**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчет

**по лабораторной работе № 7**

**по дисциплине «Программирование»**

Тема: Указатели на структуры и функции.

|  |  |  |
| --- | --- | --- |
| Студент гр. 3312 |  | Шарапов И. Д. |
| Преподаватель |  | Аббас С. А. |

Санкт-Петербург

2024

**Содержание**

[Цель работы 3](#_Toc161040894)

[Задание (Вариант 14) 3](#_Toc161040895)

[Постановка задачи и описание решения 3](#_Toc161040896)

[Описание переменных 3](#_Toc161040897)

[Структура вызова функций 4](#_Toc161040898)

[Схема алгоритма 4](#_Toc161040899)

[Текст программы 4](#_Toc161040900)

[Контрольные примеры 4](#_Toc161040901)

[Содержимое файлов 4](#_Toc161040902)

[Примеры выполнения программы 4](#_Toc161040903)

[Выводы 4](#_Toc161040904)

# Цель работы

Целью работы является изучение указателей на структуры и функции в языке Си, а также работа с динамическими массивами структур.

# Задание (Вариант 14)

Для выбранной предметной области создать динамический массив структур, содержащих характеристики объектов предметной области.

Обязательный набор полей:

* динамический массив символов, включая пробелы (name)
* произвольный динамический массив символов
* числовые поля типов int и float (не менее двух полей каждого типа)
* поле с числовым массивом.

Написать программу, обеспечивающую начальное формирование массива структур при чтении из файла (текст с разделителями — CSV) с последующим возможным дополнением элементов массива при вводе с клавиатуры. Следует использовать указатели на структуры и указатели на функции обработки массива в соответствии с вариантом задания.

Во всех случаях, когда при поиске записей результат отсутствует, следует вывести сообщение.

Выбор записей, в которых значение любого символьного поля (выбор из меню) содержит указанную подстроку без учёта регистра, сортировка результата по возрастанию значений любого из элементов поля с числовым массивом (выбор признака сортировки — из меню).

# Постановка задачи и описание решения

Для получения данных будем запрашивать у пользователя имя файла до тех пор, пока не сможем успешно открыть его. После открытия файла, мы спросим у пользователя количество строк, которые он хочет прочитать из файла.

Далее, мы обработаем данные из файла. Для этого, мы будем построчно считывать информацию об атлете. Каждую считанную строчку мы будем обрабатывать с помощью функции *fill\_struct()*, которая выделит память для новой сущности структуры атлета и заполнит её. После считывания всех данных из файла, мы закроем его и выведем сообщение об успешной обработке данных.

Затем, мы выведем пользователю список доступных команд. Мы будем считывать команды до тех пор, пока пользователь не введёт строку *"!end"*.

Если пользователь введёт строку *"!print"*, мы выведем таблицу атлетов с помощью функции *pprint()*.

Если пользователь введёт строку *"!find"*, мы спросим у него, по какому полю он хочет искать атлетов (по имени или университету). Если пользователь введёт число, отличное от 1 и 2, мы выведем сообщение *"Invalid command!"*. В противном случае, мы вызовем функцию *find\_users()* с соответствующим параметром. Эта функция спросит у пользователя строку, приведёт её к нижнему регистру. Затем, она переберёт все элементы массива структур и проверит, является ли введённая строка подстрокой соответствующего поля данного атлета, приведённого к нижнему регистру. Все подходящие атлеты будут выведены в виде таблицы. Если ни один атлет не соответствует критериям, мы выведем сообщение *"No matches found!"*.

Если пользователь введёт строку *"!sort"*, мы спросим у него, по какому полю он хочет сортировать пользователей. Если пользователь введёт число, отличное от 1, 2, 3, 4, мы выведем сообщение *"Invalid command!"*. В противном случае, мы вызовем функцию *sort\_users()* с соответствующим параметром. Эта функция выполнит сортировку пузырьком, сравнивая выбранный пользователем критерий. После сортировки мы выведем сообщение об успешном окончании сортировки и выведем таблицу атлетов с помощью функции *pprint()*.

