

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  Дзенік Данило Миколайович  (*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:   |  | | --- | |  | |

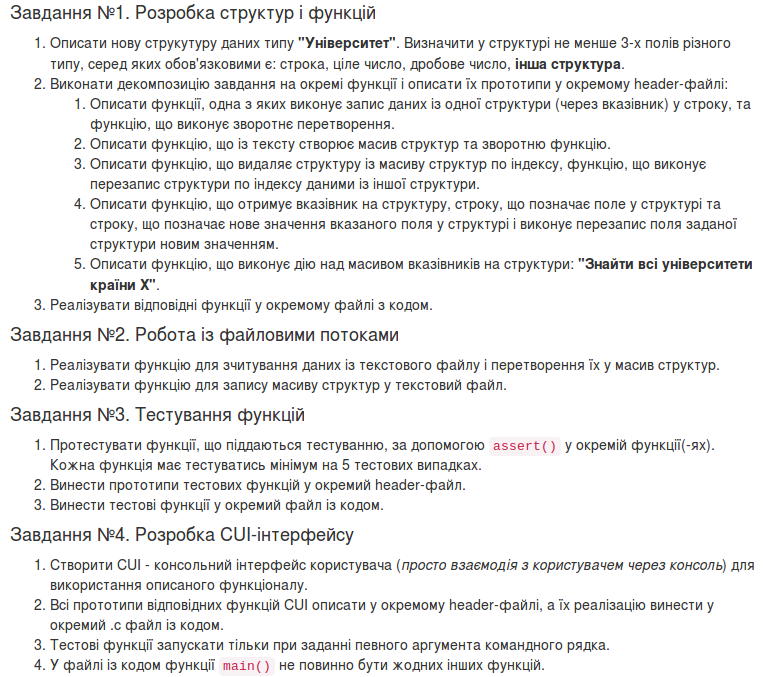
Київ 2016

**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 <string.h>  #include <ctype.h>  #include "file.h"  #include "function.h"  #include "menu.h"  #include "assert.h"  #define STR\_SIZE 100  //gcc main.c menu.c function.c file.c assert.c -std=c99 -Wall -Werror -pedantic-errors -lm -lprogbase  int main(int argc, char const \*argv[]) {  if (!strcmp(argv[1], "test") && argc == 2) {  test();  }  conHideCursor();  conResize(40,100);  char n = '\0';  while (n != 'q') {  conHideCursor();  conResize(40,100);  conClear();  parallel\_Lines();  line(1);  stand\_out("Main",2, 48, FG\_INTENSITY\_CYAN);  line(3);  stand\_out("1) Create a new data set", 10, 37, FG\_INTENSITY\_YELLOW);  line(18);  line(22);  stand\_out("2) Read array of data from file", 30, 34, FG\_INTENSITY\_YELLOW);  line(39);  n = conGetChar();  switch (n) {  case '1': {  stand\_out("Input the name of Internet Service Provider: ", 20, 4, FG\_DEFAULT);  conShowCursor();  char filename[STR\_SIZE] = "";  fgets(filename, STR\_SIZE, stdin);  filename[strlen(filename) - 1] = '\0';  clear\_line(20, 1);  FILE \* file = fopen(filename, "a");  if (NULL == file) {  conHideCursor();  stand\_out("!!!ERROR OPENNING FILE!!!", 19, 37, FG\_INTENSITY\_RED);  stand\_out("Enter some button to exit from programm", 21, 30, FG\_INTENSITY\_RED);  conGetChar();  conClear();  return EXIT\_FAILURE;  }  struct Provider \* users = (struct Provider \*)malloc(sizeof(struct Provider));  if (NULL == users) {  stand\_out("!!!ERROR TO CREATE NEW STRUCTURE!!!", 19, 37, FG\_INTENSITY\_RED);  stand\_out("Enter some button to exit from programm", 21, 30, FG\_INTENSITY\_RED);  n = conGetChar();  conClear();  return EXIT\_FAILURE;  }  clear\_line(2, 1);  stand\_out(filename, 2, 50 - strlen(filename) / 2, FG\_INTENSITY\_CYAN);  clear\_line(4, 34);  int NumberOfUsers = GetStructuresFromConsole(users);  if (NumberOfUsers != 0) {  RecordStructure(file, users, &NumberOfUsers);  }  free(users);  fclose(file);  break;  }  case '2': {  int check = -2;  struct Provider \* Users = (struct Provider \*)malloc(1 \* sizeof(struct Provider));  if (NULL == Users) {  stand\_out("!!!ERROR TO CREATE NEW STRUCTURE!!!", 19, 37, FG\_INTENSITY\_RED);  stand\_out("Enter some button to exit from programm", 21, 30, FG\_INTENSITY\_RED);  n = conGetChar();  conClear();  return EXIT\_FAILURE;  }  char filename[STR\_SIZE] = "\0";  while (check == 0 || check == -2) {  if (check == -2) {  stand\_out("Input the name of Internet Service Provider: ", 20, 4, FG\_DEFAULT);  }else {  stand\_out("Input the name of Internet Service Provider: ", 20, 4, FG\_INTENSITY\_RED);  }  conShowCursor();  fgets(filename, STR\_SIZE, stdin);  filename[strlen(filename) - 1] = '\0';  clear\_line(20, 1);  FILE \* file = fopen(filename, "r");  if (NULL == file) {  conHideCursor();  stand\_out("!!!ERROR OPENNING FILE!!!", 19, 37, FG\_INTENSITY\_RED);  stand\_out("Enter some button to exit from programm", 21, 30, FG\_INTENSITY\_RED);  n = conGetChar();  conClear();  return EXIT\_FAILURE;  }  Users = GetStructuresFromFile(file, Users, &check);  clear\_line(20, 1);  fclose(file);  }  stand\_out(filename, 2, 50 - strlen(filename) / 