|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение   высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  |  | |
|  |  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 3** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**    **Тема: «Применение хеш-таблицы для поиска данных в двоичном файле с записями фиксированной длины»** | |
|  | |
| Выполнил студент группы ИКБО-06-21 | Алиев И.А. |
| Принял преподаватель | Филатов А.С. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2022

**Задание 1**

# Цель работы

Получение навыков применения файловых потоков языка С++ (или файлов языка Си) по управлению текстовым и двоичным файлами.

# Постановка задачи

1. Разработать программу, управления текстовым файлом.
   1. Реализация ввода-вывода на основе файловых потоков С++: ofstream, ifstrim.
   2. Имя физического файла вводится пользователем и передается в функции обработки через параметр.
   3. Разработать функции для выполнения операций над текстовым файлом:
      1. создание текстового файла кодировки ASCII, содержащего десятичные числа по несколько чисел на строке;
      2. вывод содержимого текстового файла;
      3. добавление новой строки в конец файла;
      4. прочитать значение числа, указав его порядковый номер в файле, и вернуть его значение;
      5. определить количество чисел в файле.
   4. Разработать приложение и выполнить тестирование всех функций. Приложение должно содержать диалоговый интерфейс на основе текстового меню.
   5. Приложение должно осуществлять контроль открытия и существования файла в основной программе перед вызовом функций. Перед закрытием файла, проверять отсутствие ошибок ввода и вывода (метод good).
   6. Создать модуль и перенести в него все отлаженные функции. Исключить функции из приложения. Отладить приложение, подключив к нему модуль с функциями.
   7. Разработать функции для реализации дополнительных операций, определенных вариантом и сохранить их в модуле с остальными функциями.
   8. Выполнить тестирование приложения в полном объеме.

Вариант №1. Дополнительная операция:

|  |
| --- |
| Удаление значения строки с заданным номером, путем создания нового файла, удаление старого и переименование нового, указав ему имя удаленного файла. |

# Решение

В С++ аппарат работы с файлами основан на объектах классов, хранящихся в файлах ostream и istream, fstream.

Класс файловых потоков – это иерархия, базовым классов этой иерархии является абстрактный класс с именем ios, который находится на вершине иерархии. Он используется только в качестве базового класса для остальных потоков ввода-вывода.

Поток – это логический интерфейс к файлу.

При работе с потоковой библиотекой ввода-вывода программист обычно достаточно активно использует следующие классы:

istrstream - класс входных строковых потоков;

ostrstream - класс выходных строковых потоков;

strstream - класс двунаправленных строковых потоков (ввода-вывода);

ifstream - класс входных файловых потоков;

ofstream - класс выходных файловых потоков;

fstream - класс двунаправленных файловых потоков (ввода-вывода);

constream - класс консольных выходных потоков.

Тестовый файл – это последовательность символов, сохраненная на внешнем носителе. Текстовый файл может быть создан текстовым редактором кодировки ASCII и средствами программы.

Функции, разработанные в программе:

## Задание 1

В данном задании необходимо создать текстовый файл кодировки ASCII, содержащий десятичные числа по несколько чисел на строке. Для этого создадим функцию createFile(), в которой с помощью класса ofstream создадим текстовый файл с именем name и запишем в него числа считанные с клавиатуры оператором <<.

Код функции:

|  |
| --- |
| void createFile(string name)  {  ofstream file;  int k;  string str;  file.open(name);  if (file.is\_open()) {  cin >> k;  cin.ignore();  for (int I = 0; I < k; i++) {  getline(cin, str);  file << str << ‘\n’;  }  if (file.good())  file.close();  }  } |

Фрагмент кода из main:

|  |
| --- |
| cout << "Введите количество строк файла, а затем содержимое строк:\n";  createFile(name); |

## Задание 2

В данном задании необходимо вывести содержимое текстового файла. Для этого вызовем функцию outputFile(), в которой с помощью класса ifstream откроем файл с именем name в режиме чтения. Далее в цикле while получим и выведем каждый символ файла.

Код функции:

|  |
| --- |
| void outputFile(string name)  {  ifstream file;  char c;  file.open(name);  if (file.is\_open()) {  while (file.get(с)) {  cout << c;  }  if (file.good())  file.close();  }  } |

Фрагмент кода из main:

|  |
| --- |
| outputFile(name); |

## Задание 3

В данном задании необходимо добавить новую строку в конец файла. Для этого реализуем функцию addStr(), в которой с помощью класса ofstream откроем файл с именем name в режиме дозаписи и запишем в него прочитанную с клавиатуры строку оператором <<.

Код функции:

|  |
| --- |
| void addStr(string name)  {  ofstream file;  string str;  file.open(name, ofstream::app);  if (file.is\_open()) {  cin.ignore();  getline(cin, str);  file << str << '\n';  if (file.good())  file.close();  }  } |

Фрагмент кода из main:

|  |
| --- |
| cout << "Введите содержимое строки:\n";  addStr(name); |

## Задание 4

В данном задании необходимо по указанному порядковому номеру получить значение числа из файла. Для этого реализуем функцию getValueByNum(), в которой с помощью класса ifstream откроем файл с именем name в режиме чтения. Далее в цикле for будем считывать числа из файла оператором >> пока не дойдем до нашего номера num. Затем вернем искомое значение.

Код функции:

|  |
| --- |
| int getValueByNum(string name, int num)  {  ifstream file;  int value;  file.open(name);  if (file.is\_open()) {  for (int i = 0; i < num; i++) {  file >> value;  }  if (file.good())  file.close();  return value;  }  } |

Фрагмент кода из main:

|  |
| --- |
| int num;  cout << "Введите номер числа: ";  cin >> num;  cout << "Искомое число: " << getValueByNum(name, num) << '\n'; |

## Задание 5

В данном задании необходимо определить количество чисел в файле. Для этого реализуем функцию getAmount(), в которой с помощью класса ifstream откроем файл с именем name в режиме чтения. Далее в цикле while будем считывать числа из файла оператором >> и инкрементировать счетчик amount. В конце вернем значение счетчика.

Код функции:

|  |
| --- |
| int getAmount(string name)  {  ifstream file;  int value, amount = 0;  file.open(name);  if (file.is\_open()) {  while (file >> value) {  amount++;  }  if (file.good())  file.close();  return amount;  }  } |

Фрагмент кода из main:

|  |
| --- |
| cout << "Количество чисел в файле: " << getAmount(name) << '\n'; |

## Задание 6

В данном задании необходимо удалить значения строки с заданным номером, путем создания нового файла, удаления старого и переименования нового, указав ему имя удаленного файла. Для этого реализуем функцию personalTask(), в которой с помощью класса ifstream откроем файл с именем name в режиме чтения, а также с помощью класса ofstream новый файл в режиме записи. В цикле while будем считывать строки из файла функцией getline() и записывать их в новый файл, кроме того случая когда номер строки будет равен параметру num. В конце удалим файл, а новому файлу поменяем имя на name.

Код функции:

|  |
| --- |
| void personalTask(string name, int num)  {  string str;  int num, i = 1;  ifstream file;  file.open(name);  ofstream newFile;  newFile.open("newText.txt");  if (file.is\_open() && newFile.is\_open()) {  while (getline(file, str)) {  if (num == i)  i++;  else {  newFile << str << '\n';  i++;  }  }  if (file.good())  file.close();  remove(name.c\_str());  if (newFile.good())  newFile.close();  rename("newText.txt", name.c\_str());  }  } |

Фрагмент кода из main:

|  |
| --- |
| cout << "Введите номер строки: ";  cin >> num;  personalTask(name, num); |

## Работа программы

Работа программы производится с помощью специально разработанного интерфейса, в котором выводятся: номер работы, фамилия и имя студента. Далее, после того как вводится имя файла, выводятся последовательно номера заданий с условиями. Затем пользователю просто необходимо ввести номер для выбора соответствующего задания.

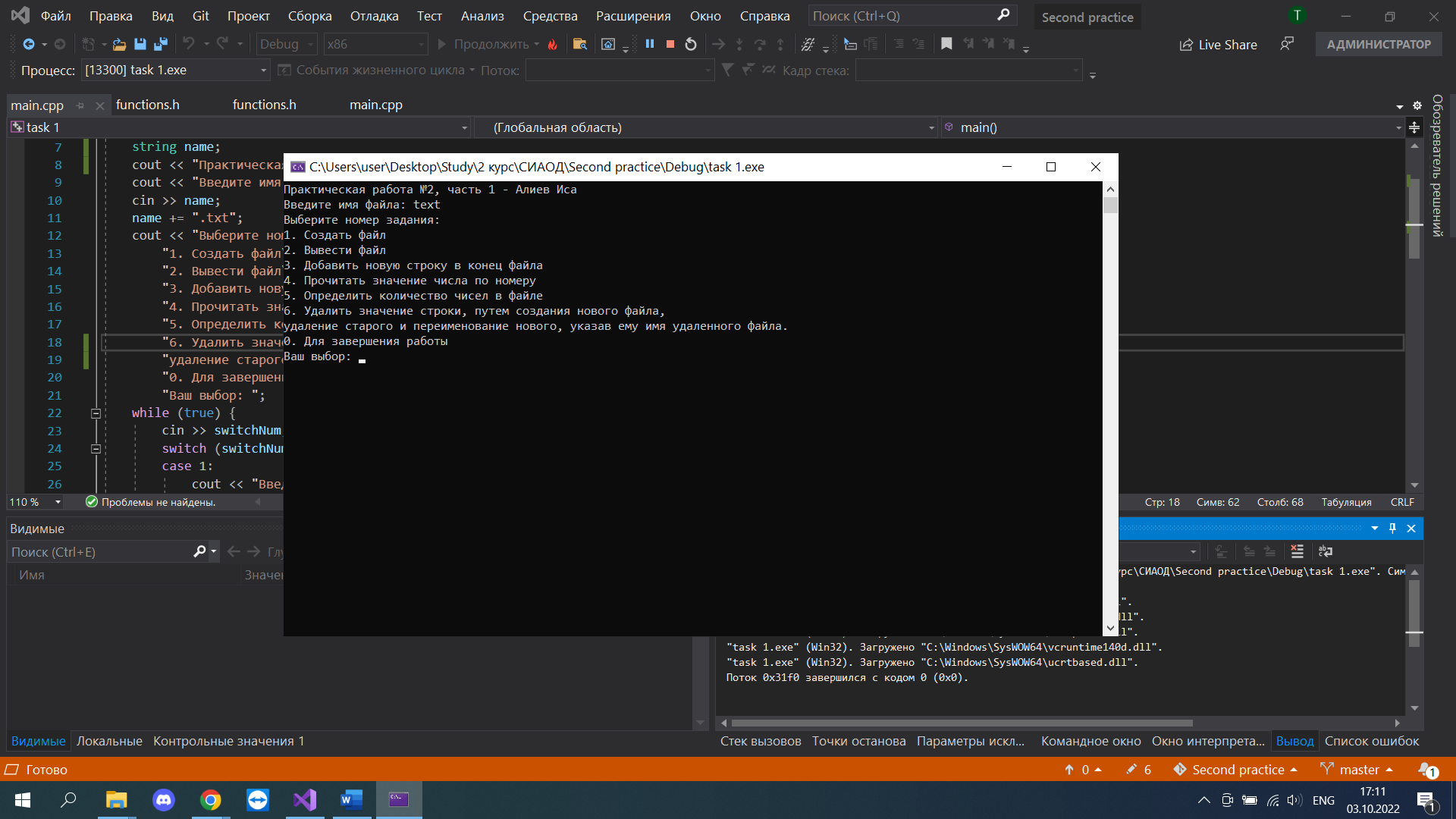


Рисунок 1. Интерфейс программы

Реализуется данный интерфейс с помощью бесконечного цикла while(true) и конструкции switch(switchNum), где параметр switchNum вводится пользователем и отвечает за номер задания. При вводе несуществующего номера программа выдает предупреждение, а при вводе нуля программа завершает работу.

