**ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №7**

**Тема: «Работа с файлами»**

***Цель работы:*** Научиться использовать функции в C++, необходимые для работы с файлами.

***Задание №1:*** Программа спрашивает, сколько записей необходимо добавить и запрашивает данные о работнике предприятия. Пользователь вводит:

* ФИО;
* Пол;
* Дата рождения;
* Место рождения;
* Адрес проживания;
* Семейное положение;
* Номер телефона;
* Образование;
* Должность;
* Стаж;
* Размер заработной платы.

У пользователя должна быть возможность выхода из программы до того, как ввёл все записи. Все введённые данные должны сохраняться в файле.

***Ход выполнения***

Запрашиваем у пользователя количество записей и сохраняем в переменную. Создаём переменные под данные:

* Для ФИО (массив из 99 символов),
* Для пола (символ),
* День, месяц, год (число),
* Точка (служебная),
* Адрес рождения (массив из 499 символов),
* Адрес проживания (массив из 499 символов),
* Семейное положение (массив из 14 символов),
* Телефон (целочисленный),
* Образование (массив из 9 символов),
* Должность (массив из 199 символов),
* Стаж и зарплата (целочисленные).

Затем очищаем файл и открываем его для записи, записываем туда текущую дату и количество записей.

В цикле от 0 до введённого пользователем значения запрашиваем данные и записываем их в файл. После чего закрываем файл.

Чтобы досрочно выйти из программы её можно просто закрыть уже введённые данные сохранятся в файл.

На листинге 1 представлен код программы. На рисунке 1 – результат теста программы.

Листинг 1 – Код программы

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13.  14.  15.  16.  17.  18.  19.  20.  21.  22.  23.  24.  25.  26.  27.  28.  29.  30.  31.  32.  33.  34.  35.  36.  37.  38.  39.  40.  41.  42.  43.  44.  45.  16.  47.  48.  49.  50.  51.  52.  53.  54.  55.  56.  57.  58.  59.  60.  61.  62.  63.  64.  65.  66.  67.  68.  69.  70.  71.  72.  73.  74.  75.  76.  77.  78.  79.  80.  81.  82.  83.  84.  85.  86.  87.  88.  89.  90. | #include <iostream>  #include <cstring>  #include <cstdio>  #include <ctime>  #include<limits>  #include <fstream>  using namespace std;  int main(){  int num\_of\_entries;  cout << "Введите количество записей: ";  cin >> num\_of\_entries;  char fullName[100]; // ФИО  char sex; //Пол  int day; // День рождения  int month; // Месяц рождения  int year; // Год рождения  char dot; // Точка (функциональный элемент)  char place\_of\_birth[500]; // Адрес рождения  char place\_of\_residence[500]; // Адрес проживания  char marital\_status[15]; // Семейное положение  unsigned int telephone\_number; // Телефонный номер  char education[10]; // Образование  char post[200]; // Должность  int experience; // Опыт  int wages; // З\п  ofstream file;  file.open("persons.txt", ios\_base::trunc);  file.close();  file.open("persons.txt", ios\_base::app);  time\_t now = time(0);  char\* dt = ctime(&now);  file << dt << endl << num\_of\_entries << endl;  for (int i = 0; i < num\_of\_entries; ++i){  cout << i+1 << " запись" << endl;  cout << "ФИО: ";  cin.clear(); // очищаем буфер  cin.ignore(numeric\_limits<streamsize>::max(), '\n');  cin.getline(fullName, 99, '\n');  file << fullName << endl;  cout << "Пол: ";  // cin.clear(); // очищаем буфер  // cin.ignore(numeric\_limits<streamsize>::max(), '\n');  // cin.getline(sex, 1, '\n');  cin >> sex;  file << sex << endl;  cout << "Дата рождения: ";  // cin.clear();  // cin.ignore(numeric\_limits<streamsize>::max(), '\n');  cin >> day >> dot >> month >> dot >> year;  file << day << '.' << month << '.' << year << endl;  cout << "Место рождения: ";  cin.clear();  cin.ignore(numeric\_limits<streamsize>::max(), '\n');  cin.getline(place\_of\_birth, 499, '\n');  file << place\_of\_birth << endl;  cout << "Адрес проживания: ";  // cin.clear();  // cin.ignore(numeric\_limits<streamsize>::max(), '\n');  cin.getline(place\_of\_residence, 499, '\n');  file << place\_of\_residence << endl;  cout << "Семейное положение: ";  // cin.clear();  // cin.ignore(numeric\_limits<streamsize>::max(), '\n');  cin.getline(marital\_status, 14, '\n');  file << marital\_status << endl;  cout << "Телефонный номер: ";  // cin.clear();  // cin.ignore(numeric\_limits<streamsize>::max(), '\n');  cin >> telephone\_number;  file << telephone\_number << endl;  cout << "Образование: ";  cin.clear();  cin.ignore(numeric\_limits<streamsize>::max(), '\n');  cin.getline(education, 9, '\n');  file << education << endl;  cout << "Должность: ";  // cin.clear();  // cin.ignore(numeric\_limits<streamsize>::max(), '\n');  cin.getline(post, 199, '\n');  file << post << endl;  cout << "Опыт: ";  cin >> experience;  file << experience << endl;  cout << "З/п: ";  cin >> wages;  file << wages << endl;  }  file.close();  return 0;  } |

