

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 “**Структури даних, функції, вказівники і файлові потоки**”

|  |  |  |
| --- | --- | --- |
| Виконав  студент I курсу  групи КП-61  Телефус Ілля Анатолійович  (*прізвище, ім’я, по батькові*)  варіант № 27 |  | 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:   |  | | --- | |  | |

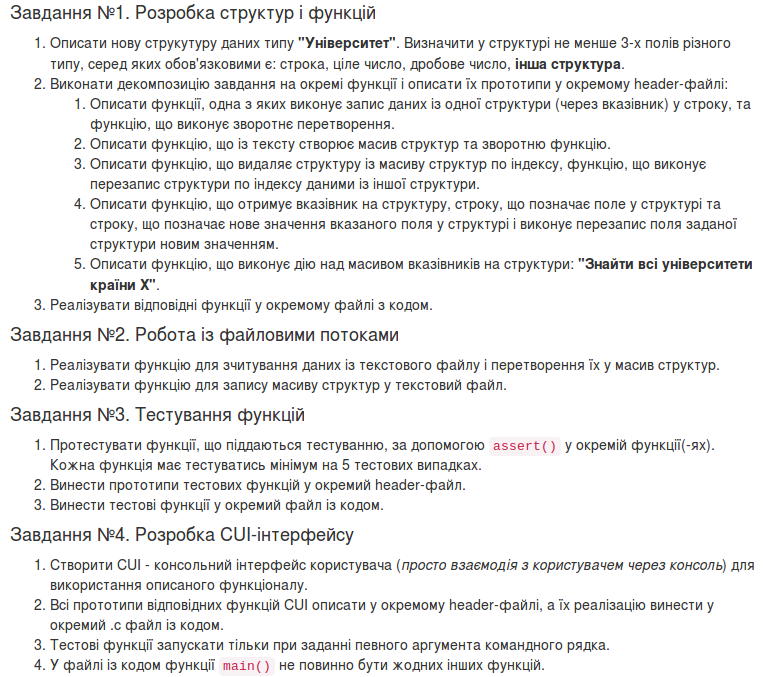
Київ 2016

**Мета роботи**

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

Навчитися оформлювати консольну програму для зручності роботи користувача.

**Постановка завдання**



**Тексти коду програм**

main.c

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <progbase.h>  #include <pbconsole.h>  #include "function.h"  #include <stdbool.h>  #include "menu.h"  #include <string.h>  #include "assert\_func.h"  int main(int argc, char \* argv[]){  struct university \* univers[1000] = {NULL};  char option;  int key\_push;  int key\_down;  char input\_country[100];  bool run = true;  bool nrun = true;  bool isRunning = true;  int i = 0;  for (i = 0; i < argc; i++) {  if(strcmp(argv[i], "test") == 0){  test\_delete\_struct(univers);  testfx(univers);  conClear();  test(univers);  conClear();  conMove(1, 1);  printf("assert test() is complete!\n");  conMove(2,1);  new\_year\_tree();  return EXIT\_SUCCESS;  }  }  while(isRunning==true){  system("clear");  fflush(stdin);  border\_row();  menu\_func2();  option = getchar();  clean();  switch(option) {  case '1':  run = true;  while (run == true){  menu\_func1();  fflush(stdin);  key\_down = conGetChar();  clean\_small();  switch(key\_down){  case '1':  write\_string\_to\_struct(univers);  break;  case '2':  print\_out(univers);  break;  case '3':  clean();  delete\_from\_struct(univers, input\_index());  break;  case '4':  clean();  copy\_struct(univers);  break;  case '5':  clean();  struct\_fields\_modify(univers);  break;  case '6':  clean();  input\_country\_additional(input\_country);  country\_search(univers, input\_country);  break;  case '7':  clean();  file\_process("read.txt", univers);  break;  case '8':  clean();  file\_write(univers);  break;  case 'q':  clean();  run = false;  break;  default:  clean();  conMove(26,2);  puts("try again only numbers 1 to 8");  conMove(27,2);  break;  }  }  break;  case '2':  nrun = true;  while (nrun == true){  file\_process("read.txt", univers);  menu\_func1();  fflush(stdin);  key\_push = conGetChar();  clean();  switch(key\_push){  case '1':  clean();  write\_string\_to\_struct(univers);  break;  case '2':  clean();  print\_out(univers);  break;  case '3':  clean();  delete\_from\_struct(univers, input\_index());  break;  case '4':  clean();  copy\_struct(univers);  break;  case '5':  clean();  struct\_fields\_modify(univers);  break;  case '6':  clean();  input\_country\_additional(input\_country);  country\_search(univers, input\_country);  break;  case '7':  clean();  file\_process("read.txt", univers);  break;  case '8':  clean();  file\_write(univers);  break;  case 'q':  nrun = false;  break;  default:  clean();  conMove(26,2);  puts("try again only numbers 1 to 8");  conMove(27,2);  break;  }  }  break;  case '\*':  testfx(univers);  conClear();  test(univers);  conClear();  conMove(1, 1);  printf("assert test() is complete!\n");  case 'z':  conMove(47,2);  isRunning = false;  conClear();  return 0;  default :  conMove(26,2);  puts("Try again to enter 1 or 2.");  break;  }  }  for(int k = 0; k < 1000; k++){  free(univers[k]);  }  return 0;  } |

