|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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 № 2** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Внешние структуры данных: текстовый и двоичный файлы»** | |
|  | |
| Выполнил студент группы ИКБО-06-21 | Школьник Т.О. |
| 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++ по управлению текстовым и двоичным файлами.

# Постановка задачи

1. Разработать программу, управления текстовым файлом.
   1. Реализация ввода-вывода на основе файловых потоков С++: ofstream, ifstream.
   2. Имя физического файла вводится пользователем и передается в функции, обработка 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 № 28. Дополнительная операция:

|  |
| --- |
| Создать два новых файла из значений исходного, переписав в один из них первую половину чисел исходного, а в другой, оставшуюся часть. В исходный файл слить данные их двух новых файлов упорядоченными по возрастанию парами, т.е. прочитать первые числа двух файлов, сначала в исходный файл записать меньшее из них, а за ним большее. |

# Решение

Язык программирования C++ предоставляет несколько классов для файловых операций ввода-вывода, включая ofstream, ifstream и fstream.

ofstream является производным от класса ostream. Он обеспечивает выходной поток для работы с файлом. Объекты потока вывода можно использовать для записи последовательностей символов в файл.

ifstream является производным от класса istream. Он обеспечивает входной поток для работы с файлом. Мы можем использовать этот входной поток для чтения из файла.

Fstream является производным от класса iostream, а iostream является производным от классов istream и ostream. Он обеспечивает как входные, так и выходные потоки для работы с файлом. Если наш объект потока принадлежит классу fstream, мы можем выполнять операции чтения и записи в файле с тем же объектом потока.

Режимы открытия

| **Режим** | **Синтаксис** | **Описание** |
| --- | --- | --- |
| Читать | ios::in | Открывает файл для чтения. |
| Написать | ios::out | Открывает файл для записи. |
| Бинарный | ios::binary | Все операции будут выполняться в бинарном режиме. |
| Усекать перед открытием | ios::trunc | Если файл уже существует, все содержимое будет немедленно удалено. |
| Добавить | ios::app | Все предоставленные данные будут добавлены в связанный файл. |
| В конце | ios::ate | Он открывает файл и перемещает элемент управления чтением/записью в конец файла. Основное различие между ios::app и этим в том, что первый всегда будет начинать запись с конца, но с этим мы можем искать любую конкретную позицию. |

Модуль функций:

1. Нужно создать текстовый файл кодировки ASCII, содержащего десятичные числа по несколько чисел на строке. Заполним случайными числами файл с помощью вложенных циклов и функции случайного заполнения.

|  |
| --- |
| // Функция создания текстового файла кодировки ASCII,  // содержащего десятичные числа по несколько чисел на строке  void createFileWithRandomNumbers(string &fileName) {  srand(time(NULL));  ofstream file;  file.open(fileName + ".txt");  if (file.is\_open()) {  int maxL = rand() % 10 + 1; // +1 чтобы была хоть 1 строка  for (int line = 0; line < maxL; line++) {  int maxN = rand() % 10 + 1; // +1, чтобы не было пустых строк  for (int number = 0; number < maxN; number++) {  file << rand() % 100 + 1;  if (number != (maxN - 1)) {  file << " ";  }  }  if (line != (maxL-1)) {  file << endl;  }  }  if (file.good()) {  file.close();  }  }  } |

1. Нужно вывести содержание файла. Применяя функцию получения строки, выведем содержимое файла, используя поток вывода.

|  |
| --- |
| // Функция вывода содержимого текстового файла  void print(string &fileName) {  ifstream file;  file.open(fileName + ".txt");  string line = "";  if (file.is\_open()) {  while (file.good()) {  getline(file, line);  cout << line << endl;  }  file.close();  }  } |

1. Нужно добавить в конец файла строку, введенную пользователем. Используем функцию открытия файла с параметром ios::app (append) и добавим переданную строку в конец файла.

|  |
| --- |
| // Функция добавления новой строки в конец файла  void addLine(string &additionLine, string &fileName) {  ofstream file;  file.open(fileName + ".txt", ios::app);  if (file.is\_open()) {  file << endl << additionLine;  if (file.good()) {  file.close();  }  }  } |