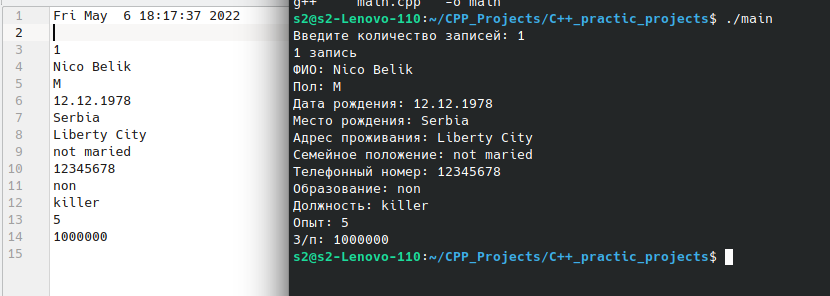


Рисунок 1 – Работа программы

***Задание №2:*** Доработать предыдущую программу, добавив возможность вывода данных из файла в консоль по введённому ФИО.

***Ход выполнения***

Для начала необходимо создать массив, в котором будут хранится все человеки, прочитанные из файла. Для этого создаём структуру, где задаём все поля (можно было использовать классы, но в них нет необходимости, и со структурами работать проще).

Затем читаем первые 3 строки файла (первая – дата создания, вторая пробел, третья – количество записей. Исходя из количества записей, создаём массив из «персон» и выводим техническую информацию.

В цикле построчно читаем файл и помещаем данные в переменную. При помощи конструкции switch-case определяем к какому полю относится текущая строка и записываем данные в массив. Как только строки закончились, запрашиваем у пользователя ФИО, и вызываем функцию для поиска данных, в которой в цикле проходимся по массиву и сравниваем ФИО с переданным, если они совпадают, то передаем текущий объект в функцию вывода. Здесь выводятся подписи к полям и сами поля. Если ФИО не найдено, то выводим сообщение.

Код программы представлен на листинге 2. Тест программы – на рисунке 2.