menu.h

|  |
| --- |
| #include <stdio.h>  #include <progbase.h>  #include <pbconsole.h>  void border\_row(){  int row = 0;  int col = 0;  conClear();  conSetAttr(BG\_WHITE);  for (row = 0; row < 120; row++) {  conMove(1, row);  puts(" ");  conMove(25, row);  puts(" ");  conMove(45, row);  puts(" ");  }  for(col = 0; col < 46; col++){  conMove(col, 1);  puts(" ");  conMove(col, 120);  puts(" ");  }  conMove(1, 50);  conSetAttr(FG\_BLACK);  puts("Welcome to Ilya`s Program");  conReset();  }  void menu\_func1(){  conMove(2,2);  puts("1.Press the [1] in order to write string to struct.");  conMove(3,2);  puts("2.Press the [2] in order to write struct to string and output it.");  conMove(4,2);  puts("3.Press the [3] in order to delete struct.");  conMove(5,2);  puts("4.Press the [4] in order to change data between the structs (by index).");  conMove(6,2);  puts("5.Press the [5] in order to change data in chosen struct and field.");  conMove(7,2);  puts("6.Press the [6] in order to find university of entered country.");  conMove(8,2);  puts("7.Press the [7] in order to read from file.");  conMove(9,2);  puts("8.Press the [8] in order to write to file.");  conMove(10,2);  puts("8.Press the [q] in order to to escape to main menu.");  conMove(26,2);  }  void menu\_func2(){  conMove(2,2);  puts("1.Press the [1] to fill the struct manually.");  conMove(3,2);  puts("2.Press the [2] to read data for the struct from file.");  conMove(4,2);  puts("3.Press [z] to quit the programm.");  conMove(5,2);  puts("$Press [\*] to run assert.$");  conMove(26,2);  }  void clean() {  int x = 0;  int y = 0;  for (x = 26; x < 45; x++) {  for (y = 2; y < 120; y++) {  conMove(x, y);  puts(" ");  }  }  }  void clean\_small(){  int x = 0;  int y = 0;  for (x = 27; x < 45; x++) {  for (y = 2; y < 120; y++) {  conMove(x, y);  puts(" ");  }  }  } |

