';  clearfield();  conMove(22,45);  printf("Введіть прізвище ");  scanf("%s",str);  strcpy(profs[i]->information.surname,str);  rewhelp(23,45);  break;  case 54:  clearfield();  while(1)  {  str[0] = '\0';  conMove(23,45);  printf("Введіть рік народження ");  scanf("%s",str);  if(validtest(str)== 0 )  {  profs[i]->score = atoi(str);  break;  }  else  {  cleanstr(23,45);  conMove(24,45);  conSetAttr(FG\_RED);  printf("Не правильний ввід введіть ще раз");  conReset();  }  }  cleanstr(24,68);  rewhelp(24,45);  break;  }  }  }  int filecheck(const char \* readFileName)  {  const char \*read = readFileName;  FILE \* fin = fopen(read, "r");  if (fin == NULL)  {  return EXIT\_FAILURE;  }  else  {  return EXIT\_SUCCESS;  }  }  int fileread(const char \* readFileName, struct profecional \*profs[])  {  enum { BUFFER\_SIZE = 1000 };  const char \*read = readFileName;  char buffer[BUFFER\_SIZE];  int i = 0;  FILE \* fin = fopen(read, "r");  if (filecheck(read) == 1)  {  conMove(16,3);  printf("Error opening file\n");  return EXIT\_FAILURE;  }  while (!feof(fin))  {  fgets(buffer, BUFFER\_SIZE, fin);  buffer[strlen(buffer) - 1] = '\0';  while (i != NUMBEROFSTRUCTS)  {  if (profs[i] == NULL)  {  profs[i] = malloc(sizeof(struct profecional));  break;  }  else  {  i++;  }  }  sscanf(buffer,"%s %i %f %s %s %i",profs[i]->profname,&profs[i]->salary,&profs[i]->score,profs[i]->information.name,profs[i]->information.surname,&profs[i]->information.year);  buffer[0]= '\0';  i++;  }  conMove(17,3);  printf("З файла считано ");  fclose(fin);  return EXIT\_SUCCESS;  }  char filename(char str[100])  {  fgets(str, 200, stdin);  return \*str;  }  int filewrite(struct profecional \*profs[])  {  char str[100];  conMove(16,3);  printf("Введіть назву файла в який потрібно записати структури ");  filename(str);  char \*write = str;  FILE \*fout = fopen(write, "w");  for(int i = 0; i < NUMBEROFSTRUCTS; i++)  {  if(profs[i] != NULL && profs[i]->profname != '\0' && profs[i]->salary !=0 )  {  fprintf(fout,"Професія:%s| Зарплата:%i| Рейтинг професії%1.f| Імя: %s Прізвище: %s| Рік: %i\n ", profs[i]->profname,profs[i]->salary,profs[i]->score,profs[i]->information.name,profs[i]->information.surname,profs[i]->information.year);  }  }  fclose(fout);  return EXIT\_SUCCESS;  }  void highestsalary(struct profecional \*profs[])  {  const unsigned long MILLIS = 500;  int midsalary = 0;  int k = 17;  conMove(16,3);  printf("Введіть заробітню плату відносно якої потрібно вивести профеcії з більшою зарплатою ");  scanf ("%i", &midsalary);  for(int i = 0; i < NUMBEROFSTRUCTS; i++)  {  if(checksalary(profs,i,midsalary) == 0 )  {  if(k==29)  {  k=17;  clear();  }  conMove(k,3);  printf("Професії з середньою зарплатою більшою за введену: %s",profs[i]->profname );  sleepMillis(MILLIS);  k++;  fflush(stdout);  }  }  }  int checksalary(struct profecional \*profs[],int i,int midsalary)  {  if(profs[i] != NULL && profs[i]->salary > midsalary)  {  return 0;  }  else  {  return 1;  }  } |