1. Нужно прочитать значение числа, указав его порядковый номер в файле, и вернуть его значение. Пройдёмся циклом по файлу до нужного порядкового номера и выведем число, стоящее на этой позиции.

|  |
| --- |
| // Функция читает значение числа, указывает его порядковый номер в файле, и возвращает его значение  int readNumber(string &fileName, const int &index) {  if (index < 0) {  cout << "Неправильный порядковый номер" << endl;  } else {  int resultNumber;  ifstream file;  file.open(fileName + ".txt");  if (file.is\_open()) {  for (int i = 0; i < index; i++) {  file >> resultNumber;  }  if (!file.good()) {  cout << "Неправильный порядковый номер" << endl;  }  file.close();  return resultNumber;  }  }  } |

1. Нужно вывести количество чисел в файле. Читая каждое число в цикле будем добавлять единицу в специальную переменную пока файл не закончится.

|  |
| --- |
| // Функция определения количества чисел в файле  int numbersCount(string &fileName) {  ifstream file;  int counter = 0;  string number;  file.open(fileName + ".txt");  if (file.is\_open()) {  while (file.good()) {  file >> number;  counter++;  }  file.close();  }  return counter;  } |

1. Нужно создать два новых файла из значений исходного, поделённых пополам и слитых в исходный упорядоченными по возрастанию парами. Пока идём до середины исходного файла, считываем числа в файл А, остальное – в файл Б. Дальше читаем числа из файлов А и Б под соответствующими номерами и записываем в единый файл сначала наименьшее из двух чисел, затем наибольшее.

|  |
| --- |
| // Функция создания нового файла из значений исходного,  // поделённых пополам на два файла и слитых в исходный упорядоченными по возрастанию парами  void additionalOperation(string &fileName) {  int flag = 0;  ifstream file;  file.open(fileName + ".txt");  int sourceFileSize = numbersCount(fileName);  int halfSourceFileSize = floor((sourceFileSize / 2));  string fileNameA = "fileA";  string fileNameB = "fileB";  int number;  ofstream fileB;  ofstream fileA;  fileA.open(fileNameA + ".txt");  // Разделяем исходный файл на два  if (file.is\_open()) {  for (int i = 0; i < sourceFileSize; i++) {  if (i < halfSourceFileSize) {  file >> number;  fileA << number << endl;  } else {  if (flag == 0) {  fileA.close();  fileB.open(fileNameB + ".txt");  flag = 1;  }  file >> number;  fileB << number << endl;  }  }  if (fileB.good()) {  fileB.close();  if (file.good()) {  file.close();  }  }  }  int numberFromA;  int numberFromB;  ifstream forReadFileB;  ifstream forReadFileA;  ofstream fileForRecord;  fileForRecord.open(fileName + "Updated" + ".txt");  forReadFileA.open(fileNameA + ".txt");  forReadFileB.open(fileNameB + ".txt");  flag = 0;  // Вливаем числа из двух файлов в исходный файл упорядоченными парами  if (fileForRecord.is\_open()) {  for (int i = 0; i < sourceFileSize - halfSourceFileSize; i++) {  if (i < halfSourceFileSize) {  forReadFileA >> numberFromA;  forReadFileB >> numberFromB;  if (numberFromA < numberFromB) {  fileForRecord << numberFromA << " ";  fileForRecord << numberFromB << endl;  } else {  fileForRecord << numberFromB << " ";  fileForRecord << numberFromA << endl;  }  } else {  if (flag == 0) {  forReadFileA.close();  flag = 1;  }  forReadFileB >> numberFromB;  fileForRecord << numberFromB << endl;  }  }  if (forReadFileB.good()) {  forReadFileB.close();  if (fileForRecord.good()) {  fileForRecord.close();  }  }  }  } |
|  |

# Тестирование

На старте программы появляется главное меню, где нужно выбрать номер интересующего задания (Рисунок 1.1).