Function.c

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <stdbool.h>  #include <pbconsole.h>  #include <ctype.h>  #include <string.h>  #include "function.h"  #include "menu.h"  #define VAR 1000  char scan\_str(char \*str) {  fgets(str, 200, stdin);  return \*str;  }  char scan\_str\_for5(char \*str) {  scanf("%s",str);  return \*str;  }  int file\_pointer\_for\_fw(){  char file\_out[30];  conMove(26,2);  printf("Enter name of file where you wanna write the struct");  conMove(27,2);  scanf("%s", file\_out);  clean();  return \*file\_out;  }  int file\_write(struct university \* ptr[]){  char file\_out[100];  conMove(26,2);  printf("Enter name of file where you wanna write the struct");  conMove(27,2);  scanf("%s", file\_out);  clean();  conMove(26,2);  char \*write = file\_out;  FILE \*fout = fopen(write, "w");  if (file\_out == NULL ){  puts("ERROR, NO FILE");  return 1;  }  for(int i = 0; i < VAR; i++){  if(ptr[i] != NULL){  fprintf(fout,"Full name: %s, Location: %s, Success level: %f, Rank: %i, Students:%i, Matan p: %i\n", ptr[i]->full\_name, ptr[i]->location, ptr[i]->success\_level, ptr[i]->rank, ptr[i]->more\_data.students, ptr[i]->more\_data.matan\_points);  }  }  fclose(fout);  return 0;  }  int file\_process(const char \*file\_in, struct university \* ptr[]){  enum{BUFFER\_SIZE = 100};  char buffer[BUFFER\_SIZE];  int i = 0;  FILE \*fin = fopen(file\_in, "r");  if (fin == NULL ){  conMove(27,2);  return 1;  }  while (fgets(buffer, BUFFER\_SIZE, fin) != NULL) {  buffer[strlen(buffer) - 1] = '\0';  for(i = 0; i < VAR; i++){  if(ptr[i] == NULL){  ptr[i] = (struct university\*)malloc(sizeof(struct university));  break;  }  }  sscanf(buffer,"%s %s %3f %i %i %i",ptr[i]->full\_name, ptr[i]->location, &ptr[i]->success\_level, &ptr[i]->rank, &ptr[i]->more\_data.students, &ptr[i]->more\_data.matan\_points);  }  fclose(fin);  return 0;  }  void write\_string\_to\_struct(struct university \* ptr[]) {  char buffer[200];  int i;  char key = '1';  for (i = 0; i < VAR; i++){  if(ptr[i] == NULL){  while(key == '1'){  conMove(27,2);  scan\_str(buffer);  ptr[i] = (struct university\*)malloc(sizeof(struct university));  sscanf(buffer,"%s %s %3f %i %i %i",ptr[i]->full\_name, ptr[i]->location, &ptr[i]->success\_level, &ptr[i]->rank, &ptr[i]->more\_data.students, &ptr[i]->more\_data.matan\_points);  conMove(26,2);  printf("Type in 1 to continue writing or 'z' to stop entering anything\n");  clean\_small();  conMove(27,2);  key = conGetChar();  if( key == 'z' ){  clean();  break;  }  else if( key != '1'){  clean();  conMove(26,2);  puts("Type in only 1 or z, cat cries.");  conMove(26,2);  }  }  break;  }  }  }  void print\_out(struct university \* ptr[]) {  int i = 0;  int k = 0;  char key = 1;  while(key != '1'){  for(i = 0; i < VAR; i++){  if(ptr[i] != NULL){  conMove(26 + k, 2);  printf("Full name: %s, Location: %s, Success level: %f, Rank: %i, Students:%i, Matan p: %i", ptr[i]->full\_name, ptr[i]->location, ptr[i]->success\_level, ptr[i]->rank, ptr[i]->more\_data.students, ptr[i]->more\_data.matan\_points);  conMove(27 + k,2);  puts("press z to exit this function");  conMove(28 + k,2);  k++;  key = conGetChar();  if(key == 'z'){  clean();  break;  }  else if( key != '2'){  clean();  conMove(26,2);  puts("Type in only 1 or z, cat cries.");  conMove(27,2);  }  }  }  break;  }  }  int input\_index (){  int input;  bool is\_out = true;  while (is\_out == true){  conMove(26,2);  printf("Type in index of a struct to delete.\n");  conMove(27,2);  scanf("%i", &input);  clean\_small();  conMove(26,2);  printf("If you wanna delete more structs press 1, to stop press 2.\n");  conMove(27,2);  int quit = conGetChar();  if(quit != 1){  clean();  break;  }  }  return input;  }  struct university \* delete\_from\_struct(struct university \* ptr[], int input){  struct university \* pntr = ptr[input];  ptr[input] = NULL;  return pntr;  free(ptr[input]);  }  void copy\_struct(struct university \* ptr[]){  bool is\_out = true;  int input;  int output;  while (is\_out == true){  clean();  conMove(26,2);  printf("Type in index of a struct to scan data from.\n");  conMove(27,2);  scanf("%i", &input);  clean();  conMove(26,2);  printf("Type in index of a struct to input data in.\n");  conMove(27,2);  scanf("%i", &output);  clean\_small();  if(ptr[input] == NULL){  clean();  conMove(29,45);  printf("There is no such struct!