Листинг 2 – Код программы

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13.  14.  15.  16.  17.  18.  19.  20.  21.  22.  23.  24.  25.  26.  27.  28.  29.  30.  31.  32.  33.  34.  35.  36.  37.  38.  39.  40.  41.  42.  43.  44.  45.  46.  47.  48.  49.  50.  51.  52.  53.  54.  55.  56.  57.  58.  59.  60.  61.  62.  63.  64.  65.  66.  67.  68.  69.  70.  71.  72.  73.  74.  75.  76.  77.  78.  79.  80.  81.  82.  83.  84.  85.  86.  87.  88.  89.  90.  91.  92.  93.  94.  95.  96.  97.  98.  99.  100.  101.  102.  103.  104.  105.  106.  107.  108.  109.  110.  111.  112.  113.  114.  115.  116.  117.  118.  119.  120.  121.  122.  123. | #include <iostream>  #include <cstring>  #include <ctime>  #include <fstream>  #include <string>  using namespace std;  struct Persona{  string fullName; // ФИО  char sex; //Пол  string date\_of\_bird; // Дата рождения  string place\_of\_birth; // Адрес рождения  string place\_of\_residence; // Адрес проживания  string marital\_status; // Семейное положение  unsigned int telephone\_number; // Телефонный номер  string education; // Образование  string post; // Должность  int experience; // Опыт  int wages; // З\п  };  bool find\_data(Persona\* arr, string fn, int len);  void print\_data(Persona obj);  int main(){  string s, date\_of\_create, space, persons\_number\_s;  int i = 0;  int k = 1;  int l = 0;  ifstream file("persons.txt");  getline(file, date\_of\_create);  getline(file, space);  getline(file, persons\_number\_s);  const int j = atoi( persons\_number\_s.c\_str() );  Persona persons[j];  cout << "Persons: " << j << endl << "Date of creation: " << date\_of\_create  << endl;  while(getline(file, s)){ // пока не достигнут конец файла класть  очередную строку в переменную (s)  if(l<j){  switch (k){  case 1:{  persons[l].fullName = s;  break;  }  case 2:{  const char\* sx = s.c\_str();  persons[l].sex = sx[0];  break;  }  case 3:{  persons[l].date\_of\_bird = s;  break;  }  case 4:{  persons[l].place\_of\_birth = s;  break;  }  case 5:{  persons[l].place\_of\_residence = s;  break;  }  case 6:{  persons[l].marital\_status = s;  break;  }  case 7:{  persons[l].telephone\_number = atoi( s.c\_str() );  break;  }  case 8:{  persons[l].education = s;  break;  }  case 9:{  persons[l].post = s;  break;  }  case 10:{  persons[l].experience = atoi( s.c\_str() );  break;  }  case 11:{  persons[l].wages = atoi( s.c\_str() );  break;  }  }  if(k<11){  k++;  }else{  k=0;  l++;  }  }  }  file.close();  string fullName;  cout << "Введите ФИО: ";  getline(cin, fullName);  find\_data(persons, fullName, j);  return 0;  }  bool find\_data(Persona\* arr, string fn, int len){  for (int i = 0; i < len; ++i){  if(arr[i].fullName == fn){  print\_data(arr[i]);  return 1;  }  }  cout << "Не найдено..." << endl;  return 0;  };  void print\_data(Persona obj){  cout << "ФИО: " << obj.fullName << endl;  cout << "Пол: " << obj.sex << endl;  cout << "Дата рождения: " << obj.date\_of\_bird << endl;  cout << "Место рождения: " << obj.place\_of\_birth << endl;  cout << "Место проживания: " << obj.place\_of\_residence << endl;  cout << "Семейное положение: " << obj.marital\_status << endl;  cout << "Телефонный номер: " << obj.telephone\_number << endl;  cout << "Образование: " << obj.education << endl;  cout << "Должность: " << obj.post << endl;  cout << "Стаж: " << obj.experience << endl;  cout << "З/п: " << obj.wages << endl;  }; |