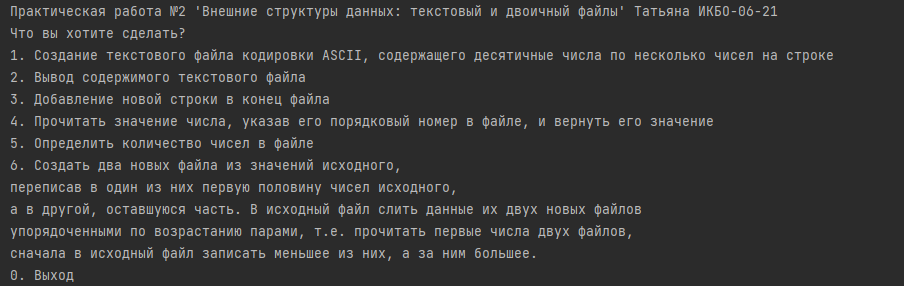


Рисунок 1.1. Главное меню программы

Протестируем создание файла со случайными числами и вывод содержимого файла (Рисунок 1.2).

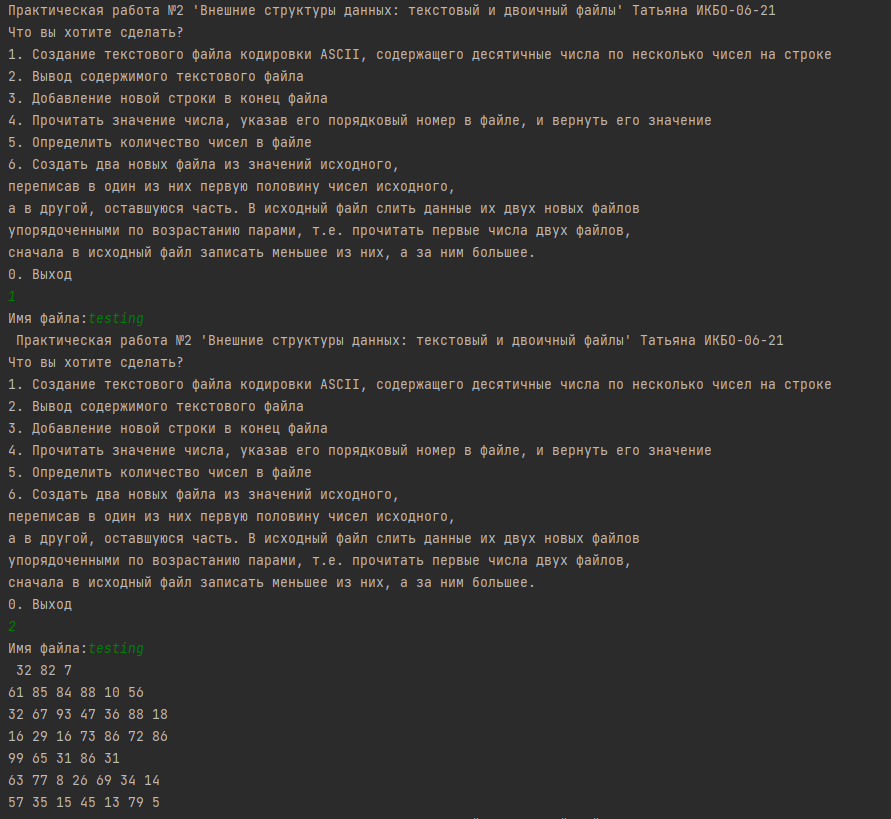


Рисунок 1.2. Тестирование создания файла со случайными числами и вывода содержимого файла

Протестируем добавление строки, введённой пользователем в конец файла. В наш исходный файл:  
32 82 7

61 85 84 88 10 56

32 67 93 47 36 88 18

16 29 16 73 86 72 86

99 65 31 86 31

63 77 8 26 69 34 14

57 35 15 45 13 79 5  
добавим строку “5 47 34” и выведем содержимое файла (Рисунок 1.3).