Фрагмент кода:

|  |
| --- |
| while (true)  {  switch (switchNum)  {  case 1:  …  break;  case 2:  …  break;  case 3:  …  break;  case 4:  …  break;  case 5:  …  break;  case 0:  return 0;  default:  cout << "Нет такого номера!\n";  break;  }  cout << "\nВыберите следующий номер задания: ";  cin >> switchNum;  }  } |

# Тестирование

## Задание 1

Для создания файла сначала вводим количество строк, а затем числа каждой строки.

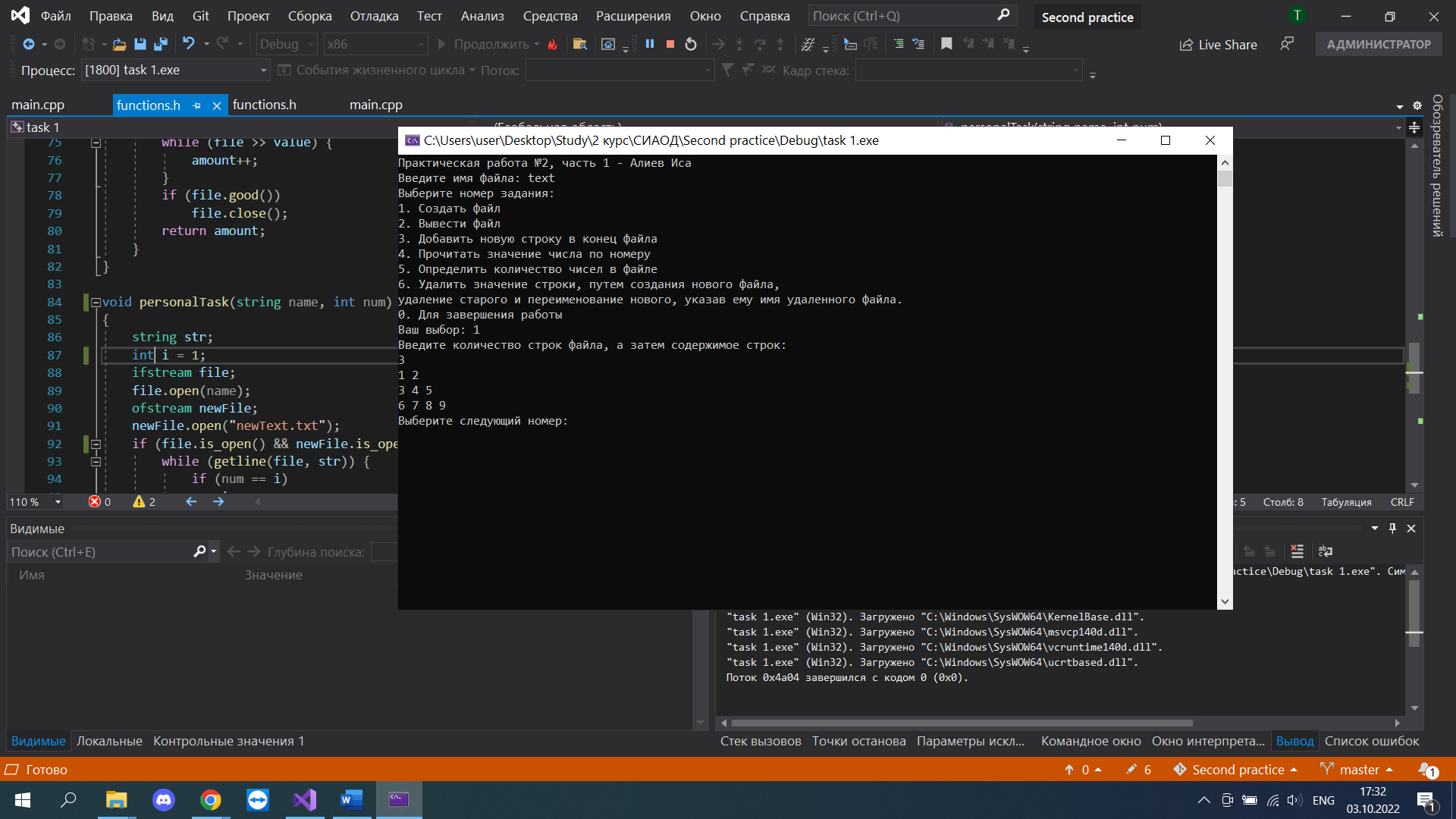


Рисунок 2. – Результат программы задания 1

## Задание 2

В данном задании программа выводит содержимое файла, что нам и требуется.

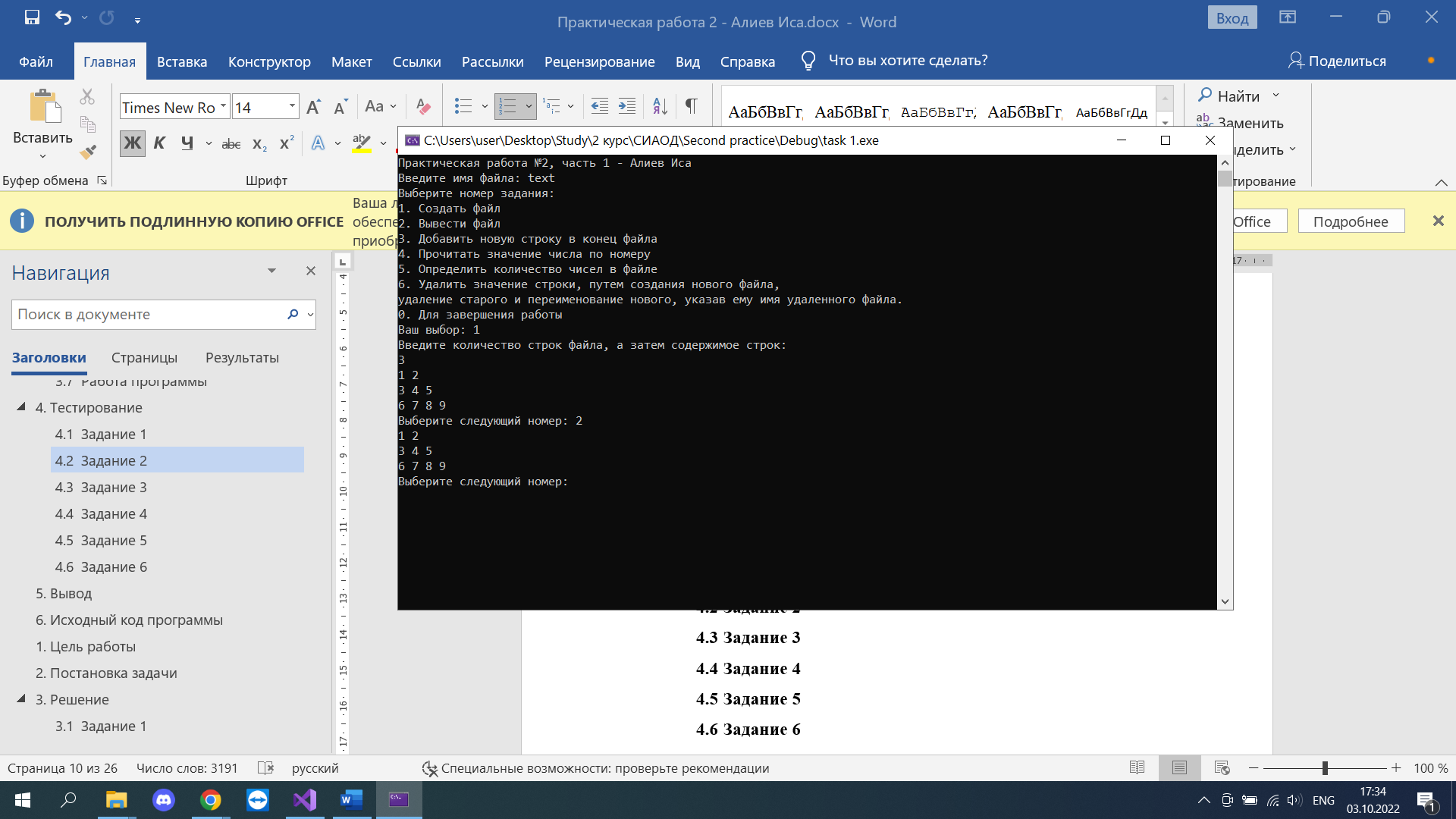


Рисунок 3. – Результат программы задания 2

## Задание 3

Для добавления новой строки просто введем ее содержимое. Далее выведем файл для проверки.