2, FG\_INTENSITY\_CYAN);  n = '\0';  int eror = 0;  while (n != 'r') {  clear\_line(4, 32);  PrintUsers(Users, &check);  stand\_out("'1' Delete specified User", 7, 66, FG\_YELLOW);  stand\_out("'2' Rewrite data in specified position", 13, 59, FG\_YELLOW);  stand\_out("'3' Rewrite specified User", 19, 65, FG\_YELLOW);  stand\_out("'4' Find 'N' users with the best Speed ", 25, 59, FG\_YELLOW);  stand\_out("'s' Save changes", 31, 71, FG\_YELLOW);  stand\_out("'r' Return to Main Menu", 37, 67, FG\_YELLOW);  conHideCursor();  n = conGetChar();  switch (n) {  case '1': {  int index = -1;  while( index == 0 || index == -1) {  clear\_field(10, 69, 30);  if (index == -1) {  stand\_out("User :", 10, 72, FG\_GREEN);  }else {  stand\_out("User :", 10, 72, FG\_INTENSITY\_RED);  }  if (eror == 0) {  char buffer[STR\_SIZE] = "\0";  fgets(buffer, STR\_SIZE, stdin);  eror = 1;  }  char buffer[STR\_SIZE] = "\0";  conMove(10, 80);  fgets(buffer, STR\_SIZE, stdin);  buffer[strlen(buffer) - 1] = '\0';  index = GetIndex(buffer);  if (index > check) {  index = 0;  }  }  Users = DeleteStructure(Users, &check, index);  break;  }  case '2': {  int index = -1;  while( index == 0 || index == -1) {  clear\_field(16, 57, 42);  if (index == -1) {  stand\_out("Index :", 16, 72, FG\_GREEN);  }else {  stand\_out("Index :", 16, 72, FG\_INTENSITY\_RED);  }  if (eror == 0) {  char buffer[STR\_SIZE] = "\0";  fgets(buffer, STR\_SIZE, stdin);  eror = 1;  }  char buffer[STR\_SIZE] = "\0";  conMove(16, 80);  fgets(buffer, STR\_SIZE, stdin);  buffer[strlen(buffer) - 1] = '\0';  index = GetIndex(buffer);  if (index > check \* 4) {  index = 0;  }  }  Users = RewriteData(Users, index);  break;  }  case '3': {  int index = -1;  while( index == 0 || index == -1) {  clear\_field(10, 69, 30);  if (index == -1) {  stand\_out("User :", 22, 72, FG\_GREEN);  }else {  stand\_out("User :", 22, 72, FG\_INTENSITY\_RED);  }  if (eror == 0) {  char buffer[STR\_SIZE] = "\0";  fgets(buffer, STR\_SIZE, stdin);  eror = 1;  }  char buffer[STR\_SIZE] = "\0";  conMove(22, 80);  fgets(buffer, STR\_SIZE, stdin);  buffer[strlen(buffer) - 1] = '\0';  index = GetIndex(buffer);  if (index > check) {  index = 0;  }  }  Users = GetStructure(Users, index, 1);  break;  }  case '4': {  int number = -1;  while( number == 0 || number == -1) {  clear\_field(28, 57, 42);  if (number == -1) {  stand\_out("Number :", 28, 72, FG\_GREEN);  }else {  stand\_out("Number :", 28, 72, FG\_INTENSITY\_RED);  }  if (eror == 0) {  char buffer[STR\_SIZE] = "\0";  fgets(buffer, STR\_SIZE, stdin);  eror = 1;  }  char buffer[STR\_SIZE] = "\0";  conMove(28, 81);  fgets(buffer, STR\_SIZE, stdin);  buffer[strlen(buffer) - 1] = '\0';  number = GetIndex(buffer);  if (number > check ) {  number = 0;  }  }  clear\_line(4, 35);  GetNhigherSpeed(Users, number, check);  conGetChar();  break;  }  case 's': {  remove(filename);  FILE \* save = fopen(filename, "w");  if (NULL == save) {  conHideCursor();  stand\_out("!!!ERROR OPENNING FILE!!!", 34, 60, FG\_INTENSITY\_RED);  stand\_out("Enter some button to exit from programm", 34, 57, FG\_INTENSITY\_RED);  n = conGetChar();  conClear();  return EXIT\_FAILURE;  }  RecordStructure(save, Users, &check);  fclose(save);  break;  }  case 'q': {  conClear();  free(Users);  return 0;  break;  }  }  }  free(Users);  break;  }  case 'q': {  return EXIT\_SUCCESS;  }  }  }  } |