\n");  conMove(27,2);  }  if(ptr[output] == NULL){  ptr[input] = (struct university\*)malloc(sizeof(struct university));  }  strcpy(ptr[input]->location, ptr[output]->location);  strcpy(ptr[input]->full\_name, ptr[output]->full\_name);  ptr[input]->success\_level = ptr[output]->success\_level;  ptr[input]->rank = ptr[output]->rank;  ptr[input]->more\_data.students = ptr[output]->more\_data.students;  ptr[input]->more\_data.matan\_points = ptr[output]->more\_data.matan\_points;  conMove(26,2);  printf("If you wanna continue press 1, to stop press 2.\n");  conMove(27,2);  int quit = conGetChar();  if(quit != 1){  clean();  break;  }  }  }  void struct\_fields\_modify(struct university \* ptr[]){  bool is\_out = true;  int struct\_n = 0;  int option;  char buffer[200];  float success\_level;  int rank;  int students;  int matan\_points;  conMove(26,2);  printf("type in number of struct you wanna modify\n");  conMove(27,2);  scanf("%i", &struct\_n);  clean\_small();  if(ptr[struct\_n] == NULL && isdigit(struct\_n)){  ptr[struct\_n] = (struct university\*)malloc(sizeof(struct university));  }  is\_out = true;  while (is\_out == true){  clean();  conMove(26,2);  puts("Type in number of field you wanna change from 1-6 or e to exit");  conMove(27,2);  option = conGetChar();  switch (option) {  case '1':  clean();  conMove(26,2);  puts("Type in university`s abreviation");  conMove(27,2);  scan\_str\_for5(buffer);  sscanf(buffer,"%s",ptr[struct\_n]->full\_name);  clean();  break;  case '2':  clean();  conMove(26,2);  puts("Type in university`s location");  conMove(27,2);  scan\_str\_for5(buffer);  sscanf(buffer,"%s",ptr[struct\_n]->location);  clean();  break;  case '3':  clean();  conMove(26,2);  puts("Type in university`s success level");  conMove(27,2);  scanf("%f", &success\_level);  if(isalpha(success\_level) !=0 && ispunct(success\_level) != 0){  conMove(27,2);  puts("It has to be a number");  }  else{  ptr[struct\_n]->success\_level = success\_level;  clean();  }  break;  case '4':  clean();  conMove(26,2);  puts("Type in university`s rank");  conMove(27,2);  scanf("%i", &rank);  if(isalpha(rank) !=0 && ispunct(rank) != 0){  conMove(27,2);  puts("It has to be a number");  conMove(28,2);  }  else{  ptr[struct\_n]->rank = rank;  clean();  }  break;  case '5':  clean();  conMove(26,2);  puts("Type in number of students");  conMove(27,2);  scanf("%i", &students);  if(isalpha(students) !=0 && ispunct(students) != 0){  conMove(27,2);  puts("It has to be a number");  }  else{  ptr[struct\_n]->more\_data.students = students;  clean();  }  break;  case '6':  clean();  conMove(26,2);  puts("Type in students` matan points");  conMove(27,2);  scanf("%i", &matan\_points);  if(isalpha(matan\_points) !=0 && ispunct(matan\_points) != 0){  conMove(27,2);  puts("It has to be a number");  }  else{  ptr[struct\_n]->more\_data.matan\_points = matan\_points;  clean();  }  break;  default:  conMove(29,45);  puts("There is no field with such number");  break;  case 'e':  is\_out = false;  clean();  break;  }  break;  }  }  char \* input\_country\_additional(char input\_country[]){  conMove(26,2);  puts("Type in a country university of which you wanna find.");  conMove(27,2);  fgets(input\_country, 100, stdin);  scanf("%s", input\_country);  clean();  return input\_country;  }  int country\_search(struct university \* ptr[], char input\_country[]){  int flag = 0;  for (int i = 0; i < VAR; i++){  if(ptr[i] != NULL){  if(strcmp(input\_country, ptr[i]->location) == 0){  conMove(26 + i,2);  printf("Full name: %s, Location: %s, Success level: %f, Rank: %i, Students:%i, Matan p: %i", ptr[i]->full\_name, ptr[i]->location, ptr[i]->success\_level, ptr[i]->rank, ptr[i]->more\_data.students, ptr[i]->more\_data.matan\_points);  flag = 1;  }  }  }  if (flag == 0) {  conMove(30, 2);  puts("There is no universities from such country.");  }  return 0;  }  void new\_year\_tree(){  int level;  int levelcount = 5;  int layercount = 5;  int layer;  int star;  int starcount;  int spacecount;  int space;  int length;  printf("\n Chrismas Tree \n");  conSetAttr(FG\_RED);  length = (layercount\*levelcount);  starcount = 1;  spacecount = length;  for (layer = 1 ; layer <= layercount ; layer++) {  for (level = 1 ; level <= levelcount ; level++) {  for (space = 1 ; space <= spacecount ; space++)  printf(" ");  for (star = 1 ; star <= starcount ; star++)  printf("\*");  printf("\n");  starcount += 2;  spacecount--;  }  starcount -= 2;  spacecount++;  if(levelcount <= 3){  starcount -= 2;  spacecount += 1;  }  else{  starcount -= 4;  spacecount += 2;  }  }  spacecount = length;  for (layer = 1 ; layer <= 4; layer++) {  for (space = 1 ; space <= spacecount-1 ; space++)  printf(" ");  for (star = 1 ; star <= 3 ; star++)  printf("#");  printf("\n");  }  }; |