Если пользователь введёт строку *"!add"*, мы выведем ему формат ввода нового пользователя. Мы выделим дополнительную память в динамическом массиве структур, прочитаем информацию о новом атлете, обработаем ее с помощью функции *fill\_struct()* и добавим в массив. При успешном добавлении пользователя мы выведем сообщение *"The data has been successfully added!"*.

Если пользователь введёт какую-то, не описанную выше строку, мы выведем сообщение *"Unknown command!"*.

# Описание переменных

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| Функция *int from\_str\_to\_int(char \*x)* | | | |
| 1 | x | char\* | Строка, которую будем переводить в число |
| 2 | ans | int | Число, которое получим после обработки |
| Функция *int from\_str\_to\_float(char \*x)* | | | |
| 1 | x | char\* | Строка, которую будем переводить в число |
| 2 | ans | int | Число, которое получим после обработки |
| 3 | a | int | На что домножаем целую часть |
| 4 | b | int | На что делим дробную часть |
| Функция *void from\_str\_to\_int\_mas(char \*x, int \*mas)* | | | |
| 1 | x | char\* | Строка, которую будем переводить в массив |
| 2 | mas | int\* | Массив, который будем заполнять |
| 3 | ind | int | Индекс массива |
| 4 | j | int | Индекс в строке |
| Функция *Athlete \*fill\_struct(char \*str)* | | | |
| 1 | str | char\* | Строка в формате CSV |
| 2 | user | Athlete\* | Сущность структуры Athlete |
| 3 | word | char\* | Отдельное поле, разделённое ; |
| 4 | ind | int | Индекс поля, в котором мы находимся |
| 5 | tt | int | Счётчик для перебора строки |
| 6 | pole | char\*[] | Массив указателей на начало полей |
| Функция *void print\_user(Athlete \*user)* | | | |
| 1 | user | Athlete\* | Атлет, которого выводим |
| Функция *void pprint(Athlete \*\*users, int n)* | | | |
| 1 | users | Athlete\*\* | Динамический массив атлетов |
| 2 | n | int | Количество атлетов |
| Функция *void my\_swap(Athlete \*a, Athlete \*b)* | | | |
| 1 | a | Athlete\* | Первый атлет |
| 2 | b | Athlete\* | Второй атлет |
| 3 | c | Athlete\* | Вспомогательный элемент для обмена |
| Функция *void sort\_users(Athlete \*\*users, int n, int param)* | | | |
| 1 | users | Athlete\*\* | Динамический массив атлетов |
| 2 | n | int | Количество атлетов |
| 3 | param | int | Флаг, показывающий поле для сортировки |
| Функция *char \*m\_strlwr(const char \*str)* | | | |
| 1 | str | char\* | Исходная строка |
| 2 | len | size\_t | Длина строки |
| 3 | new\_str | char\* | Строка, приведённая к нижнему регистру |
| Функция *void find\_users(Athlete \*\*users, int n, int param)* | | | |
| 1 | users | Athlete\*\* | Динамический массив атлетов |
| 2 | n | int | Количество атлетов |
| 3 | param | int | Флаг, показывающий поле для поиска |
| 4 | x | char[] | Подстрока, которую ищем |
| 5 | str | char\* | Строковое поле, по которому ищем |
| 6 | fl | int | Флаг – найдены пользователи или нет |
| Функция *int main()* | | | |
| 1 | filename | char[] | Имя файла |
| 2 | f | FILE\* | Файл для считывания |
| 3 | n | int | Количество пользователей |
| 4 | ch | int | Выбор режима для сортировки и поиска |
| 5 | users | Athlete\*\* | Динамический массив атлетов |
| 6 | str | char[] | Строка команд |
| 7 | text | char[][] | Массив всех строк о пользователях |