menu.h

|  |
| --- |
| #ifndef MENU\_H  #define MENU\_H  void line(int x);  void clear\_line(int line, int numbers);  void parallel\_Lines(void);  void stand\_out(char str[],int x, int y, int attr);  void clear\_field(int x, int startY, int numbers);  #endif |

Function.c

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include "function.h"  #include <pbconsole.h>  #include "menu.h"  #include <ctype.h>  #include <string.h>  int GetStructuresFromConsole(struct Provider \* users) {  int count = 1;  char n = 0;  char space = 0;  while (n != '2' && n != '3') {  users = (struct Provider \*)realloc(users, (size\_t)count \* sizeof(struct Provider));  GetStructure(users, count, space);  space = 1;  clear\_line(4, 34);  stand\_out("1) Add New User", 10, 27, FG\_INTENSITY\_YELLOW);  stand\_out("2) Return to Main menu and save Users in file ", 20, 27, FG\_INTENSITY\_YELLOW);  stand\_out("3) Return to Main menu without saving", 30, 27, FG\_INTENSITY\_YELLOW);  n = conGetChar();  count++;  if (n == '3') {  count = 0;  }else if (n == '2') {  count--;  }  }  return count;  }  char \* GetName(char string[]) {  char \* str = NULL;  str = string;  if (!strcmp(string, "")) {  str = "0";  return str;  }  for (int i = 0; \*(str + i) != '\0'; i++) {  if (!isalpha(\*(str + i))) {  str = "0";  return str;  }  }  return str;  }  int GetYear(char string[]) {  int Year = 0;  int i = 0;  if (!strcmp(string, "")) {  return 0;  }  while (string[i] != '\0') {  if (!isdigit(string[i])) {  return -1;  }  i++;  }  Year = atoi(string);  if (Year > 2017 || Year <= 1900) {  return -1;  }  return Year;  }  int GetSpeed(char string[]) {  int Speed = 0;  int i = 0;  int Mb[14] = {1, 2, 4, 8, 16, 32, 50, 64, 80, 100, 128, 256, 512, 1000};  while (string[i] != '\0') {  if (!isdigit(string[i])) {  return 0;  }  i++;  }  Speed = atoi(string);  for (int i = 0; i < 14; i++) {  if(Speed == Mb[i]) {  return Speed;  }  }  return 0;  }  int GetIndex(char string[]) {  int index = 0;  int i = 0;  if (!strcmp(string, "")) {  return 0;  }  while (string[i] != '\0') {  if (!isdigit(string[i])) {  return 0;  }  i++;  }  index = atoi(string);  if (index <= 0) {  return 0;  }  return index;  }  float GetMonthlyFee(char string[]) {  int i = 0;  float DoubNumber = 0.0;  while (string[i] != '\0') {  if (!isdigit(string[i]) && !(string[i] == '.' && i != 0 && i != strlen(string) && strlen(string) - i <= 3)) {  return DoubNumber;  }  i++;  }  DoubNumber = atof(string);  return DoubNumber;  }  void PrintUsers(struct Provider \* Users, int \* size) {  int shiftX = 0;  conSetAttr(FG\_GREEN);  for (int i = 2; i < 37; i++) {  conMove(2 + i, 56);  printf("|");  }  conSetAttr(FG\_DEFAULT);  for (int i = 0; i < \*size; i += 2) {  conMove(5 + shiftX, 10);  printf("User №%i", i + 1);  conMove(6 + shiftX, 3);  printf("%i) Name: %s", i \* 4 + 1, Users[i].Name);  conMove(7 + shiftX, 3);  if ( Users[i].Year != 0) {  printf("%i) Year of birth: %i", i \* 4 + 2, Users[i].Year);  }else {  printf("%i) Year of birth: ", i \* 4 + 2);  }  conMove(8 + shiftX, 3);  printf("%i) Speed: %i", i \* 4 + 3, Users[i].Speed);  conMove(9 + shiftX, 3);  printf("%i) Monthly Fee: %g", i \* 4 + 4, Users[i].MonthlyFee);  shiftX += 6;  }  shiftX = 0;  for (int i = 1; i < \*size; i += 2) {  conMove(5 + shiftX, 37);  printf("User №%i", i + 1);  conMove(6 + shiftX, 30);  printf("%i) Name: %s", i \* 4 + 1, Users[i].Name);  conMove(7 + shiftX, 30);  if ( Users[i].Year != 0) {  printf("%i) Year of birth: %i", i \* 4 + 2, Users[i].Year);  }else {  printf("%i) Year of birth: ", i \* 4 + 2);  }  conMove(8 + shiftX, 30);  printf("%i) Speed: %i", i \* 4 + 3, Users[i].Speed);  conMove(9 + shiftX, 30);  printf("%i) Monthly Fee: %g", i \* 4 + 4, Users[i].MonthlyFee);  shiftX += 6;  }  return;  }  struct Provider \* DeleteStructure(struct Provider \* Users, int \* size , int index) {  for (int i = index - 1; i < \*size - 1; i++) {  strcpy(Users[i].Name, Users[i + 1].Name);  Users[i].Year = Users[i + 1].Year;  Users[i].Speed = Users[i + 1].Speed;  Users[i].MonthlyFee = Users[i + 1].MonthlyFee;  }  \*size = \*size - 1;  Users = (struct Provider \*)realloc(Users, \*size \* sizeof(struct Provider));  return Users;  }  struct Provider \* RewriteData(struct Provider \* Users, int index) {  switch (index % 4) {  case 0: {  index /= 4;  Users[index].MonthlyFee = -1;  while(Users[index].MonthlyFee == 0.0 || Users[index].MonthlyFee == -1) {  clear\_field(16, 58, 42);  if (Users[index].MonthlyFee == -1) {  stand\_out("Monthly fee(UAN) :", 16, 62, FG\_GREEN);  }else {  stand\_out("Monthly fee(UAN) :", 16, 62, FG\_INTENSITY\_RED);  }  char buffer[STR\_SIZE] = "\0";  conMove(16, 82);  fgets(buffer, STR\_SIZE, stdin);  buffer[strlen(buffer) - 1] = '\0';  Users[index].MonthlyFee = GetMonthlyFee(buffer);  }  break;  }  case 1: {  index /= 4;  strcpy(Users[index].Name, "-1");  while( !strcmp(Users[index].Name, "0") || !strcmp(Users[index].Name, "-1")) {  clear\_field(16, 58, 42);  if (!strcmp(Users[index].Name, "-1")) {  stand\_out("Names :", 16, 70, FG\_GREEN);  }else {  stand\_out("Names :", 16, 70, FG\_INTENSITY\_RED);  }  char buffer[STR\_SIZE] = "\0";  conMove(16, 79);  fgets(buffer, STR\_SIZE, stdin);  buffer[strlen(buffer) - 1] = '\0';  strcpy(Users[index].Name, GetName(buffer));  }  break;  }  case 2: {  index /= 4;  Users[index].Year = -2;  while(Users[index].Year == -1 || Users[index].Year == -2) {  clear\_field(16, 58, 42);  if (Users[index].Year == -2) {  stand\_out("Year of birth:", 16, 62, FG\_GREEN);  }else {  stand\_out("Year of birth:", 16, 62, FG\_INTENSITY\_RED);  }  char buffer[STR\_SIZE] = "\0";  conMove(16, 78);  fgets(buffer, STR\_SIZE, stdin);  buffer[strlen(buffer) - 1] = '\0';  Users[index].Year = GetYear(buffer);  }  break;  }  case 3: {  index /= 4;  Users[index].Speed = -1;  while(Users[index].Speed == 0 || Users[index].Speed == -1) {  clear\_field(16, 58, 42);  if (Users[index].Speed == -1) {  stand\_out("Speed(Mb/c) :",16, 65, FG\_GREEN);  }else {  stand\_out("Speed(Mb/c) :",16, 65, FG\_INTENSITY\_RED);  }  char buffer[STR\_SIZE] = "\0";  conMove(16, 80);  fgets(buffer, STR\_SIZE, stdin);  buffer[strlen(buffer) - 1] = '\0';  Users[index].Speed = GetSpeed(buffer);  }  break;  }  }  return Users;  }  struct Provider \* GetStructure(struct Provider \* users, int count, int entrance) {  clear\_line(4,35);  conShowCursor();  conMove(6, 45);  conSetAttr(FG\_YELLOW);  printf("User №%i", count);  conSetAttr(FG\_DEFAULT);  line(24);  stand\_out(" - Fields which must be filled", 27, 12, FG\_YELLOW);  stand\_out("\*", 27, 12, FG\_INTENSITY\_RED);  stand\_out("Name - 0nly the alphabetic characters", 29, 12, FG\_YELLOW);  stand\_out("Year of birth - within [1900 - 2016] or '0' If you want to leave it empty ", 31, 12, FG\_YELLOW);  stand\_out("Speed(Mb/c) - Only {1, 2, 4, 8, 16, 32, 50, 64, 80, 100, 128, 256, 512, 1000}", 33, 12, FG\_YELLOW);  stand\_out("MonthlyFee(UAH) - Only positive float numbers", 35, 12, FG\_YELLOW);  stand\_out("Names :", 8, 36, FG\_GREEN);  stand\_out("\*", 8, 41, FG\_INTENSITY\_RED);  stand\_out("Year of birth:", 12, 36, FG\_GREEN);  stand\_out("Speed(Mb/c) :",16, 36, FG\_GREEN);  stand\_out("\*", 16, 47, FG\_INTENSITY\_RED);  stand\_out("Monthly