Assert\_func.c

|  |
| --- |
| #include <assert.h>  #include"assert\_func.h"  #include "function.h"  #include <stdio.h>  #include <stdlib.h>  #include <math.h>  void test (struct university \* univers[]){  assert( file\_process (" ", univers) == 1);  assert( file\_process ("read.txt",univers ) == 0);  assert( file\_process ("\*read.exe\*", univers) == 1);  }  void testfx (struct university \* univers[]){  assert( file\_write (univers) == 0);  } |

**Assert\_func.h**

|  |
| --- |
| #ifndef ASSERT\_FUNK\_H\_INCLUDED  #define ASSERT\_FUNK\_H\_INCLUDED  #include "function.h"  void testfx (struct university \* univers[]);  void test (struct university \* univers[]);  #endif |

**Function.h**

|  |
| --- |
| #ifndef FUNCTION\_H\_INCLUDED  #define FUNCTION\_H\_INCLUDED  struct university{  char full\_name[150];  char location[50];  float success\_level;  int rank;  struct {  int students;  int matan\_points;  } more\_data;  };  int input\_index();  int file\_pointer\_for\_fw();  int file\_write(struct university \* ptr[]); //ready  int file\_process(const char \*file\_in, struct university \* ptr[]); //ready  char \* input\_country\_additional(char input\_country[]);  int country\_search(struct university \* ptr[], char input\_country[]);  void struct\_fields\_modify(struct university \* ptr[]);  void copy\_struct(struct university \* ptr[]);  struct university \*delete\_from\_struct(struct university \* ptr[], int input); //ready  void write\_string\_to\_struct(struct university \* ptr[]);  void print\_out(struct university \* ptr[]);  char scan\_str(char \*str);  char scan\_str\_for5(char \*str);  void new\_year\_tree();  #endif |