# **Структура вызова функций**

# **Схема алгоритма**

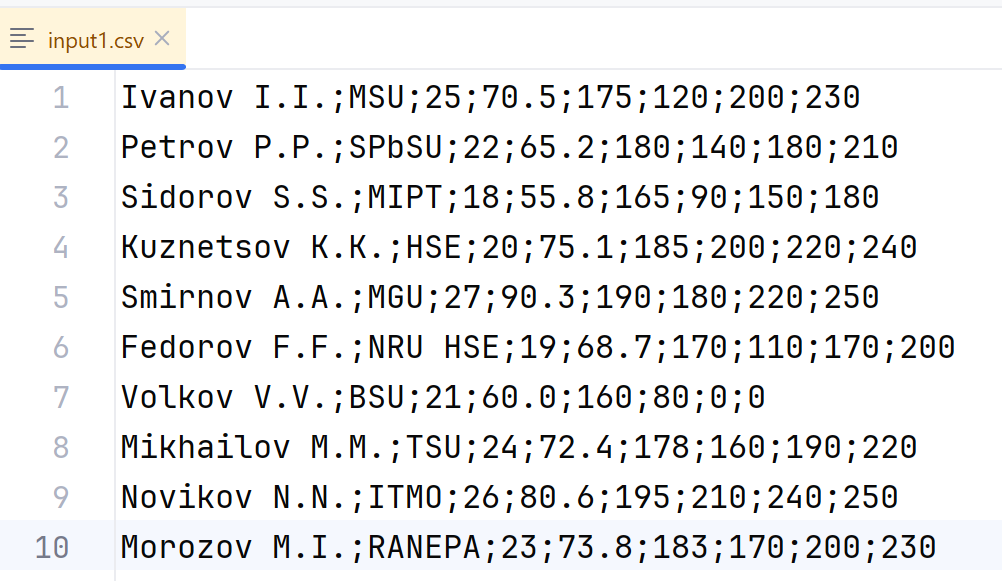
# Текст программы

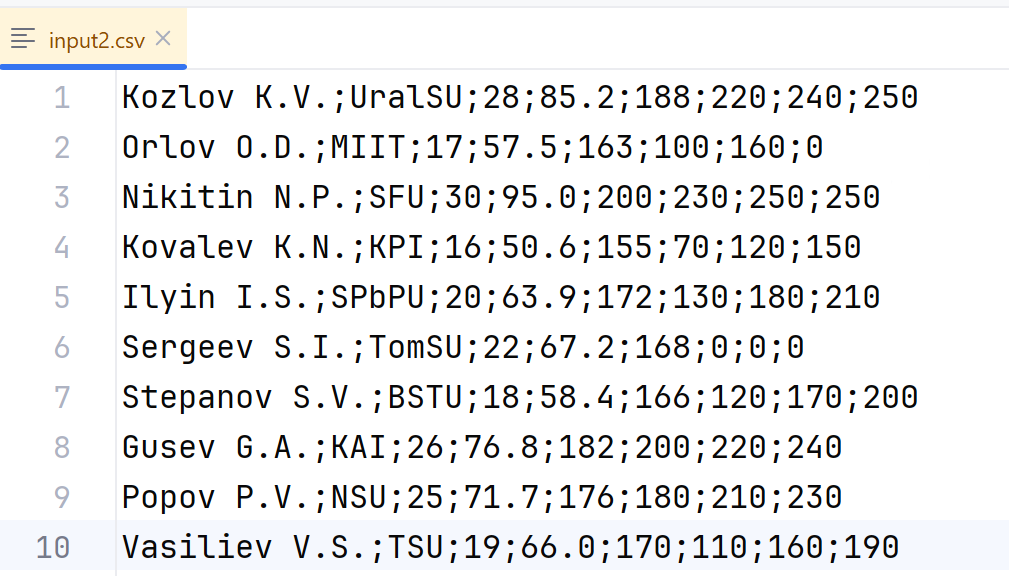
|  |
| --- |
| **#include <stdio.h> #include <stdlib.h> #include <string.h>  #define MAX\_USERS 100  struct Athlete {  char \*name; /\* имя \*/  char \*university; /\* университет \*/  int age; /\* возраст \*/  float weight; /\* вес \*/  int height; /\* рост \*/  int result[3]; /\* результаты \*/  float index; /\* индекс результата к массе \*/ };  typedef struct Athlete Athlete;   int from\_str\_to\_int(char \*x);  float from\_str\_to\_float(char \*x);  void from\_str\_to\_int\_mas(char \*x, int \*mas);  Athlete \*fill\_struct(char \*str);  void print\_line();  void print\_user(Athlete \*user);  void pprint(Athlete \*\*users, int n);  void my\_swap(Athlete \*a, Athlete \*b);  void sort\_users(Athlete \*\*users, int n, int param);  char \*m\_strlwr(const char \*str);  void find\_users(Athlete \*\*users, int n, int param);   int main() {  char filename[128];  FILE \*f;  int n, ch;  Athlete \*\*users;  char str[128];   printf("Please enter the file name:\n");  scanf("%s", filename);  f = fopen(filename, "r");  while (f == NULL) {  printf("Something went wrong!\n"  "Perhaps such a file does not exist.\n"  "Please enter the file name again:\n");  scanf("%s", filename);  f = fopen(filename, "r");  }   printf("Please enter count of lines in file (max %i):\n", MAX\_USERS);  scanf("%i", &n);  getchar();   char text[MAX\_USERS][1024];  users = (Athlete \*\*) malloc(n \* sizeof(Athlete \*));  if (users != NULL) {  for (int i = 0; i < n; ++i) {  if (fgets(text[i], sizeof(text[i]), f)) {  users[i] = fill\_struct(text[i]);  } else {  n = i;  }  }  fclose(f);  printf("The file has successfully been processed!\n");  } else {  printf("Memory error!");  }   printf("To display the data, enter the command \"!print\"\n"  "To find users, enter the command \"!find\"\n"  "To sort the data, enter the command \"!sort\"\n"  "To add new data, enter the command \"!add\"\n"  "To end the program, enter the command \"!end\"\n");   do {  scanf("%s", str);  if (!strcmp(str, "!end")) {  printf("Goodbye!\n");  } else if (!strcmp(str, "!print")) {  pprint(users, n);  } else if (!strcmp(str, "!find")) {  printf("Select a field to find by:\n"  "1 = name\n"  "2 = university\n"  "Enter only one number!