fee(UAH) :", 20, 36, FG\_GREEN);  stand\_out("\*", 20, 52, FG\_INTENSITY\_RED);  strcpy(users[count - 1].Name, "-1");  while( !strcmp(users[count - 1].Name, "0") || !strcmp(users[count - 1].Name, "-1")) {  clear\_line(8,1);  if (!strcmp(users[count - 1].Name, "-1")) {  stand\_out("Names :", 8, 36, FG\_GREEN);  }else {  stand\_out("Names :", 8, 36, FG\_INTENSITY\_RED);  }  if (entrance == 0) {  char buffer[STR\_SIZE] = "\0";  fgets(buffer, STR\_SIZE, stdin);  entrance = 1;  }  stand\_out("\*", 8, 41, FG\_INTENSITY\_RED);  char buffer[STR\_SIZE] = "\0";  conMove(8, 45);  fgets(buffer, STR\_SIZE, stdin);  buffer[strlen(buffer) - 1] = '\0';  strcpy(users[count - 1].Name, GetName(buffer));  }  stand\_out("Names :", 8, 36, FG\_GREEN);  stand\_out("\*", 8, 41, FG\_INTENSITY\_RED);  users[count - 1].Year = -2;  while(users[count - 1].Year == -1 || users[count - 1].Year == -2) {  clear\_line(12,1);  if (users[count - 1].Year == -2) {  stand\_out("Year of birth:", 12, 36, FG\_GREEN);  }else {  stand\_out("Year of birth:", 12, 36, FG\_INTENSITY\_RED);  }  char buffer[STR\_SIZE] = "\0";  conMove(12, 52);  fgets(buffer, STR\_SIZE, stdin);  buffer[strlen(buffer) - 1] = '\0';  users[count - 1].Year = GetYear(buffer);  }  stand\_out("Year of birth:", 12, 36, FG\_GREEN);  users[count - 1].Speed = -1;  while(users[count - 1].Speed == 0 || users[count - 1].Speed == -1) {  clear\_line(16,1);  if (users[count - 1].Speed == -1) {  stand\_out("Speed(Mb/c) :",16, 36, FG\_GREEN);  }else {  stand\_out("Speed(Mb/c) :",16, 36, FG\_INTENSITY\_RED);  }  stand\_out("\*", 16, 47, FG\_INTENSITY\_RED);  char buffer[STR\_SIZE] = "\0";  conMove(16, 52);  fgets(buffer, STR\_SIZE, stdin);  buffer[strlen(buffer) - 1] = '\0';  users[count - 1].Speed = GetSpeed(buffer);  }  stand\_out("Speed(Mb/c) :",16, 36, FG\_GREEN);  stand\_out("\*", 16, 47, FG\_INTENSITY\_RED);  users[count - 1].MonthlyFee = -1;  while(users[count - 1].MonthlyFee == 0.0 || users[count - 1].MonthlyFee == -1) {  clear\_line(20,1);  if (users[count - 1].MonthlyFee == -1) {  stand\_out("Monthly fee(UAN) :", 20, 36, FG\_GREEN);  }else {  stand\_out("Monthly fee(UAN) :", 20, 36, FG\_INTENSITY\_RED);  }  stand\_out("\*", 20, 52, FG\_INTENSITY\_RED);  char buffer[STR\_SIZE] = "\0";  conMove(20, 56);  fgets(buffer, STR\_SIZE, stdin);  buffer[strlen(buffer) - 1] = '\0';  users[count - 1].MonthlyFee = GetMonthlyFee(buffer);  }  stand\_out("Monthly fee(UAN) :", 20, 36, FG\_GREEN);  stand\_out("\*", 20, 52, FG\_INTENSITY\_RED);  conHideCursor();  return users;  }  void PrintStructure(struct Provider \* Users, int index, int x, int y) {  conMove(x, y + 7);  printf("User №%i", index + 1);  conMove(x + 1, y);  printf("%i)Name: %s", index \* 4 + 1, Users[index].Name);  conMove(x + 2, y);  if ( Users[index].Year != 0) {  printf("%i) Year of birth: %i", index \* 4 + 2, Users[index].Year);  }else {  printf("%i) Year of birth: ", index \* 4 + 2);  }  conMove(x + 3, y);  printf("%i) Speed: %i", index \* 4 + 3, Users[index].Speed);  conMove(x + 4, y);  printf("%i) Monthly Fee: %g", index \* 4 + 4, Users[index].MonthlyFee);  return;  }  void GetNhigherSpeed(struct Provider \* Users, int number, int length) {  int shiftX = 0;  int shiftY = 0;  int size = length;  int temp[size];  for (int i = 0; i < size; i++) {  temp[i] = Users[i].Speed;  }  for (int i = 0; i < number; i++) {  int posMax = 0;  for (int j = 0; j < size; j++) {  if(temp[j] > temp[posMax]) {  posMax = j;  }  }  temp[posMax] = -1;  PrintStructure(Users, posMax, 5 + shiftX, 3 + shiftY);  shiftY += 24;  if (shiftY == 96) {  shiftY = 0;  shiftX += 6;  }  }  } |