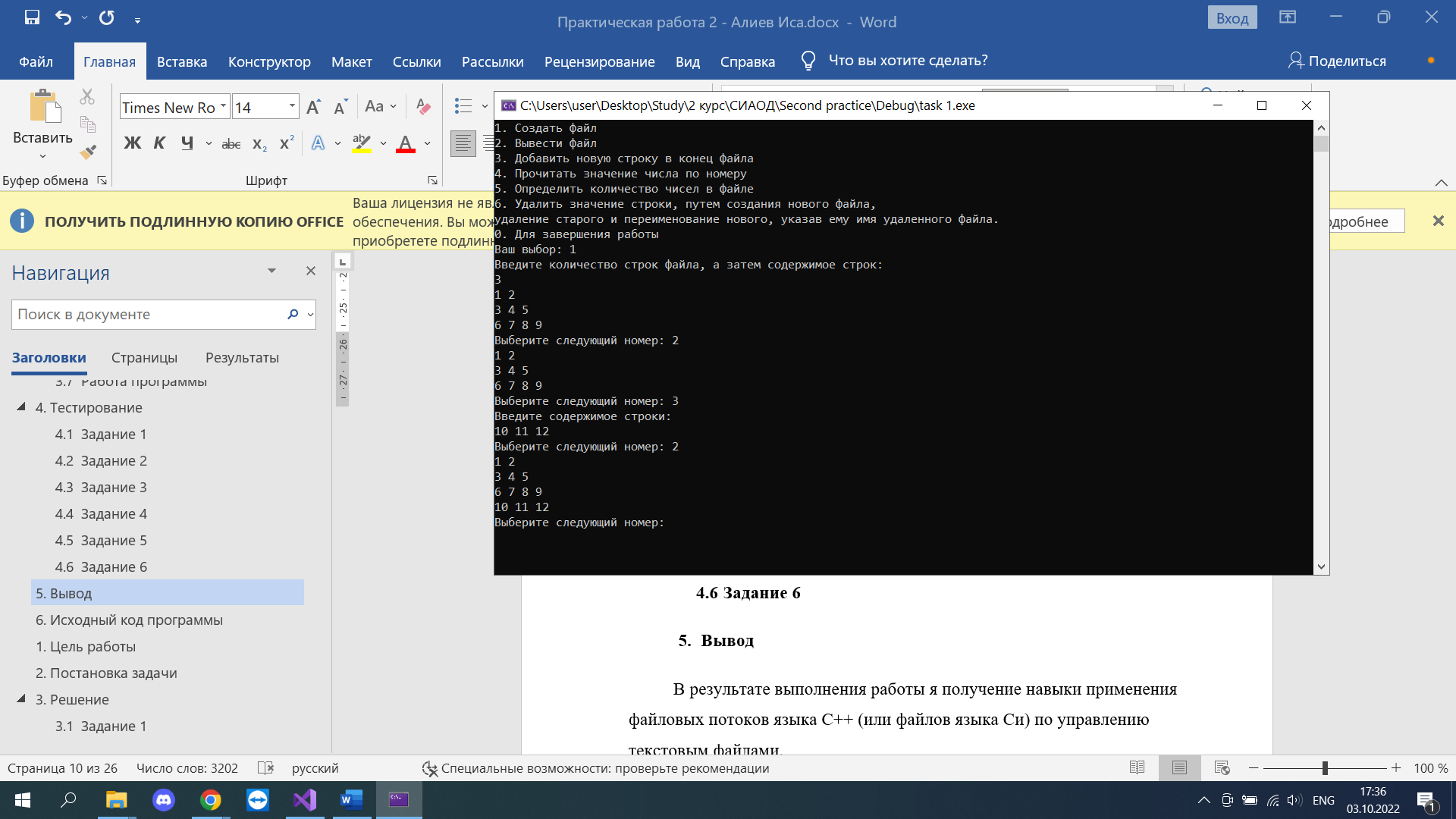


Рисунок 4. – Результат программы задания 3

## Задание 4

Для проверки попробуем получить значение 5-ого числа в файле. В данном случае это число 5.

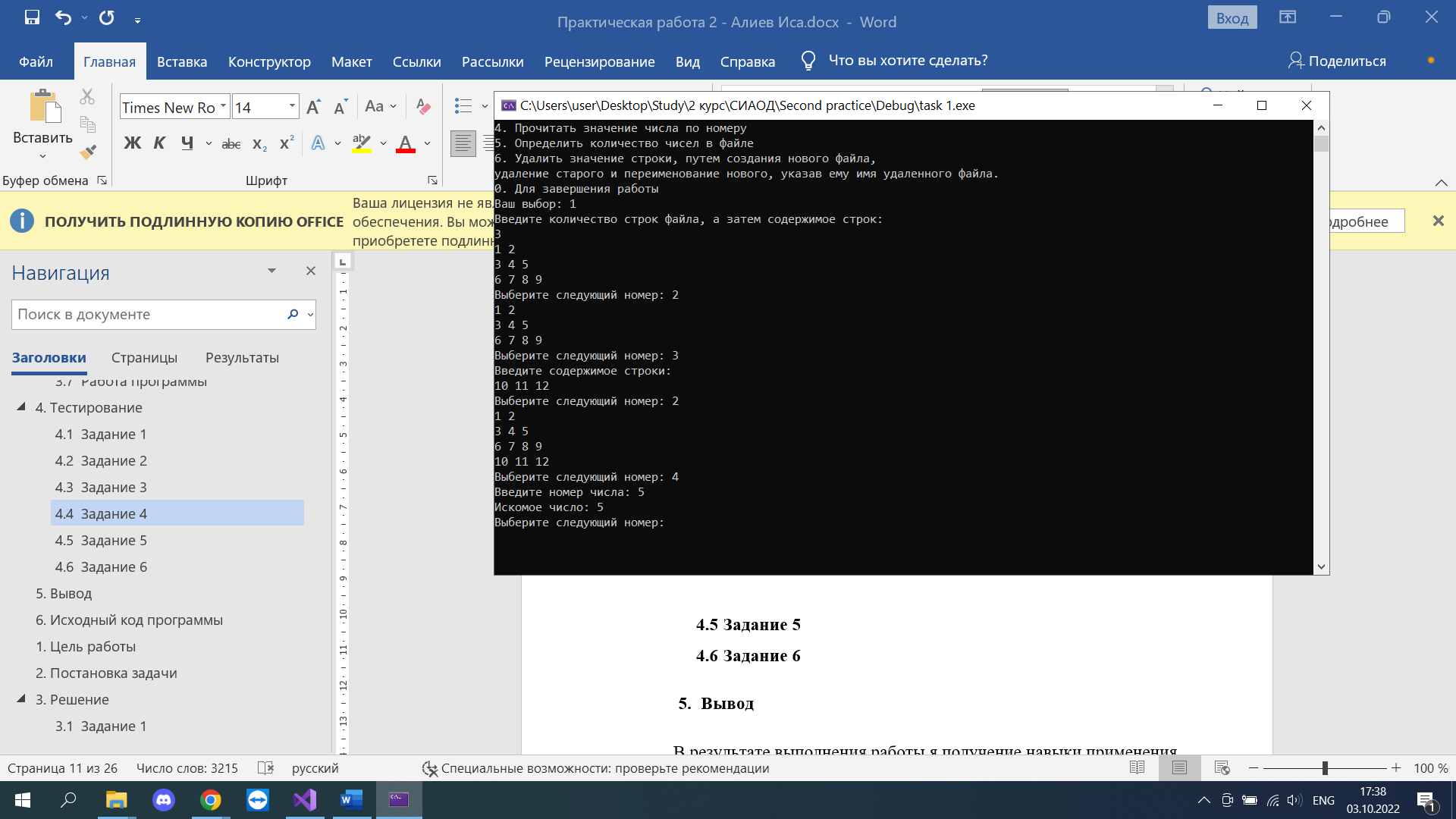


Рисунок 5. – Результат программы задания 4

## Задание 5

Результат программы вывел, что в файле 12 чисел, что верно.

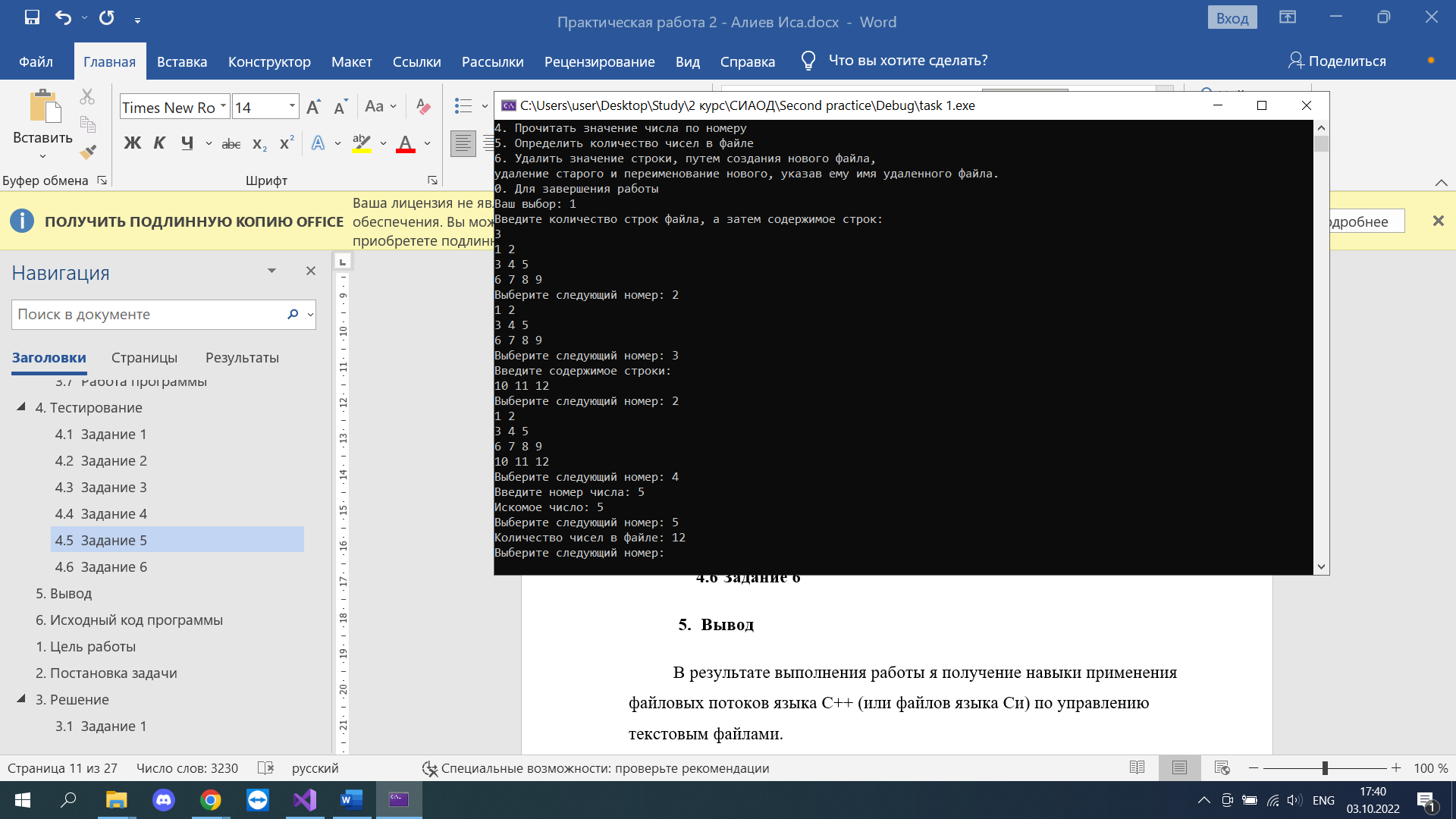


Рисунок 6. – Результат программы задания 5

## Задание 6

В данном задании выберем номер строки, которую хотим удалить. Затем выведем содержимое файла. Видим, что 3-я строка “6 7 8 9” стерлась.