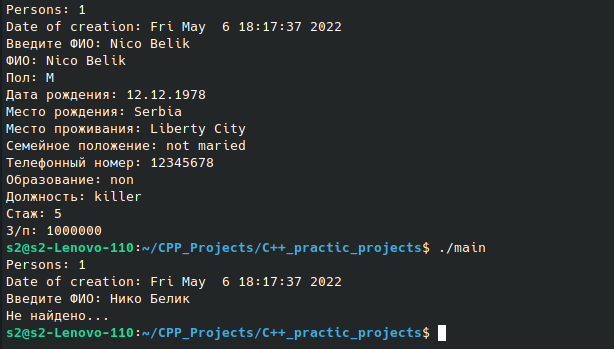


Рисунок 2 – Тест программы

***Задание №3:*** Написать текстовый квест. В игре должны быть варианты выбора, которые должны влиять на концовку игры. Весь сюжет и иллюстрации (если они будут) должны быть в текстовом файле.

***Ход выполнения***

Файл, хранящий сам квест состоит их строк следующего формата (каждая строка – это шаг квеста):

<Номер шага>|<Текст шага>|< «Кнопка» 1>^<следующий шаг>^< «Кнопка» 2>^<следующий шаг>...

Таким образом каждый шаг имеет свой номер, текст, и набор кнопок при «нажатии» на которые осуществляется переход к следующему шагу.

Сам квест не дописан, т.к. я программист, а не автор захватывающих сюжетов, но основные моменты в нём представлены.

В программе файл читается построчно и далее разбивается по составляющим. Создаётся вектор элементов, из строки создаётся буфер чтения, и посредством функции getline(ss, item, delim) в цикле происходит чтение строки по разделителю «|» и запись элементов в векторный массив. Первый элемент массива – номер шага, поэтому сравнивая его с переменной «следующий шаг» можно понять, выводить ли текущий элемент на печать. Второй элемент массива – это текст, который сразу можно вывести на экран.

Далее считываем «кнопки», для этого создаём двумерный векторный массив, и аналогично разбиваем строку на подстроки, только на этот раз по сепаратору «^». Выводим «кнопки» на печать, запрашиваем у пользователя номер «кнопки» и в соответствии с выбором помещаем в переменную следующего шага то или иное значение. Таким образом при чтении следующей строки переданный следующий шаг будет сравниваться с тем, что прочтён из файла и если они не совпадают, то осуществляется переход к следующей строке.

Код программы представлен на листинге 3, результат тестов на рисунке 3.

Листинг 3 – Код программы

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13.  14.  15.  16.  17.  18.  19.  20.  21.  22.  23.  24.  25.  26.  27.  28.  29.  30.  31.  32.  33.  34.  34.  35.  36.  37.  38.  39.  40.  41.  42.  43.  44.  45.  46.  47.  48.  49.  50.  51.  52.  53.  54.  55.  56.  57.  58.  59.  60.  61.  62.  63.  64.  65.  66.  67.  68.  69.  70.  71.  72.  73.  74.  75.  76.  77.  78. | #include <iostream>  #include <cstring>  #include <fstream>  #include <vector>  #include <sstream>  using namespace std;  int main(){  string s;  int next\_step = 1;  ifstream file("quest.txt");  while(getline(file, s)){  const char delim = '|';  const char delim2 = '^';  vector<string> elems;  stringstream ss(s);  string item;  while(getline(ss, item, delim)){  elems.push\_back(item);  }  int cur\_step = stoi(elems[0]);  if(next\_step > cur\_step)  continue;  else if(next\_step < cur\_step)  break;  string text = elems[1];  cout << text << endl;  vector<vector<string>> buttons;  stringstream sb(elems[2]);  bool k = 0;  string txt;  while(getline(sb, item, delim2)){  if(k == 0){  txt = item;  k = 1;  }else{  buttons.push\_back({txt, item});  k = 0;  }  }  int step\_num = 1;  cout << endl;  for(auto& it : buttons){  cout << step\_num << ") " << it[0] << " ";  step\_num++;  }  cout << endl;  int user\_step;  next\_step = 0;  while(!next\_step){  cin >> user\_step;  switch(user\_step){  case 1:  if(buttons.size()>0)  next\_step = stoi(buttons[0][1]);  else  cout << "Неверное значение!" << endl;  break;  case 2:  if(buttons.size()>1)  next\_step = stoi(buttons[1][1]);  else  cout << "Неверное значение!" << endl;  break;  case 3:  if(buttons.size()>2)  next\_step = stoi(buttons[2][1]);  else  cout << "Неверное значение!" << endl;  break;  default:  cout << "Неверное значение!" << endl;  break;  }  }  }  cout << "Конец!" << endl;  file.close();  return 0;  } |