\n");  scanf("%i", &ch);  if (ch < 1 || 2 < ch) {  printf("Invalid command!\n");  } else {  find\_users(users, n, ch);  }  } else if (!strcmp(str, "!sort")) {  printf("Select a field to sort by:\n"  "1 = age\n"  "2 = weight\n"  "3 = height\n"  "4 = index\n"  "Enter only one number!\n");  scanf("%i", &ch);  if (ch < 1 || 4 < ch) {  printf("Invalid command!\n");  } else {  sort\_users(users, n, ch);  printf("The data has been successfully sorted!\n");  pprint(users, n);  }  } else if (!strcmp(str, "!add")) {  printf("Enter data of the athlete in format:\n"  "name;university;age;weight;height;result1,result2,result3\n");  ++n;  users = (Athlete \*\*) realloc(users, (n) \* sizeof(Athlete \*));  if (users != NULL) {  getchar();  fgets(text[n - 1], sizeof(text[n - 1]), stdin);  users[n - 1] = fill\_struct(text[n - 1]);  printf("The data has been successfully added!\n");  } else {  printf("Something went wrong!");  }  } else {  printf("Unknown command!\n");  }  } while (strcmp(str, "!end"));   free(users);  return 0; }  int from\_str\_to\_int(char \*x) {  int ans = 0;   while (\*x != '\0') {  ans = ans \* 10 + (\*x - '0');  ++x;  }  return ans; }  float from\_str\_to\_float(char \*x) {  float ans = 0, a = 10, b = 1;   while (\*x != '\0') {  if (\*x == '.' || \*x == ',') {  a = 1;  b = 10;  } else {  ans = ans \* a + (float) (\*x - '0') / b;  if (b > 1) b \*= 10;  }  ++x;  }  return ans; }  void from\_str\_to\_int\_mas(char \*x, int \*mas) {  int ind = 0, j = 0;   while (x[j] != '\0') {  if (x[j] == ';') {  x[j] = '\0';  mas[ind++] = from\_str\_to\_int(x);  x += j + 1;  j = -1;  }  ++j;  }  mas[ind] = from\_str\_to\_int(x); }  Athlete \*fill\_struct(char \*str) {  Athlete \*user;  char \*word = str;  int ind = 0, tt;  char \*pole[5];   user = (Athlete \*) malloc(sizeof(Athlete));  if (user != NULL) {  for (tt = 0; str[tt] != '\n' && str[tt] != '\0'; ++tt) {  if (str[tt] == ';' && ind < 5) {  str[tt] = '\0';  pole[ind++] = word;  word = str + tt + 1;  }  }  str[tt] = '\0';  user->name = pole[0];  user->university = pole[1];  user->age = from\_str\_to\_int(pole[2]);  user->weight = from\_str\_to\_float(pole[3]);  user->height = from\_str\_to\_int(pole[4]);  from\_str\_to\_int\_mas(word, user->result);  