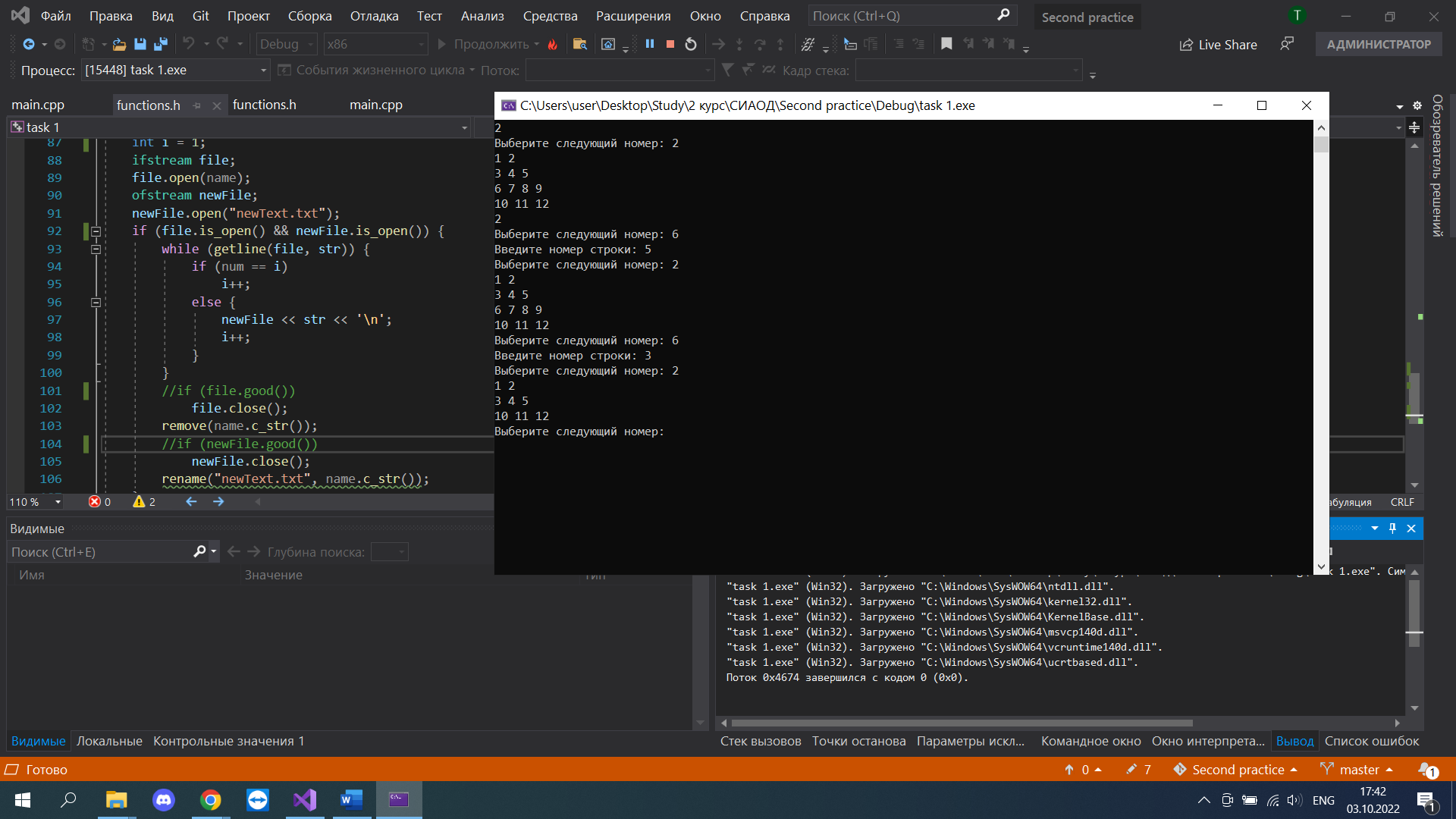


Рисунок 7. – Результат программы задания 6

# Вывод

В результате выполнения работы я получение навыки применения файловых потоков языка С++ (или файлов языка Си) по управлению текстовым файлами.

# Исходный код программы

|  |
| --- |
| int main() {  setlocale(LC\_ALL, "Rus");  int switchNum;  string name;  cout << "Введите имя файла: ";  cin >> name;  name += ".txt";  cout << "Выберите номер задания:\n"  "1. Создать файл\n"  "2. Вывести файл\n"  "3. Добавить новую строку в конец файла\n"  "4. Прочитать значение числа по номеру\n"  "5. Определить количество чисел в файле\n"  "6. Удалить значение строки, путем создания нового файла, удаление старого и переименование нового, указав ему имя удаленного файла.\n"  "0. Для завершения работы\n"  "Ваш выбор: ";  while (true) {  cin >> switchNum;  switch (switchNum) {  case 1:  cout << "Введите количество строк файла, а затем содержимое строк:\n";  createFile(name);  break;  case 2:  outputFile(name);  break;  case 3:  cout << "Введите содержимое строки:\n";  addStr(name);  break;  case 4:  int num;  cout << "Введите номер числа: ";  cin >> num;  cout << "Искомое число: " << getValueByNum(name, num) << '\n';  break;  case 5:  cout << "Количество чисел в файле: " << getAmount(name) << '\n';  break;  case 6:  cout << "Введите номер строки: ";  personalTask(name);  break;  case 0:  return 0;  default:  cout << "Нет такого номера!\n";  break;  }  cout << "Выберите следующий номер: ";  }  } |

**Задание 2**

# Цель работы

Получение навыков применения файловых потоков языка С++ (или файлов языка Си) по управлению текстовым и двоичным файлами.

# Постановка задачи

1. Разработать программу управления двоичными файлами с записями фиксированной длины. Общие требования: файл состоит из записей определенной структуры, согласно варианту. Записи имеют ключ, уникальный в пределах файла.
   1. Разработать структуру записи двоичного файла согласно варианту задания.
   2. Подготовить тестовые данные в текстовом файле с кодировкой ASCII, в соответствии со структурой записи варианта. При открытии файла выполнить контроль его существования и открытия.
   3. Имя файла вводит пользователь.
   4. При открытии файла обеспечить контроль существования и открытия файла.
   5. При применении механизма прямого доступа к записи файла выполнить контроль присутствия записи с заданным номером в файле.
   6. Разработать функции для выполнения операций:
      1. преобразование тестовых данных из текстового файла в двоичный файл;
      2. преобразование данных из двоичного файла в текстовый;
      3. вывод всех записей двоичного файла;
      4. доступ к записи по ее порядковому номеру в файле, используя механизм прямого доступа к записи в двоичном файле;
      5. удаление записи с заданным значением ключа, выполнить путем замены на последнюю запись.
      6. манипулирование записями в двоичном файле согласно дополнительным операциям, определенным в варианте;
   7. Сохранить функции в новом модуле.
   8. Разработать приложение, демонстрирующее выполнение всех операций, подключив к нему модуль с функциями.
   9. Выполнить тестирование приложения, продемонстрировав выполнение всех операций.

Вариант №1. Дополнительное задание:

|  |  |  |
| --- | --- | --- |
| № | Задание | |
| 1 | Структура записи | Читательский абонемент: номер читательского билета – целое пятизначное число, ФИО, адрес. |
| Доп. операция | 1. Поиск записи с заданным значением ключа. 2. Удаление найденного значения. |

# Решение

Двоичные файлы используются для более компактного хранения информации. Хранят данные в машинном формате, т.е. в том виде как они представлены в оперативной памяти. Например, значение вещественной переменной типа float будет записано в файл в формате с плавающей точкой. Под это значение будет отведено столько байт, сколько требуется переменной формата float.

Логически файл состоит из записей фиксированной длины, минимальная длина один байт, максимальная соответствует размеру записанной при создании файла записи. Поэтому из такого файла удобно читать записи: *каким форматом записали, таким и будем читать.*

Создается файл программным путем. Как и текстовый файл имеет последовательную организацию, т.е. новые записи добавляются в конец файла. Обрабатывается файл так же последовательно от первой записи к последней. Для двоичных файлов с записями фиксированной длины применяется метод произвольного доступа, при котором можно обрабатывать отдельную запись: прочитать, изменить и записать на старое место.

Для применения в программе двоичного файла необходимо определить поток, поддерживающий двоичные операции.

Структура, используемая в программе:

|  |
| --- |
| struct readerSub {  unsigned int number;  char name[200];  char address[200];  }; |

## Задание 1

В данном задании необходимо преобразовать данные из текстового файла в двоичный файл. Для этого реализуем функцию convertTxtToBin() в которой в цикле while пока не найден конец файла, будем считывать данные текстового файла в локальную структуру и затем записывать ее поля в бинарный файл методом write().

Код функции:

|  |
| --- |
| void convertTxtToBin(ifstream& textFile, ofstream& binFile, int &subsAmount)  {  subsAmount = 0;  while (!textFile.eof()) {  readerSub r;  string name, address;  textFile >> r.number;  textFile.get();  getline(textFile, name);  getline(textFile, address);  strcpy\_s(r.name, name.c\_str());  strcpy\_s(r.address, address.c\_str());  binFile.write((char\*)&r, sizeof(readerSub));  subsAmount += 1;  }  } |

В функции main открываем текстовый и бинарные файлы и передаем их адреса в функцию.

Фрагмент кода из main:

|  |
| --- |
| textFileIf.open("subs.txt", ios::out);  binFileOf.open(binFileName, ios::out | ios::binary);  if (!textFileIf || !binFileOf)  {  cout << "файл не открыт";  return 1;  }  convertTxtToBin(textFileIf, binFileOf, subsAmount);  cout << "Файл преобразован!\n";  textFileIf.close();  binFileOf.close(); |

## Задание 2

В данном задании необходимо преобразовать данные из двоичного файла в текстовый. Для этого реализуем функцию convertBinToTxt(), в которой в цикле while методом read() читаем данные из бинарного файла в локальную структуру, затем записываем поля структуры в текстовый файл.

Код функции:

|  |
| --- |
| void convertBinToTxt(ofstream& textFile, ifstream& binFile)  {  readerSub r;  binFile.read((char\*)&r, sizeof(readerSub));  while (binFile.good()) {  textFile << r.number << '\n' << r.name << '\n' << r.address;  binFile.read((char\*)&r, sizeof(readerSub));  if (!binFile.eof())  textFile << '\n';  }  } |

В функции main открываем текстовый и бинарные файлы и передаем их адреса в функцию.

Фрагмент кода из main:

|  |
| --- |
| binFileIf.open(binFileName, ios::in | ios::binary);  textFileOf.open("subs.txt");  if (!binFileIf || !textFileOf)  {  cout << "файл не открыт";  return 1;  }  convertBinToTxt(textFileOf, binFileIf);  textFileOf.close();  binFileIf.close();  cout << "Файл преобразован!\n"; |

## Задание 3

В данном задании необходимо вывести все записи двоичного файла. Для этого реализуем функцию outputBinFile(), в которой в цикле while методом read() считываем данные бинарного файла в локальную структуру, затем выводим ее поля в консоль.