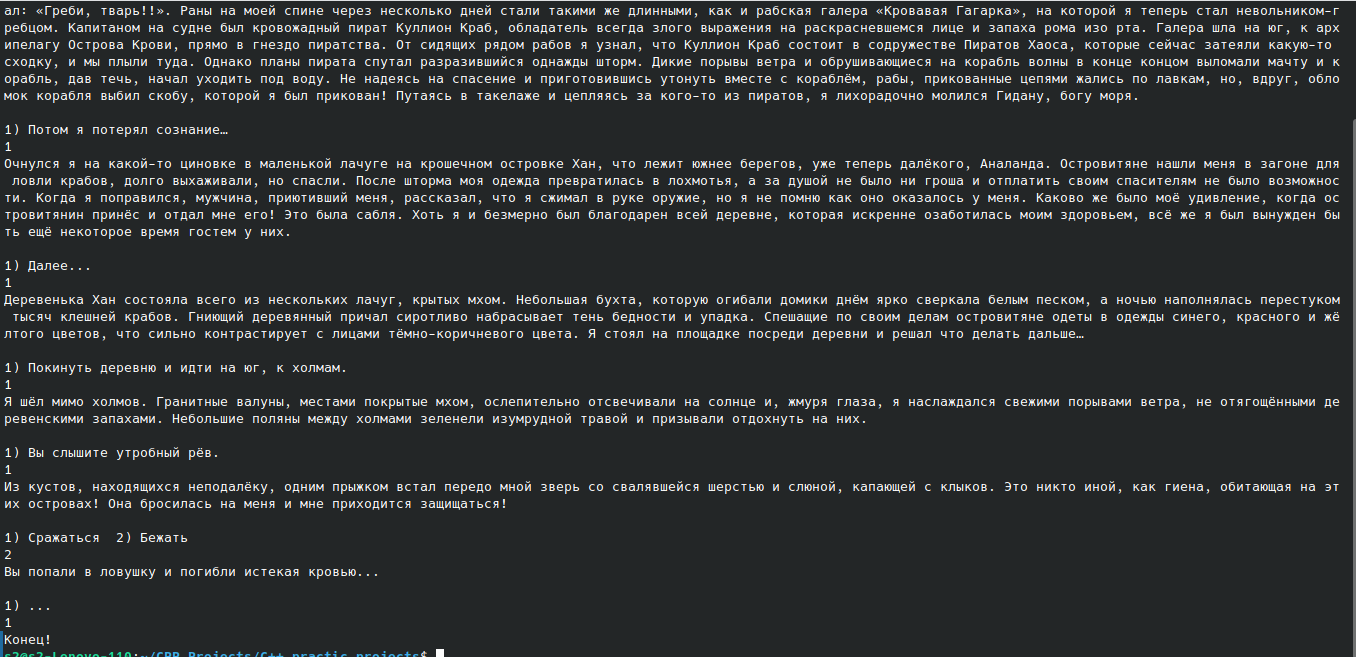


Рисунок 3 – Тест программы

***Задание №4:*** Необходимо написать программу, которая выводит код программы в текстовый файл, а имя файла, откуда выгружается код, вводит пользователь. Программа проверяет, у всех ли открывающихся скобок имеются закрывающиеся пары. Результат программы также выводится в отдельный файл и на экран.