assert.c

|  |
| --- |
| #include <assert.h>  #include <stdlib.h>  #include <stdio.h>  #include <string.h>  #include "assert.h"  void validtesting()  {  assert(validtest("11") == 0);  assert(validtest("1a1") == 1);  assert(validtest(" ") == 1);  assert(validtest(".") == 1);  assert(validtest("<^-^>") == 1);  }  void validtestfloatin()  {  assert(validtestfloat("sss") == 1);  assert(validtestfloat("2.3") == 0);  assert(validtestfloat(" ") == 1);  assert(validtestfloat("sssss2.3sss") == 1);  }  void check()  {  int size = 3;  struct profecional \* prof[size];  for (int i = 0; i < size; i++)  {  prof[i] = malloc(sizeof(struct profecional));  }  strcpy(prof[1]->profname, "cccc");  prof[1]->salary = 1000;  assert(checksalary(prof,1,1001) == 1);  assert(checksalary(prof,1,1) == 0);  for (int i = 0; i < size; i++)  {  free(prof[i]);  }  }  void filetest()  {  assert(filecheck("input.txt") == 0);  assert(filecheck("1244444") == 1);  }  void validdel1()  {  assert(scanint("isss") == -1);  assert(scanint("55") == 55);  assert(scanint("<-<") == -1);  assert(scanint("dd4") == -1);  }  //int deltstruct1(struct profecional \* profs[],int i);  void checkdelete()  {  int size = 3;  struct profecional \* prof[size];  for (int i = 0; i < size; i++)  {  prof[i] = malloc(sizeof(struct profecional));  }  prof[0] = NULL;  assert(deltstruct1(prof,2) == 0);  assert(deltstruct1(prof,0) == 1);  for (int i = 0; i < size; i++)  {  free(prof[i]);  }  }  void checkstruct2()  {  int size = 4;  struct profecional \* prof[size];  for (int i = 0; i < size; i++)  {  prof[i] = malloc(sizeof(struct profecional));  }  prof[0] = NULL;  assert(checkstruct(prof,2) == 0);  assert(checkstruct(prof,0) == 1);  for (int i = 0; i < size; i++)  {  free(prof[i]);  }  }  //int copstr(struct profecional \* profs[],int i ,int k);  void checkcopy()  {  int size = 3;  struct profecional \* prof[size];  for (int i = 0; i < size; i++)  {  prof[i] = malloc(sizeof(struct profecional));  }  prof[0] = NULL;  prof[1]->salary = 1000;  prof[1]->score = 2.5;  prof[1]->information.year = 2012;  prof[2]->salary = 5698;  prof[2]->score = 8.6;  prof[2]->information.year = 2010;  assert(copstr(prof,1,2)->salary == 1000);  assert(copstr(prof,1,2)->score == 2.5);  assert(copstr(prof,1,2)->information.year == 2012);  assert(copstr(prof,0,2) == NULL);  assert(copstr(prof,0,2) == NULL);  for (int i = 0; i < size; i++)  {  free(prof[i]);  }  } |

menu.h

|  |
| --- |
| #ifndef MENU\_h  void line();  void printfirstmenu();  void clearmainmenu();  void printsecondmenu();  void clearstringtostruct();  void clear();  void rewhelp(int i, int j);  void clearfield();  void printthirdmenu();  void cleanall();  void cleanstr(int i, int j);  void printdots(int k);  #endif //MENU\_h |

assert.h

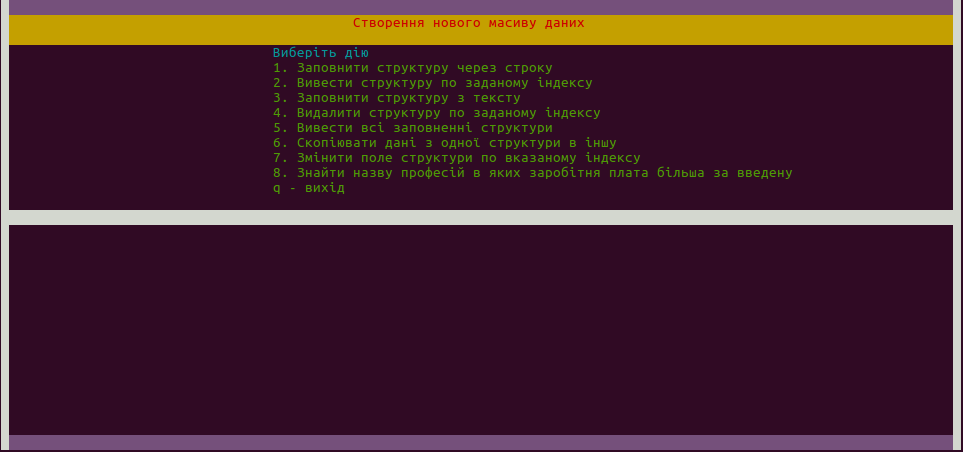
|  |
| --- |
| #ifndef LAB\_3TEST  #include "function.h"  void validtesting();  void validtestfloatin();  void check ();  void filetest();  void validdel1();  void checkdelete();  void checkstruct2();  void checkcopy();  #endif //LAB\_3TEST |