Код функции:

|  |
| --- |
| void outputBinFile(ifstream& binFile)  {  readerSub r;  binFile.read((char\*)&r, sizeof(readerSub));  while (!binFile.eof()) {  cout << r.number << " " << r.name << " " << r.address << '\n';  binFile.read((char\*)&r, sizeof(readerSub));  }  } |

В функции main открываем бинарные файл и передаем его адрес в функцию.

Фрагмент кода из main:

|  |
| --- |
| binFileIf.open(binFileName, ios::in | ios::binary);  if (!binFileIf)  {  cout << "файл не открыт";  return 1;  }  outputBinFile(binFileIf);  binFileIf.close(); |

## Задание 4

В данном задании необходимо получить доступ к записи по ее порядковому номеру в файле, используя механизм прямого доступа к записи в двоичном файле. Для этого реализуем функцию getSubByNum(), в которой с помощью метода seekg() перемещаем указатель на нужную нам запись, читаем ее методом read() в локальную структуру и возвращаем результат как строку.

Код функции:

|  |
| --- |
| string getSubByNum(ifstream& binFile, int num)  {  readerSub r;  string value = "";  num -= 1;  binFile.seekg(sizeof(r) \* num, ios::beg);  binFile.read((char\*)&r, sizeof(readerSub));  value += to\_string(r.number) + " " + r.name + " " + r.address;  return value;  } |

В функции main открываем бинарные файл и передаем его адрес в функцию.

Фрагмент кода из main:

|  |
| --- |
| binFileIf.open(binFileName, ios::in | ios::binary);  if (!binFileIf)  {  cout << "файл не открыт";  return 1;  }  int num;  cout << "Введите номер записи: ";  cin >> num;  if (num > subsAmount || num <= 0)  cout << "Данного номера нет в файле\n";  else {  cout << getSubByNum(binFileIf, num) << '\n';  binFileIf.close();  } |

## Задание 5

В данном задании необходимо удалить записи с заданным значением ключа, путем замены на последнюю запись. Для этого реализуем функцию delSubByKeyWithReplace(), в которой мы читаем последнюю запись в локальную структуру r1 с помощью метода seekg() и read(). Далее в цикле while мы читаем каждую запись бинарного файла и, если ключ записи совпадает с параметром key, записываем в новый бинарный методом write() файл стуктуру r1, иначе, если ключ r и r1 не совпадают, записываем ее в новый файл. В конце старый бинарный файл удаляем, а новый называем его именем.

Код функции:

|  |
| --- |
| string delSubByKeyWithReplace(ifstream& binFile, int key, string binFileName, int subsAmount)  {  readerSub r, r1;  ofstream tempBinFile;  bool check = false;  tempBinFile.open("tempBin.dat", ios::out | ios::binary);  binFile.seekg(sizeof(r) \* (subsAmount - 1), ios::beg);  binFile.read((char\*)&r1, sizeof(readerSub));  binFile.seekg(sizeof(r) \* 0, ios::beg); binFile.read((char\*)&r, sizeof(readerSub));  while (binFile.good())  {  if (r.number == key)  {  check = true;  tempBinFile.write((char\*)&r1, sizeof(readerSub));  }  else if (r.number != r1.number)  tempBinFile.write((char\*)&r, sizeof(readerSub));  binFile.read((char\*)&r, sizeof(readerSub));  }  binFile.close();  remove(binFileName.c\_str());  tempBinFile.close();  rename("tempBin.dat", binFileName.c\_str());  binFile.open(binFileName, ios::in | ios::binary);  if (check)  return "Запись удалена";  else  return "Данного ключа нет в файле";  } |

В функции main открываем бинарные файл и передаем его адрес в функцию.

Фрагмент кода из main:

|  |
| --- |
| binFileIf.open(binFileName, ios::in | ios::binary);  if (!binFileIf)  {  cout << "файл не открыт";  return 1;  }  int key;  cout << "Введите ключ: ";  cin >> key;  cout << delSubByKeyWithReplace(binFileIf, key, binFileName, subsAmount) << '\n';  binFileIf.close();  subsAmount -= 1; |

## Задание 6

В данном задании необходимо найти записи по ключу. Для этого реализуем функцию getSubByKey(), в которой в цикле while методом read() прочитаем запись из бинарного файла в локальную структуру. Далее если ключ структуры совпадет с параметром key, вернем запись как строку.

Код функции:

|  |
| --- |
| string getSubByKey(ifstream& binFile, int key)  {  readerSub r;  string value;  while (binFile.good()) {  binFile.read((char\*)&r, sizeof(readerSub));  if (r.number == key)  {  value += to\_string(r.number) + " " + r.name + " " + r.address;  return value;  }  }  return "Данного ключа нет в файле";  } |

В функции main открываем бинарные файл и передаем его адрес в функцию.

Фрагмент кода из main:

|  |
| --- |
| binFileIf.open(binFileName, ios::in | ios::binary);  if (!binFileIf)  {  cout << "файл не открыт";  return 1;  }  cout << "Введите ключ: ";  cin >> key;  cout << getSubByKey(binFileIf, key) << '\n';  binFileIf.close(); |

## Задание 7

В данном задании необходимо удалить запись по ключу. Для этого реализуем функцию delSubByKey(), в которой в цикле while методом read() прочитаем запись из бинарного файла в локальную структуру. Далее если ключ структуры не совпадет с параметром key, записываем ее в новый бинарный файл методом write(). В конце старый бинарный файл удаляем, а новый называем его именем.

Код функции:

|  |
| --- |
| string delSubByKey(ifstream& binFile, int key, string binFileName)  {  readerSub r;  ofstream tempBinFile;  bool check = false;  tempBinFile.open("tempBin.dat", ios::out | ios::binary);  binFile.read((char\*)&r, sizeof(readerSub));  while (binFile.good()) {  if (r.number == key)  check = true;  else  tempBinFile.write((char\*)&r, sizeof(readerSub));  binFile.read((char\*)&r, sizeof(readerSub));  }  binFile.close();  remove(binFileName.c\_str());  tempBinFile.close();  rename("tempBin.dat", binFileName.c\_str());  if (check)  return "Запись удалена";  else  return "Данного ключа нет в файле";  } |

В функции main открываем бинарные файл и передаем его адрес в функцию.

Фрагмент кода из main:

|  |
| --- |
| binFileIf.open(binFileName, ios::in | ios::binary);  if (!binFileIf)  {  cout << "файл не открыт";  return 1;  }  cout << "Введите ключ: ";  cin >> key;  cout << delSubByKey(binFileIf, key, binFileName) << '\n';  binFileIf.close();  subsAmount -= 1; |

## Работа программы

Работа программы производится с помощью специально разработанного интерфейса, в котором выводятся: номер работы, фамилия и имя студента. Далее, после того как вводится имя файла, выводятся последовательно номера заданий с условиями. Затем пользователю просто необходимо ввести номер для выбора соответствующего задания.

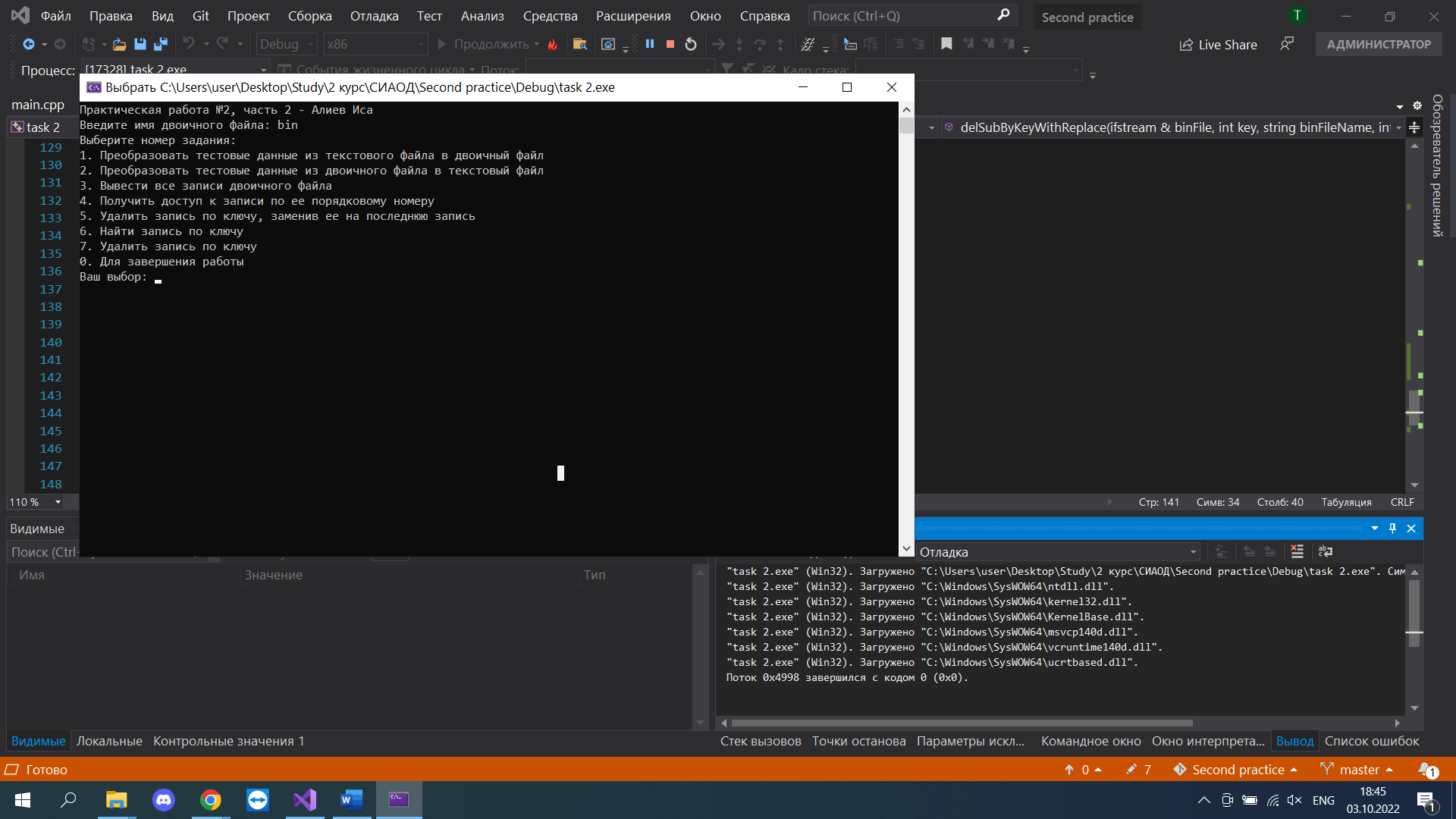


Рисунок 8. – Интерфейс программы

Реализуется данный интерфейс с помощью бесконечного цикла while(true) и конструкции switch(switchNum), где параметр switchNum вводится пользователем и отвечает за номер задания. При вводе несуществующего номера программа выдает предупреждение, а при вводе нуля программа завершает работу.