user->index =  (float) (user->result[0] + user->result[1] + user->result[2]) /  user->weight;  }  return user; }  void print\_line() {  printf("+");  for (int i = 0; i < 22; printf("-"), ++i);  printf("+------------+-----+--------+--------"  "+------+------+------+-------+\n"); }  void print\_user(Athlete \*user) {  printf("| %-20s | %-10s | %-3i | %0.1f ", user->name, user->university,  user->age, user->weight);  if (user->weight < 100) printf(" ");  printf("| %-6i | %-4i | %-4i | %-4i ", user->height, user->result[0],  user->result[1], user->result[2]);  printf("| %0.3f |\n", user->index); }  void pprint(Athlete \*\*users, int n) {  print\_line();  printf("| Name | University | Age | Weight | Height "  "| Res1 | Res2 | Res3 | Index |\n");  print\_line();  for (int i = 0; i < n; ++i) {  print\_user(users[i]);  }  print\_line(); }  void my\_swap(Athlete \*a, Athlete \*b) {  Athlete c = \*a;   \*a = \*b;  \*b = c; }  void sort\_users(Athlete \*\*users, int n, int param) {  for (int i = 0; i < n; ++i) {  for (int j = i; j < n; ++j) {  if (param == 1 && users[i]->age > users[j]->age) {  my\_swap(users[i], users[j]);  } else if (param == 2 && users[i]->weight > users[j]->weight) {  my\_swap(users[i], users[j]);  } else if (param == 3 && users[i]->height > users[j]->height) {  my\_swap(users[i], users[j]);  } else if (param == 4 && users[i]->index > users[j]->index) {  my\_swap(users[i], users[j]);  }  }  } }  char \*m\_strlwr(const char \*str) {  size\_t len = strlen(str);  char \*new\_str = malloc(len + 1);   if (new\_str != NULL) {  strcpy(new\_str, str);  strlwr(new\_str);  }  return new\_str; }  void find\_users(Athlete \*\*users, int n, int param) {  char x[128];  char \*str;  int fl = 0;   printf("Enter the search string:\n");  getchar();  fgets(x, sizeof(x), stdin);  x[strlen(x) - 1] = '\0';  strlwr(x); /\* приводит x к нижнему регистру \*/  for (int i = 0; i < n; ++i) {  if (param == 1) str = users[i]->name;  else str = users[i]->university;  if (strstr(m\_strlwr(str), x) != NULL) {  if (fl == 0) {  print\_line();  printf("| Name | University | Age | Weight "  "| Height | Res1 | Res2 | Res3 | Index |\n");  print\_line();  fl = 1;  }  print\_user(users[i]);  }  }  if (fl == 0) {  printf("No matches found!\n");  } else {  print\_line();  } }** |