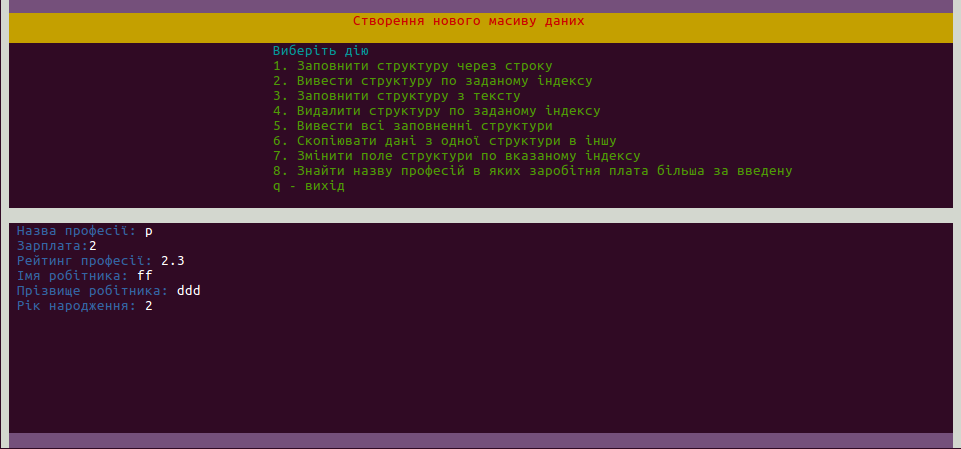
function.h

|  |
| --- |
| #ifndef LAB\_3  #define LAB\_3  struct profecional{  char profname[100];  int salary;  float score;  struct {  char name[100];  char surname[100];  int year;  } information;  };  void cat();  void stringtostruct( struct profecional \* profs[]);  int enterstuct(void);  void texttostruct(struct profecional \* profs[]);  void structtostring( struct profecional \* profs[]);  void structtotext(struct profecional \* profs[]);  //void deletestruct(struct profecional \* profs[]);  void copystruct (struct profecional \* profs[]);  void rewritefiealdinstruct(struct profecional \* profs[]);  int fileread(const char \* readFileName, struct profecional \*profs[]);  int filewrite(struct profecional \*profs[]);  void highestsalary(struct profecional \*profs[]);  char filename(char str[100]);  //void deleteallstruct(struct profecional \*profs[]);  void printstructs(struct profecional \* profs[] ,int i, char \*str);  int filecheck(const char \* readFileName);  int validtestfloat(char \*str);  int validtest( char \*str);  int checksalary(struct profecional \*profs[],int i, int midsalary);  int deltstruct1(struct profecional \* profs[],int i);  void deletestruct(struct profecional \* profs[]);  int scanint(char \* str);  int checkstruct(struct profecional \* profs[],int i);  struct profecional \* copstr(struct profecional \* profs[],int i ,int k);  #endif //LAB\_3 |