Фрагмент кода:

|  |
| --- |
| while (true)  {  switch (switchNum)  {  case 1:  …  break;  case 2:  …  break;  case 3:  …  break;  case 4:  …  break;  case 5:  …  break;  case 0:  return 0;  default:  cout << "Нет такого номера!\n";  break;  }  cout << "\nВыберите следующий номер задания: ";  cin >> switchNum;  }  } |

# Тестирование

## Задание 1

В данном задании файл преобразуется из текстового формата в бинарный. При удачном преобразовании выводится соответствующая надпись.

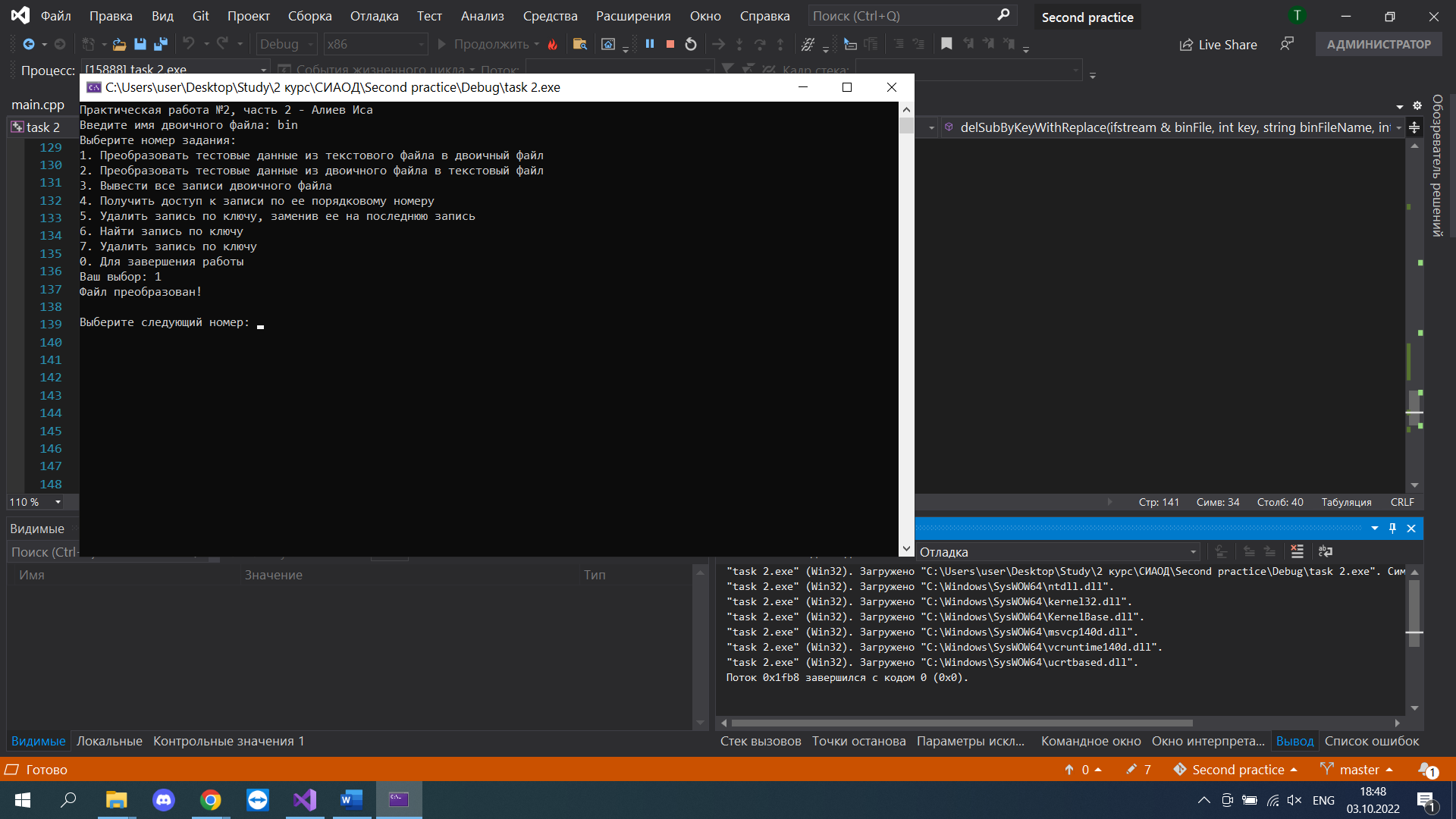


Рисунок 9. – Результат программы задания 1

## Задание 2

В данном задании файл преобразуется из бинарного формата в текстовый. При удачном преобразовании выводится соответствующая надпись.

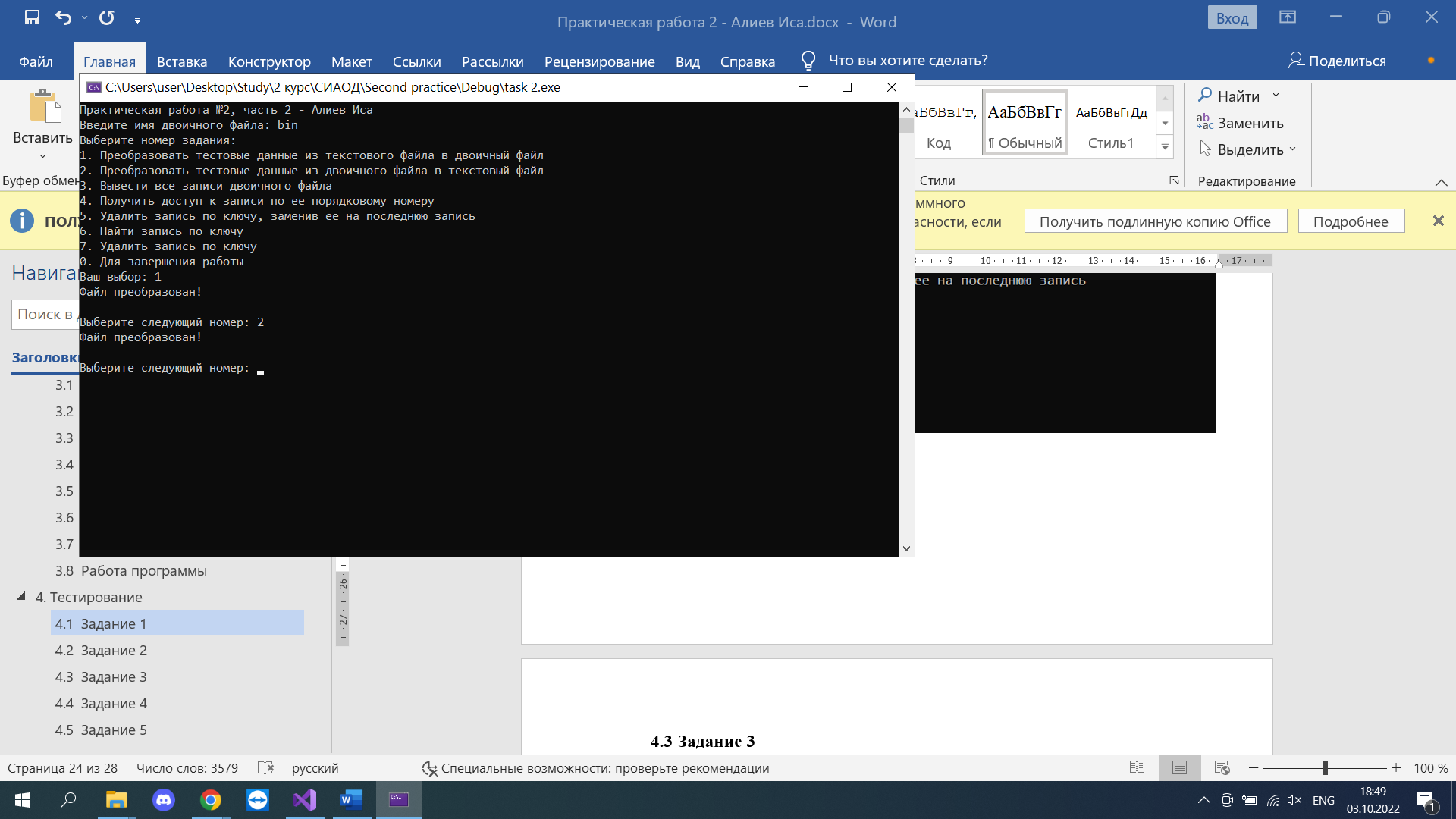


Рисунок 10. – Результат программы задания 2

## Задание 3

В данном задании выводятся все записи двоичного файла, что нам и требуется.