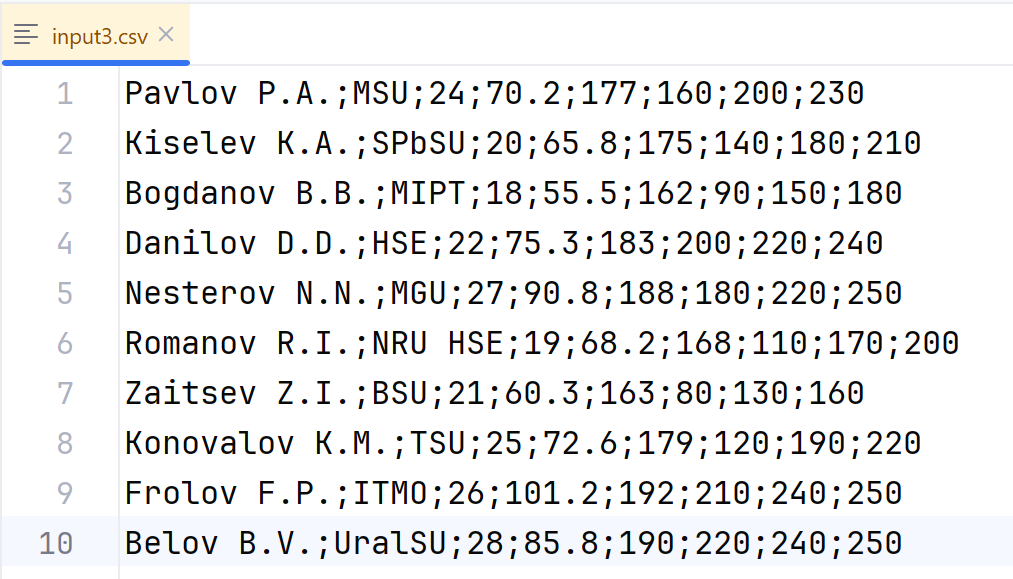
# Контрольные примеры

|  |  |  |
| --- | --- | --- |
| № | Исходные данные | Результаты |
| 1 | input1.csv  10  !print  !sort  2  !end |  |
| 2 | input2.csv  10  !find  2  m  !end |  |
| 3 | input3.csv  10  !print  !add  Klimov K.K.;SPbPU;21 ;64.5;170;130;180;210  !print  !end |  |