***Ход выполнения***

Запрашиваем у пользователя имя файла, открываем его и читаем в переменную. Проходимся по этой переменной и считаем открывшиеся и закрывшиеся скобки, если их количество равно, выводим об этом сообщение на экран и в файл, который имеет путь code\_4/<file>.log, где директорию code\_4 создаём с помощью функции system("mkdir codes\_4").

Весь считанный код записываем в файл в ту же директорию под тем же именем, что ввёл пользователь.

Стоит обратить внимание на слэш, в Windows он обратный.

Код представлен на листинге 4, тест на рисунке 4.

Листинг 4 – Код программы

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10  11.  12.  13.  14.  15.  16.  17.  18.  19.  20.  21.  22.  23.  24.  25.  26.  27.  28.  29.  30.  31.  32.  33.  34.  35.  36.  37.  38.  39.  40.  41.  42.  43.  44.  45.  46.  47.  48.  49.  50.  51.  52.  53. | #include <iostream>  #include <cstring>  #include <fstream>  #include <sstream>  #include <vector>  using namespace std;  int main(){  string file\_path;  getline(cin, file\_path);  vector<string> path;  stringstream ss(file\_path);  string item;  const char separator = '/';  while(getline(ss, item, separator)){  path.push\_back(item);  }  ifstream file(file\_path);  string s;  char temp;  if(!file) {  cerr << "File error." << endl;  return 1;  }  file.unsetf(ios::skipws);  while(file >> temp)  s += temp;  file.close();  int opening\_parenthesis = 0;  int closeing\_parenthesis = 0;  for(int i=0; i<s.length(); i++){  if(s[i] == '('){  opening\_parenthesis++;  }else if(s[i] == ')'){  closeing\_parenthesis++;  }  }  system("mkdir codes\_4");  ofstream file2("codes\_4/"+path[path.size()-1], ios\_base::trunc | ios\_base::out);  file2 << s;  file2.close();  ofstream file3("codes\_4/"+path[path.size()-1]+".log", ios\_base::trunc | ios\_base::out);  if(closeing\_parenthesis != opening\_parenthesis){  cout << "Не все скобки закрыты!!!" << endl;  file3 << "Не все скобки закрыты!!!" << endl;  }else{  cout << "Все скобки закрыты." << endl;  file3 << "Все скобки закрыты." << endl;  }  file3.close();  return 0;  } |

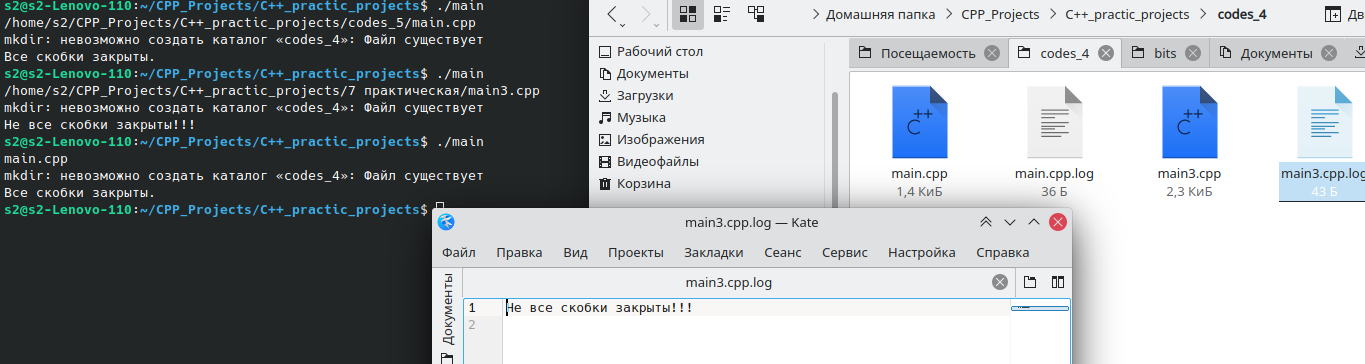


Рисунок 4 – Тест программы