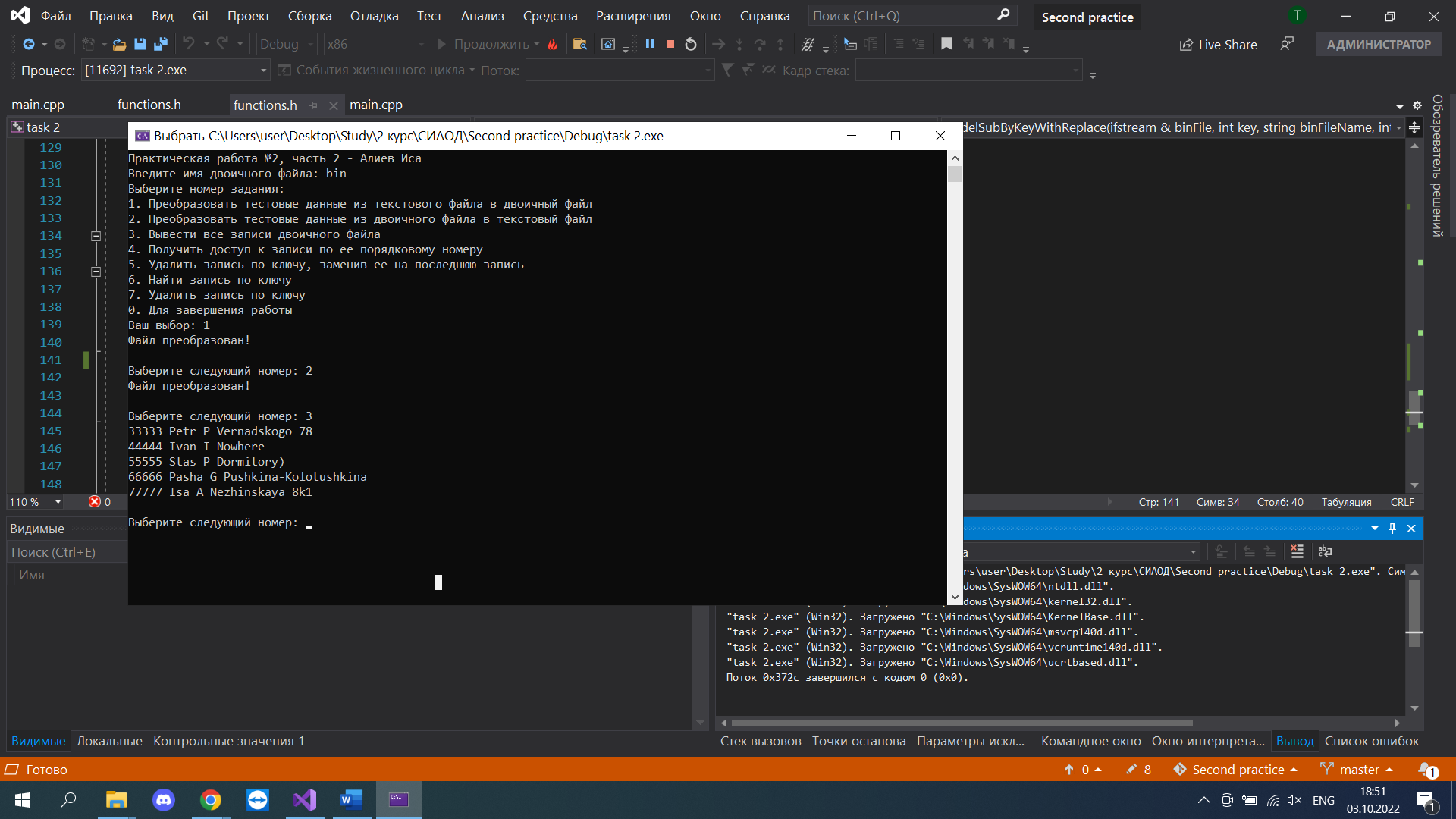


Рисунок 11. – Результат программы задания 3

## Задание 4

В данном задании выводится информация записи по ее порядковому номеру. Причем если указать некорректный номер, выдается предупреждение.

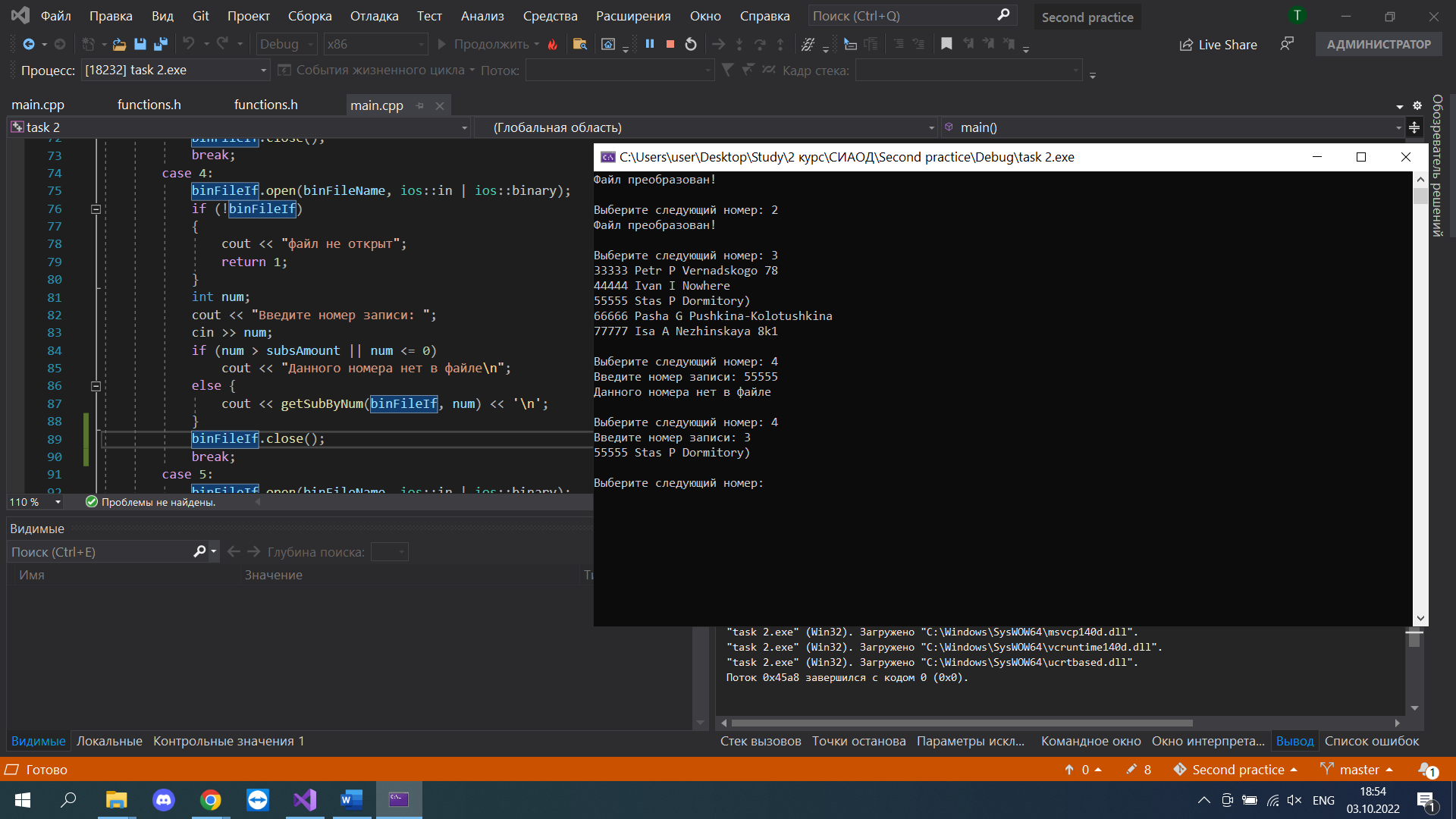


Рисунок 12. – Результат программы задания 4

## Задание 5

В данном задании удаляется запись по ключу, путем замены на последнюю запись. После удаления выведем данные файла для проверки. Как видим, запись с ключом 44444 стерлась.

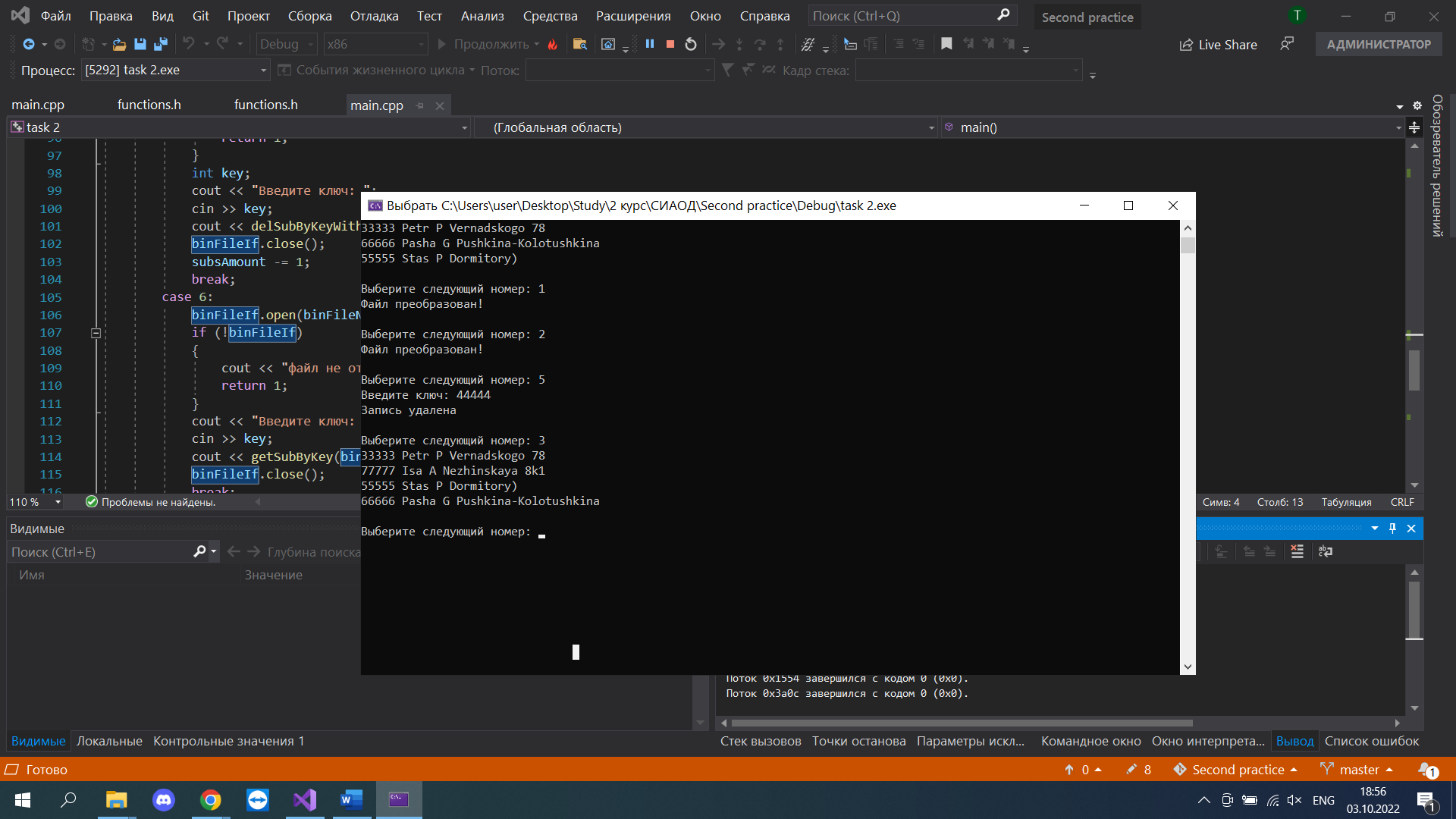


Рисунок 13. – Результат программы задания 5

## Задание 6

В данном задании выводится запись по ключу. Как видим из программы по ключу 77777 вывелась соответствующая запись.