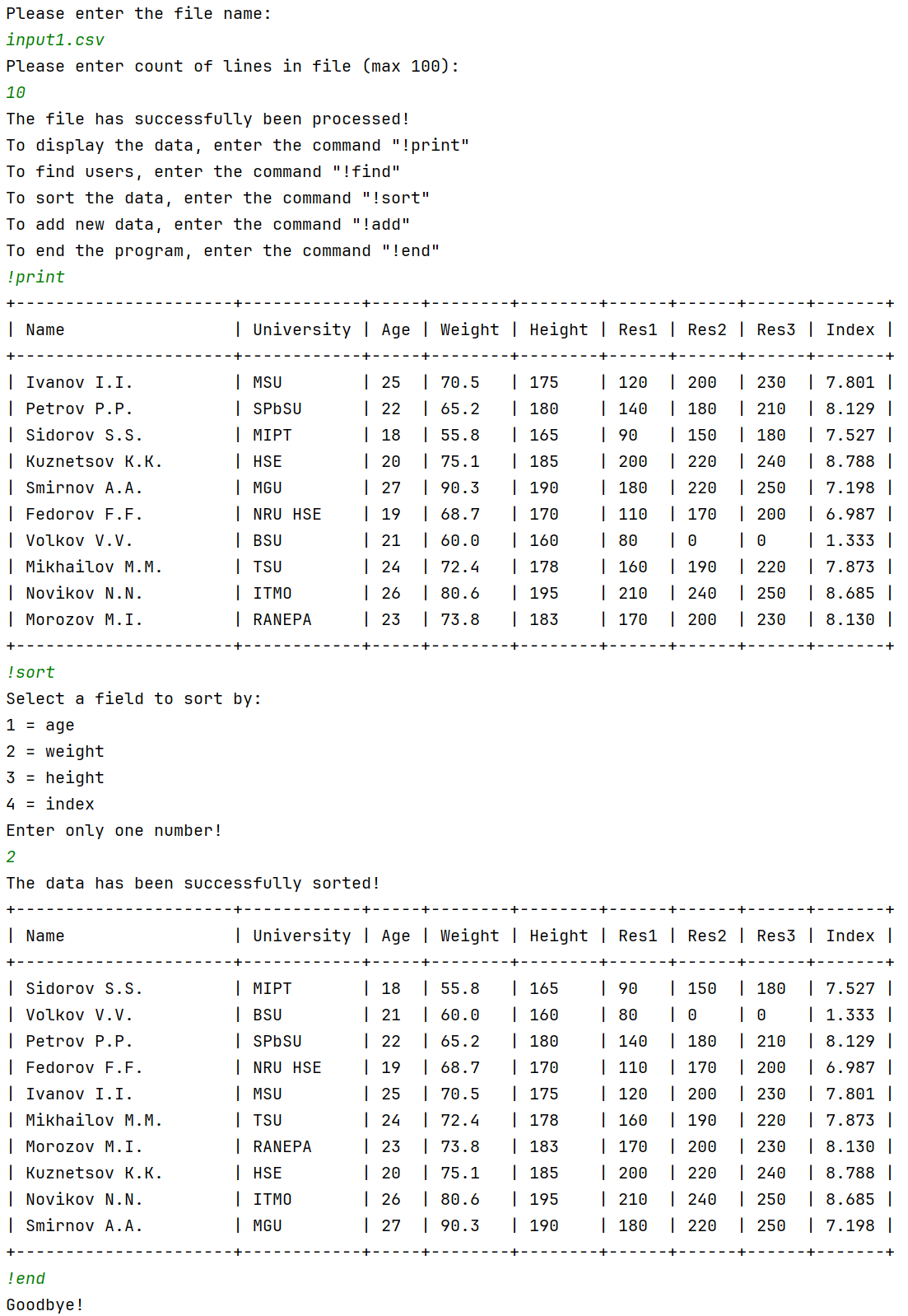
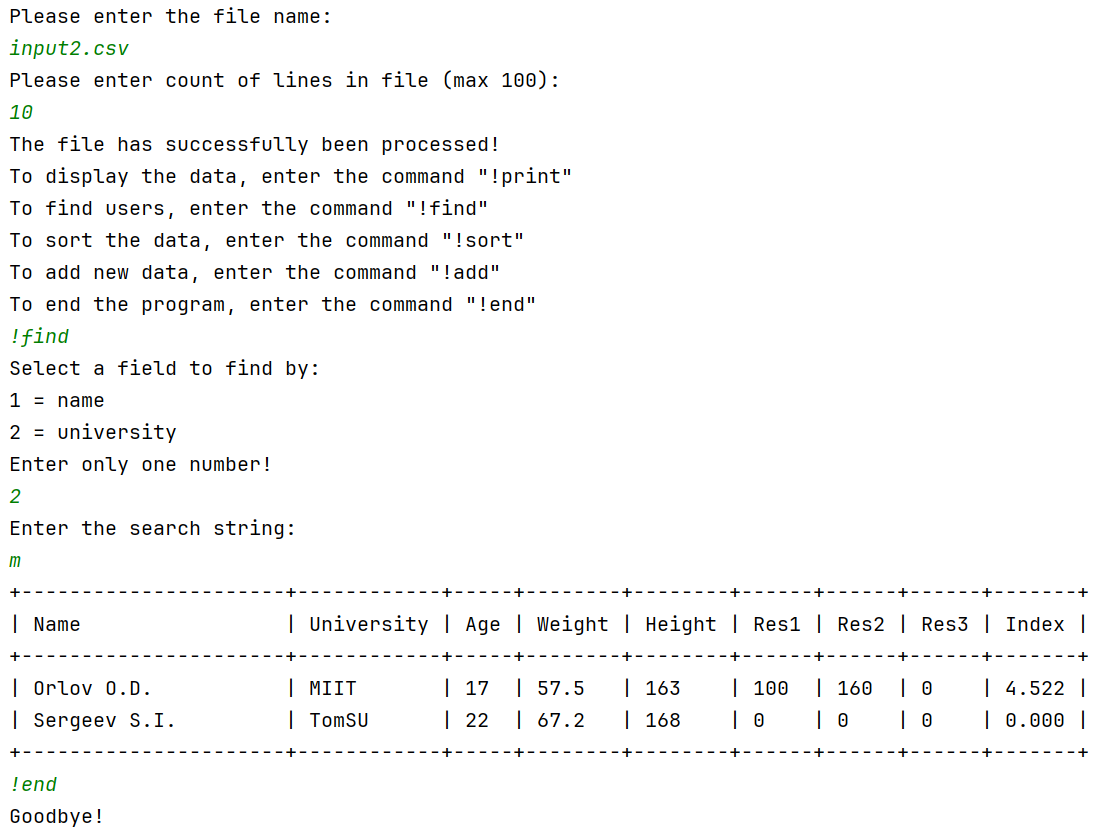
# Содержимое файлов