Assert.c

|  |
| --- |
| #include <assert.h>  #include "function.h"  #include <string.h>  #include <pbconsole.h>  #include <stdlib.h>  void test(void) {  assert(!strcmp(GetName("asdasdasdasd"), "asdasdasdasd"));  assert(!strcmp(GetName("asdasd2asdasd"), "0"));  assert(!strcmp(GetName("asd asd2asdasd"), "0"));  assert(!strcmp(GetName("asdasd asdasd"), "0"));  assert(!strcmp(GetName("asd.asdasdasd"), "0"));  assert(!strcmp(GetName("123123"), "0"));  assert(!strcmp(GetName("\n"), "0"));  assert(!strcmp(GetName(""), "0"));  assert(GetYear("12") == -1);  assert(GetYear("1999") == 1999);  assert(GetYear("asdasdasd") == -1);  assert(GetYear("12ad adasd") == -1);  assert(GetYear("0") == -1);  assert(GetYear("") == 0);  assert(GetYear("-123") == -1);  assert(GetSpeed("") == 0);  assert(GetSpeed("3") == 0);  assert(GetSpeed("asdasdasdasd") == 0);  assert(GetSpeed("2 sdadsad") == 0);  assert(GetSpeed("2") == 2);  assert(GetSpeed("100") == 100);  assert(GetSpeed("-26") == 0);  assert(GetMonthlyFee("asdasdasd") == 0.0);  assert(GetMonthlyFee("") == 0.0);  assert(GetMonthlyFee("12.345") == 0.0);  assert(GetMonthlyFee("-123") == 0.0);  assert(GetMonthlyFee("23.83") == (float)23.83);  return;  } |