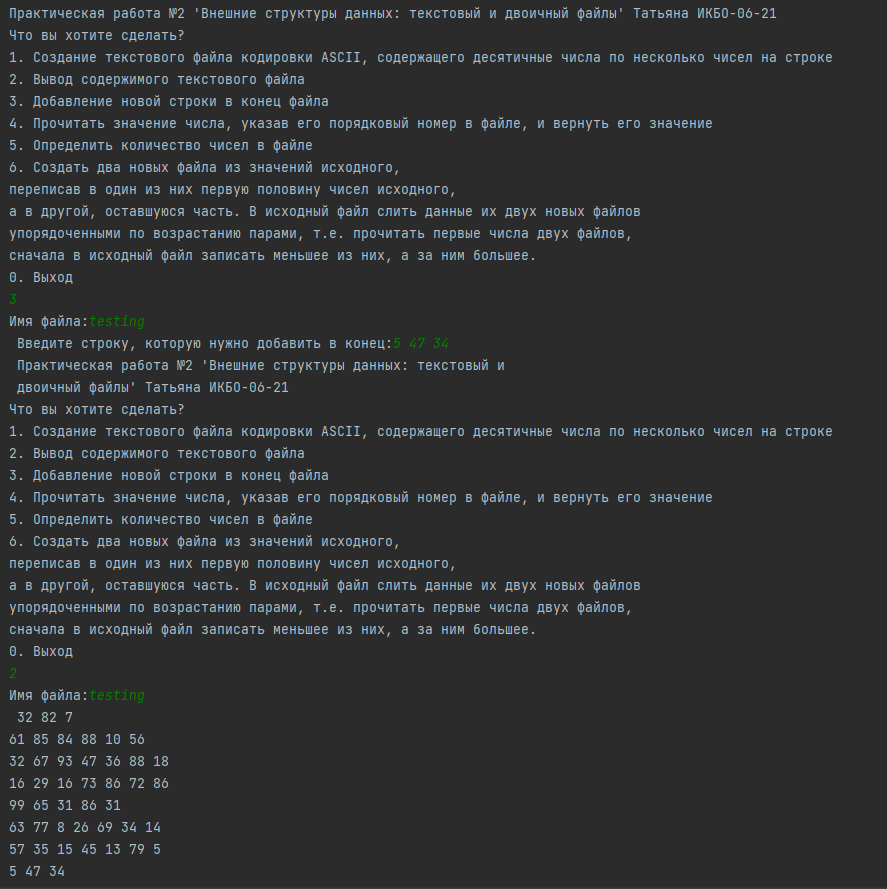


Рисунок 1.3. Тестирование функции добавления строки в конец

Протестируем чтения числа под указанным номером и определения количества чисел в файле (Рисунок 1.4).