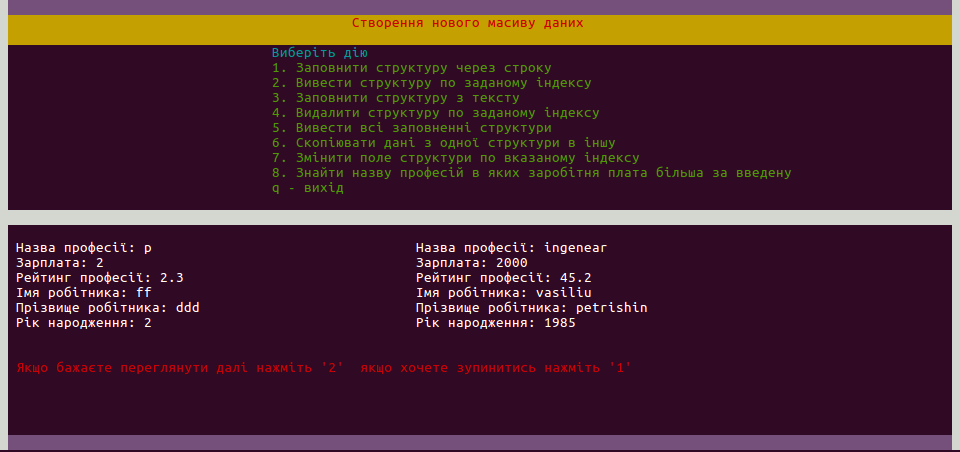
**Приклади результатів**

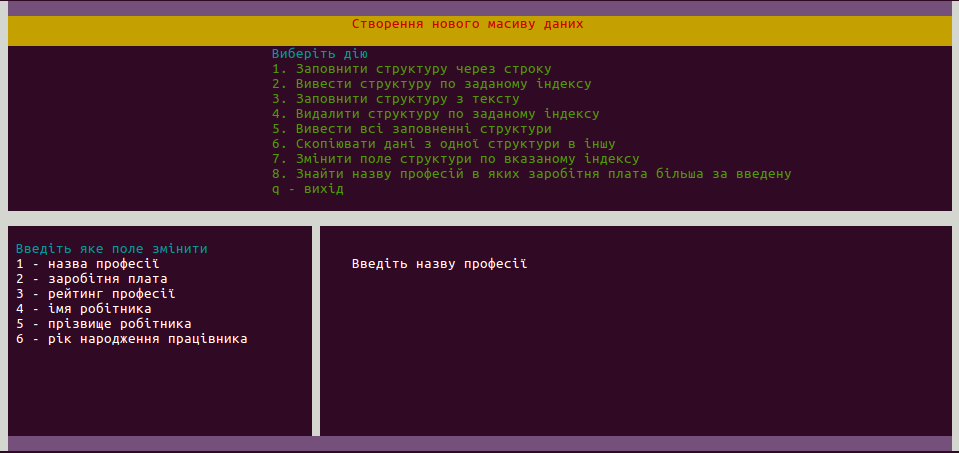


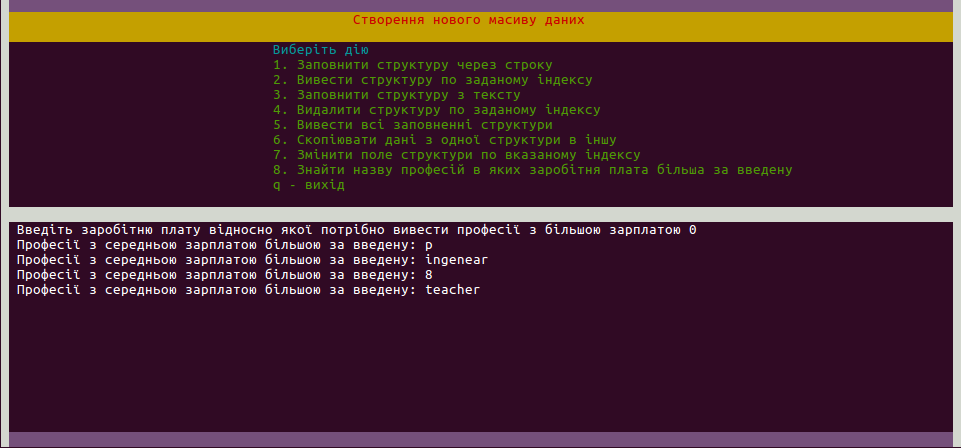












**Висновки**

Ми навчилися розбивати програму на файли і на функції. Використовували динамічне виділення памяті. Також ми вдосконалили свої вміння роботи з вказівниками. В цій лабораторній ми використовували структури.