**Assert\_func.h**

|  |
| --- |
| #ifndef ASSERT\_H  #define ASSERT\_H  void test(void);  #endif |

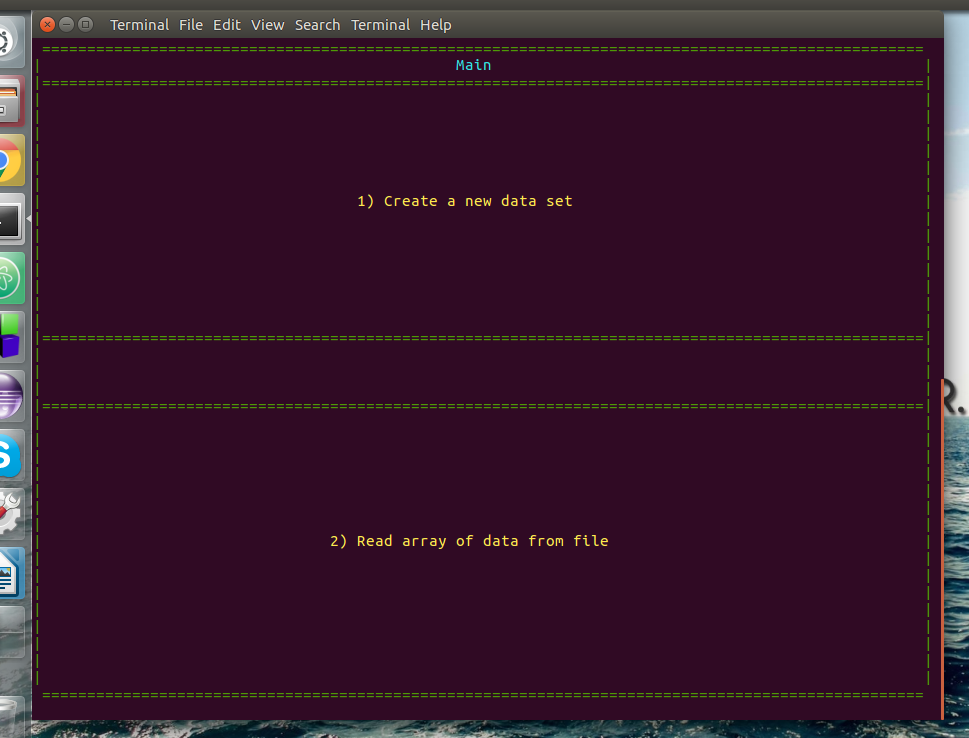
**Function.h**

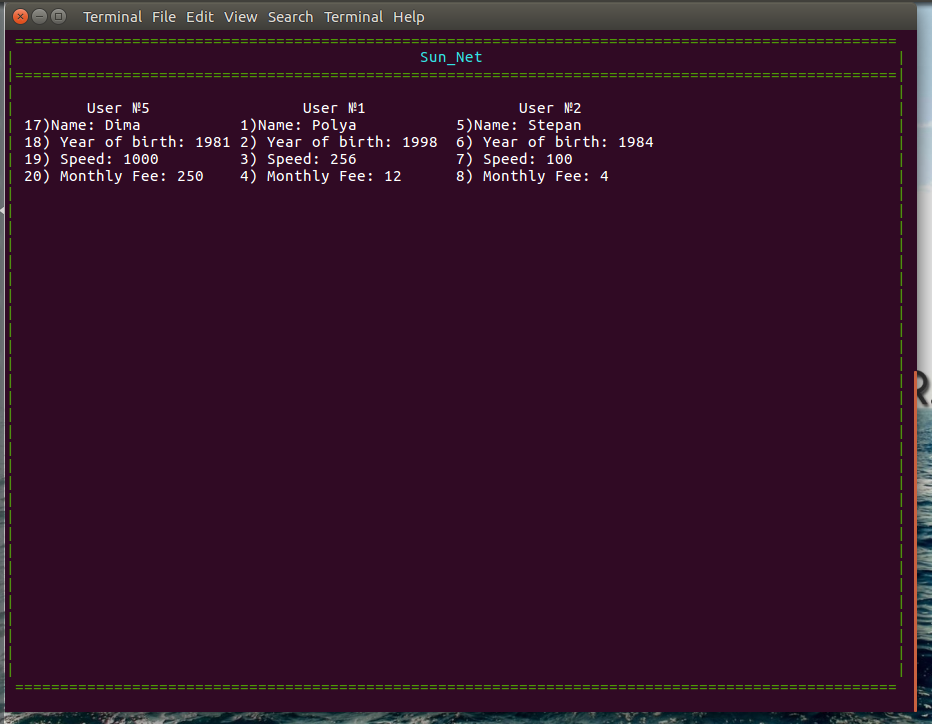
|  |
| --- |
| #ifndef FUNCTION\_H  #define FUNCTION\_H  #define STR\_SIZE 100  struct Provider {  char Name[STR\_SIZE];  int Year;  int Speed;  float MonthlyFee;  };  struct Provider \* RewriteData(struct Provider \* Users, int index);  struct Provider \* DeleteStructure(struct Provider \* Users, int \* size , int index);  void PrintUsers(struct Provider \* Users, int \* size);  int GetStructuresFromConsole(struct Provider \* users);  char \* GetName(char string[]);  int GetYear(char string[]);  int GetSpeed(char string[]);  float GetMonthlyFee(char string[]);  int GetIndex(char string[]);  struct Provider \* GetStructure(struct Provider \* Users, int count, int entrance);  void PrintStructure(struct Provider \* Users, int index, int x, int y);  void GetNhigherSpeed(struct Provider \* Users, int number, int length);  #endif |