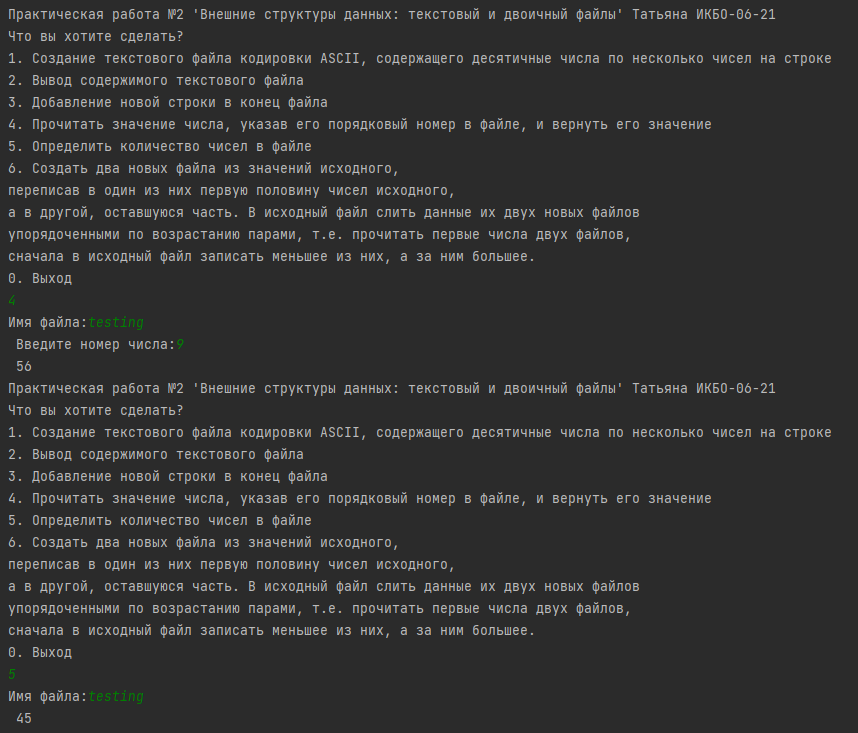


Рисунок 1.4. Тестирование чтения числа по индексу и определения количества чисел в файле

Протестируем создание новых файлов из значений исходного, поделённых пополам и слитых в исходный упорядоченными по возрастанию. Результатом выполнения функции будет создание файла и вывод его названия (Рисунок 1.5).