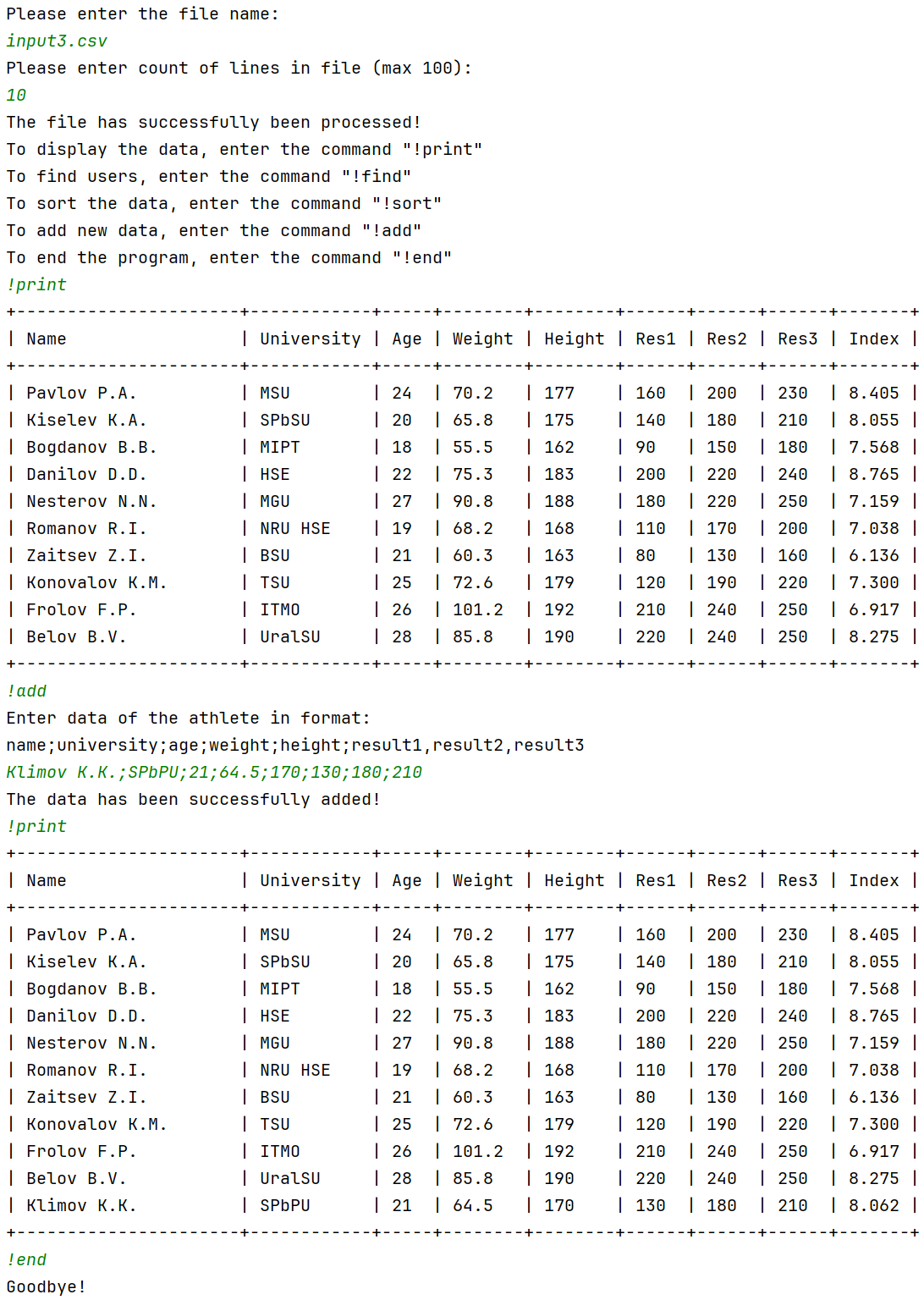






# Примеры выполнения программы



# Выводы

В результате выполнения работы изучены особенности работы с указателями на структуры и функции в языке Си. А также получены практические навыки в работе с динамическими массивами структур.