Menu.c

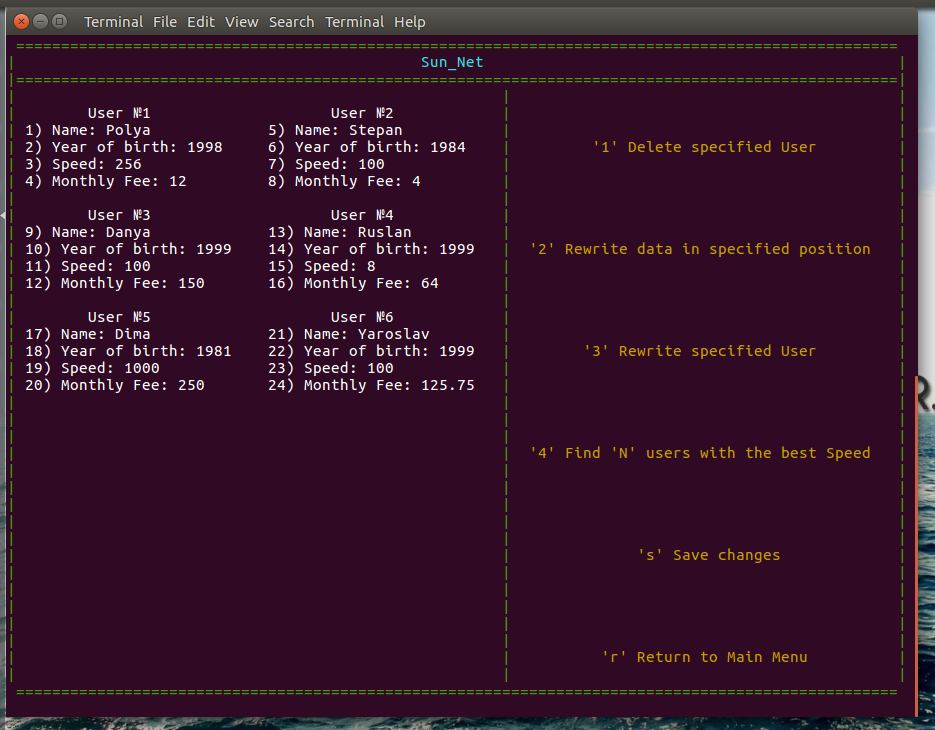
|  |
| --- |
| #include <stdlib.h>  #include <stdio.h>  #include <pbconsole.h>  void line(int x) {  int i = 0;  conMove(x,2);  conSetAttr(FG\_GREEN);  for (i = 2; i <= 99; i++) {  printf("=");  }  conSetAttr(FG\_DEFAULT);  printf("\n");  }  void clear\_line(int line, int numbers) {  int i = 0;  for (int j = 0; j < numbers; j++) {  for (i = 2; i <= 99; i++) {  conMove(line + j,i);  printf(" ");  }  }  }  void clear\_field(int x, int startY, int numbers) {  for (int i = 0; i < numbers; i++) {  conMove(x, startY + i);  printf(" ");  }  }  void parallel\_Lines(void) {  conSetAttr(FG\_GREEN);  for (int i = 0; i < 37; i++) {  conMove(2 + i, 1);  printf("|");  conMove(2 + i,100);  printf("|");  }  conSetAttr(FG\_DEFAULT);  }  void stand\_out(char str[],int x, int y, int attr) {  conMove(x,y);  conSetAttr(attr);  printf("%s", str);  conSetAttr(FG\_DEFAULT);  } |

**Приклади результатів**









**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.