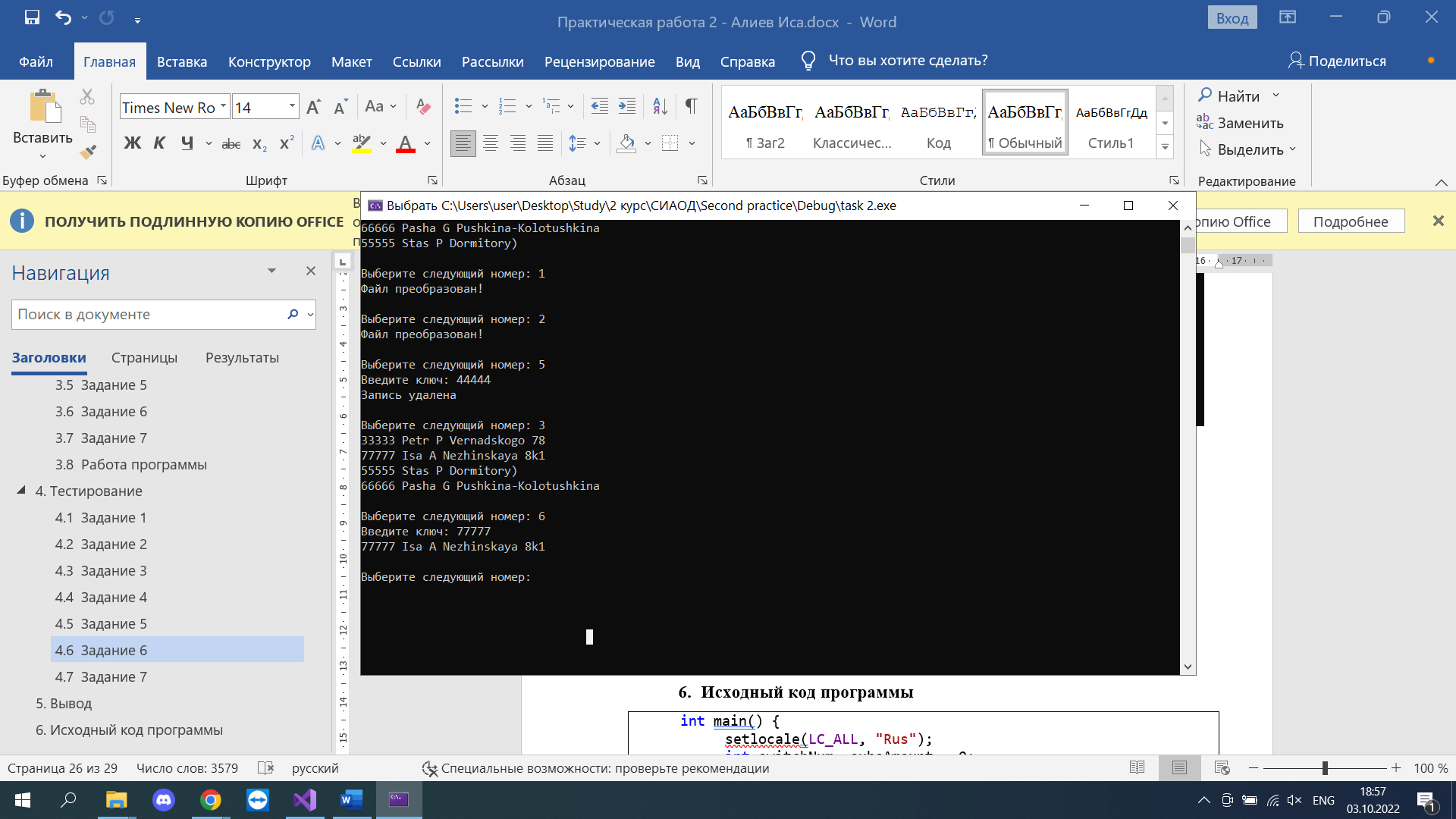


Рисунок 14. – Результат программы задания 6

## Задание 7

В данном задании удаляется запись по ключу. Как видим если ввести ключ 55555, а затем вывести список записей бинарного файла, соответствующая запись стирается.

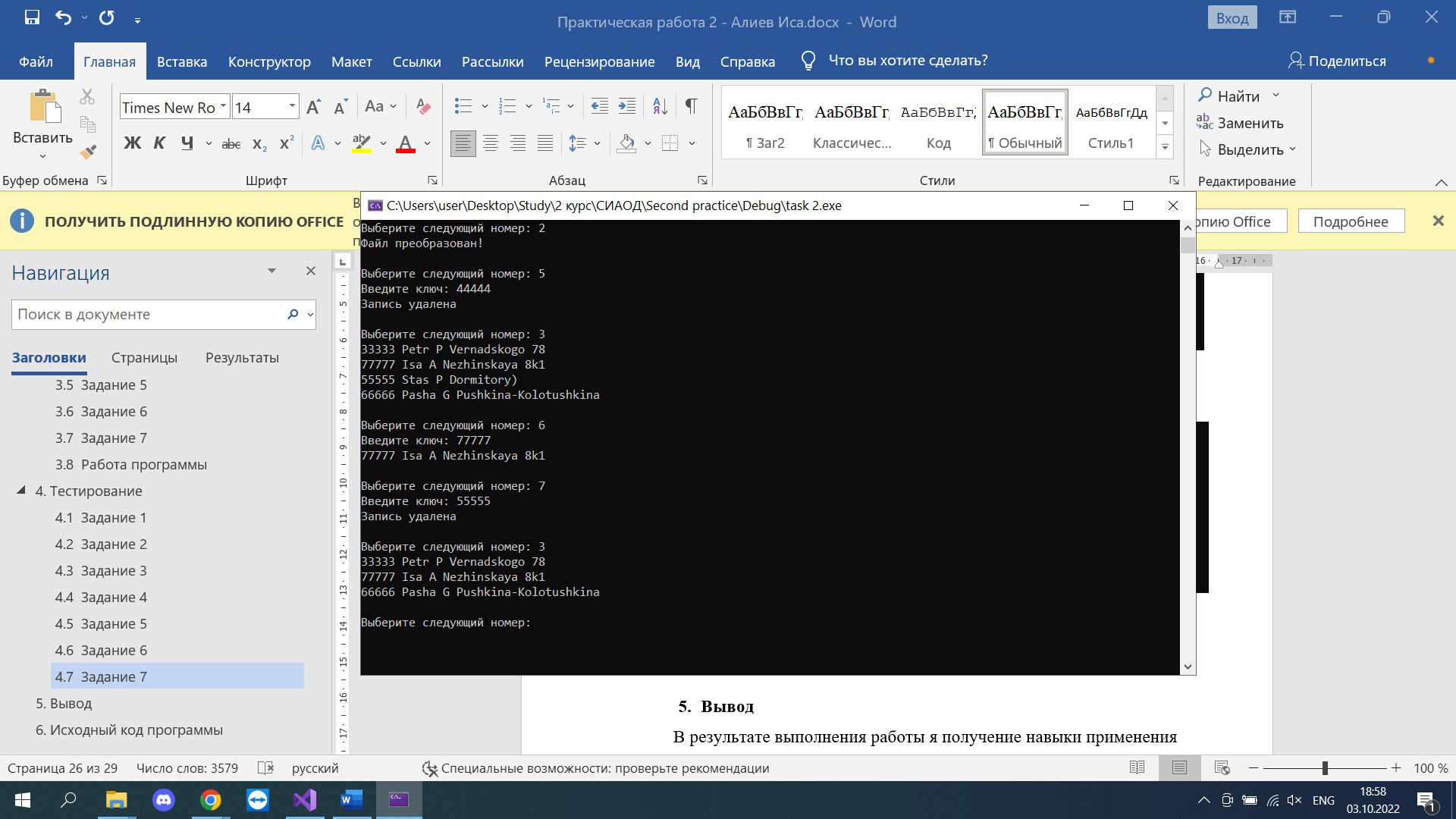


Рисунок 15. – Результат программы задания 7

# Вывод

В результате выполнения работы я получение навыки применения файловых потоков языка С++ (или файлов языка Си) по управлению бинарными файлами.

# Исходный код программы

|  |
| --- |
| int main() {  setlocale(LC\_ALL, "Rus");  int switchNum, subsAmount = 0;  ifstream textFileIf, binFileIf;  ofstream binFileOf, textFileOf;  string binFileName;  cout << "Введите имя двоичного файла: ";  getline(cin, binFileName);  binFileName += ".dat";  binFileOf.open(binFileName, ios::out | ios::binary);  if (!binFileOf)  {  cout << "файл не открыт";  return 1;  }  binFileOf.close();  cout << "Выберите номер задания:\n"  "1. Преобразовать тестовые данные из текстового файла в двоичный файл\n"  "2. Преобразовать тестовые данные из двоичного файла в текстовый файл\n"  "3. Вывести все записи двоичного файла\n"  "4. Получить доступ к записи по ее порядковому номеру\n"  "5. Удалить запись по ключу, заменив ее на последнюю запись\n"  "6. Найти запись по ключу\n"  "7. Удалить запись по ключу\n"  "0. Для завершения работы\n"  "Ваш выбор: ";  while (true) {  cin >> switchNum;  switch (switchNum) {  case 1:  textFileIf.open("subs.txt", ios::out);  binFileOf.open(binFileName, ios::out | ios::binary);  if (!textFileIf || !binFileOf)  {  cout << "файл не открыт";  return 1;  }  convertTxtToBin(textFileIf, binFileOf, subsAmount);  cout << "Файл преобразован!\n";  textFileIf.close();  binFileOf.close();  break;  case 2:  binFileIf.open(binFileName, ios::in | ios::binary);  textFileOf.open("subs.txt");  if (!binFileIf || !textFileOf)  {  cout << "файл не открыт";  return 1;  }  convertBinToTxt(textFileOf, binFileIf);  textFileOf.close();  binFileIf.close();  cout << "Файл преобразован!\n";  break;  case 3:  binFileIf.open(binFileName, ios::in | ios::binary);  if (!binFileIf)  {  cout << "файл не открыт";  return 1;  }  outputBinFile(binFileIf);  binFileIf.close();  break;  case 4:  binFileIf.open(binFileName, ios::in | ios::binary);  if (!binFileIf)  {  cout << "файл не открыт";  return 1;  }  int num;  cout << "Введите номер записи: ";  cin >> num;  if (num > subsAmount || num <= 0)  cout << "Данного номера нет в файле\n";  else {  cout << getSubByNum(binFileIf, num) << '\n';  binFileIf.close();  }  break;  case 5:  binFileIf.open(binFileName, ios::in | ios::binary);  if (!binFileIf)  {  cout << "файл не открыт";  return 1;  }  int key;  cout << "Введите ключ: ";  cin >> key;  cout << delSubByKeyWithReplace(binFileIf, key, binFileName, subsAmount) << '\n';  binFileIf.close();  subsAmount -= 1;  break;  case 6:  binFileIf.open(binFileName, ios::in | ios::binary);  if (!binFileIf)  {  cout << "файл не открыт";  return 1;  }  cout << "Введите ключ: ";  cin >> key;  cout << getSubByKey(binFileIf, key) << '\n';  binFileIf.close();  break;  case 7:  binFileIf.open(binFileName, ios::in | ios::binary);  if (!binFileIf)  {  cout << "файл не открыт";  return 1;  }  cout << "Введите ключ: ";  cin >> key;  cout << delSubByKey(binFileIf, key, binFileName) << '\n';  binFileIf.close();  subsAmount -= 1;  break;  break;  case 0:  return 0;  default:  cout << "Нет такого номера!\n";  break;  }  cout << "\nВыберите следующий номер: ";  }  } |