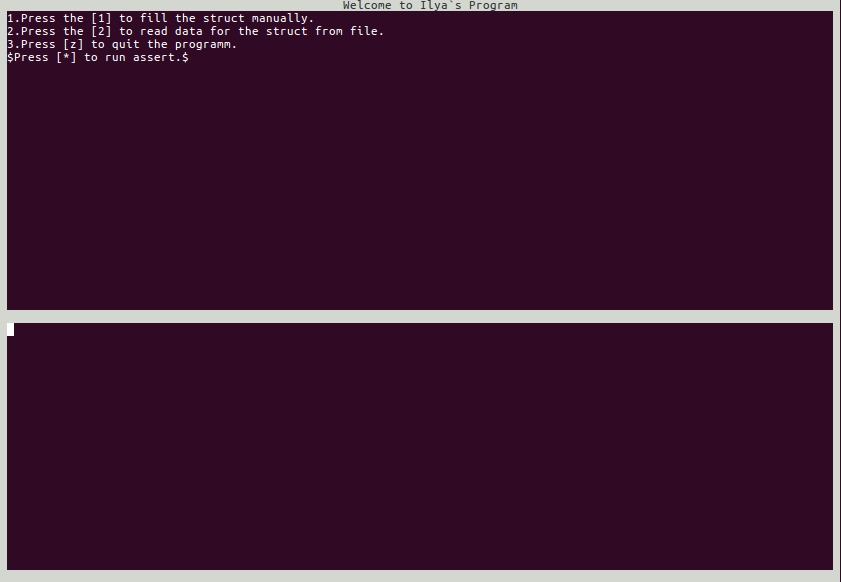
Menu.h

|  |
| --- |
| #ifndef MENU\_H\_INCLUDED  #define MENU\_H\_INCLUDED  void border\_row();  void menu\_func1();  void menu\_func2();  void clean();  void clean\_small();  #endif |

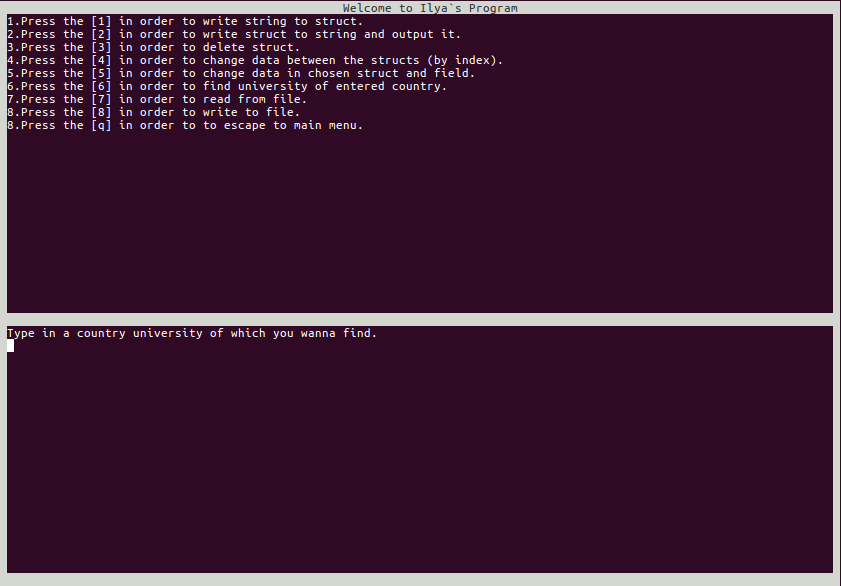
Makefile

|  |
| --- |
| # Makefile for LAB\_3 project  LAB\_3: main.o function.o menu.o assert\_func.o  gcc -o LAB\_3 main.o function.o menu.o assert\_func.o -lprogbase  main.o: main.c  gcc -c main.c  function.o: function.c  gcc -c function.c  menu.o: menu.c  gcc -c menu.c  assert\_func.o: assert\_func.c  gcc -c assert\_func.c  clean:  rm -f \*.o LAB\_3 |

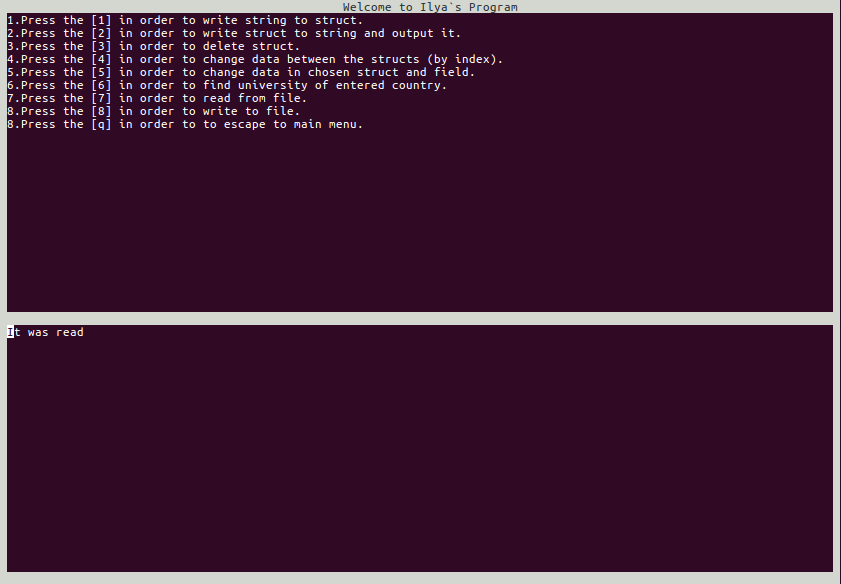
**Приклади результатів**

**Меню.**

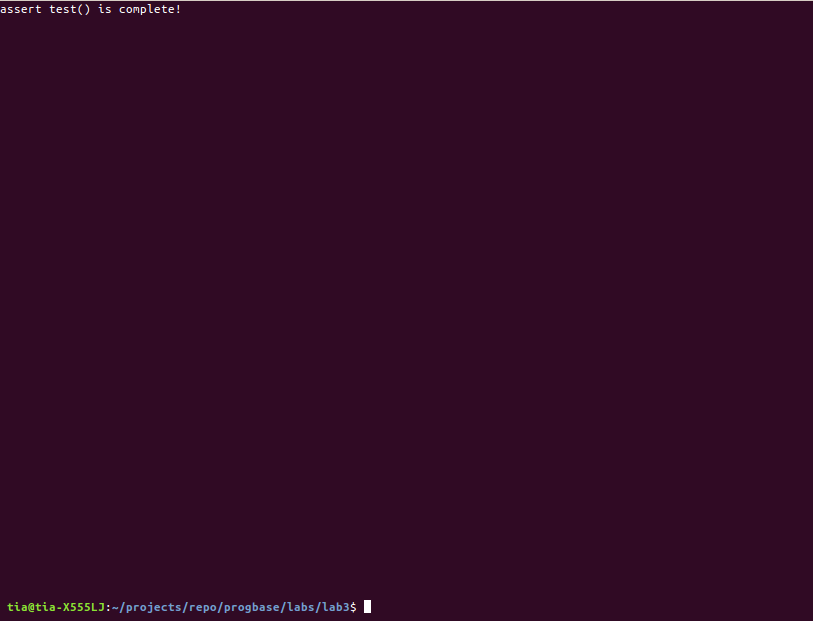
**Завдання 1.**

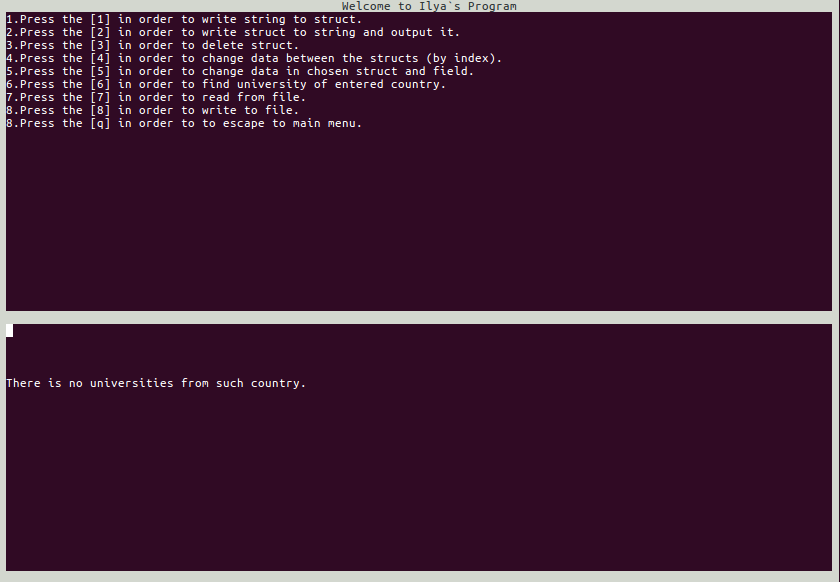


**Завдання 2.**



**Завдання 3.**







**Висновки**

Виконавши дану лабораторну роботу ми навчилися працювати з масивами та строками в мові програмування С.

Застосувати на практиці різні методи опрацювання структур вказівники на структури.

Вдосконалили вміння роботи з рядками у мові С.

Навчилися працювати з структурами та зрозуміли всі їх переваги.

Створили консольну програму з доволі зручним інтерфейсом.

В цілому, мета досягнена.