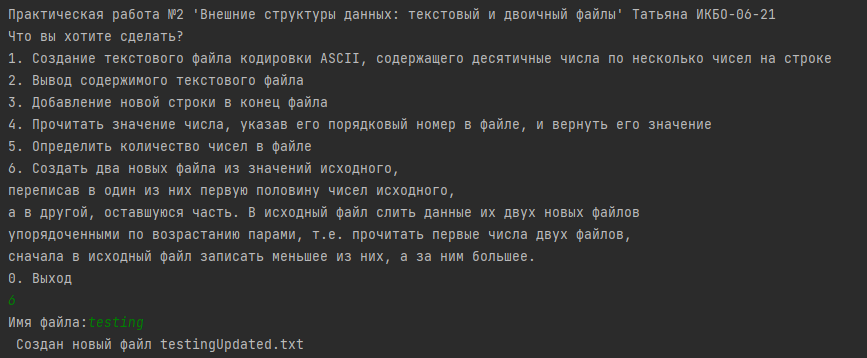


Рисунок1.5. Проверка дополнительной функции

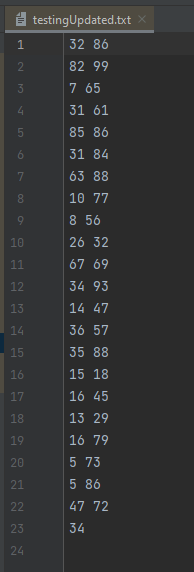


Рисунок1.6. Проверка создания необходимых файлов

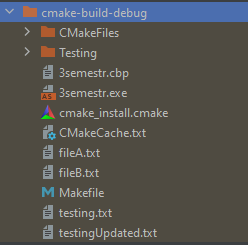


Рисунок1.7. Проверка создания необходимых файлов

# Вывод

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

1. **Исходный код программы**

|  |
| --- |
| //  // Created by t.shkolnik  //  #include <iostream>  #include <windows.h>  #include <string>  #include <ctime>  #include <fstream>  #include "functions.h"  using namespace std;  //Вводная функция, выбор задания  void onboarding() {  cout << "Практическая работа №2 'Внешние структуры данных: текстовый и двоичный файлы' Татьяна ИКБО-06-21" << endl;  cout << "Что вы хотите сделать?" << endl;  cout << "1. Создание текстового файла кодировки ASCII, содержащего десятичные числа по несколько чисел на строке"  << endl;  cout << "2. Вывод содержимого текстового файла" << endl;  cout << "3. Добавление новой строки в конец файла" << endl;  cout << "4. Прочитать значение числа, указав его порядковый номер в файле, и вернуть его значение" << endl;  cout << "5. Определить количество чисел в файле" << endl;  cout << "6. Создать два новых файла из значений исходного, "  "\nпереписав в один из них первую половину чисел исходного, "  "\nа в другой, оставшуюся часть. В исходный файл слить данные их двух новых файлов "  "\nупорядоченными по возрастанию парами, т.е. прочитать первые числа двух файлов, "  "\nсначала в исходный файл записать меньшее из них, а за ним большее." << endl;  cout << "0. Выход" << endl;  }  void task1() {  string fileName;  cout << "Имя файла: ";  cin.get();  getline(cin, fileName);  createFileWithRandomNumbers(fileName);  }  void task2() {  string fileName;  cout << "Имя файла: ";  cin.get();  getline(cin, fileName);  print(fileName);  }  void task3() {  string fileName;  string stringThatAdd;  cout << "Имя файла: ";  cin.get();  getline(cin, fileName);  cout << "Введите строку, которую нужно добавить в конец: ";  getline(cin, stringThatAdd);  addLine(stringThatAdd, fileName);  }  void task4() {  int positionNum;  string fileName;  cout << "Имя файла: ";  cin.get();  getline(cin, fileName);  cout << "Введите номер числа: ";  cin >> positionNum;  positionNum = readNumber(fileName, positionNum);  cout << positionNum << endl;  }  void task5(){  string fileName;  cout << "Имя файла: ";  cin.get();  getline(cin, fileName);  cout << numbersCount(fileName) << endl;  }  void task6() {  string fileName;  cout << "Имя файла: ";  cin.get();  getline(cin, fileName);  additionalOperation(fileName);  cout << "Создан новый файл " + fileName + "Updated.txt" << endl;  }  int main() {  SetConsoleOutputCP(CP\_UTF8);  srand(time(NULL));  int menu = 1;  while (menu != 0) {  onboarding();  cin >> menu;  switch (menu) {  case 1: task1(); break;  case 2: task2(); break;  case 3: task3(); break;  case 4: task4(); break;  case 5: task5(); break;  case 6: task6(); break;  }  }  system("pause");  return 0;  } |

**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 № 8. Дополнительное задание:

|  |  |  |
| --- | --- | --- |
| № | Задание | |
| 1 | Структура записи | Учет нарушений ПДД. Структура записи о нарушении ПДД: номер автомобиля, фамилия и инициалы владельца, модель, дата нарушения, место нарушения (текстом), статья (КоАП), наказание (сумма штрафа). |
| Доп. операция | 1. Сформировать список нарушений по автомобилю заданного номера. Результат сохранить в новом двоичном файле с той же структурой записи, что и исходный файл. 2. Увеличить сумму штрафа вдвое по всем авто за указанную дату и по заданной статье. |

# Решение

В нашем, человеческом представлении, файлы в компьютере могут хранится в двух вариантах: понятно компьютеру и понятно нам. Понятные нам файлы — это текстовые файлы, понятные компьютеру — это бинарные файлы.

Двоичные файлы представляют собой набор байтов, а не набор символов, которые содержатся в текстовых файлах. Бинарные файлы хранят информацию в том виде, в котором она представлена в памяти компьютера, поэтому, открыв такой файл с помощью текстового редактора, мы получим набор нечитаемых символов.

Из файла записи удобнее читать в том виде, в котором их записывали, потому что у записей фиксированная длина.

Структура, используемая в программе:

|  |
| --- |
| struct TrafficViolation {  int autoNumber;  char ownerFIO[50];  char model[20];  int date;  char place[50];  int article;  int punishment;  }; |

Модуль функций:

1. Нужно преобразовать тестовые данных из текстового файла в двоичный файл. При помощи создания объекта структуры, считывая значения в её поля, запишем структуру как запись в двоичный файл.

|  |
| --- |
| void convertTextToBin(string &fileName) {  ifstream textFile;  textFile.open(fileName + ".txt");  ofstream binFile;  binFile.open(fileName + "Bin.dat", ios::binary);  if (textFile.is\_open() && binFile.is\_open()) {  while (textFile.good() && binFile.good()) {  TrafficViolation trafficViolation{};  string line;  textFile >> trafficViolation.autoNumber;  textFile.get();  getline(textFile, line);  strcpy(trafficViolation.ownerFIO, line.c\_str());  getline(textFile, line);  strcpy(trafficViolation.model, line.c\_str());  textFile >> trafficViolation.date;  textFile.get();  getline(textFile, line);  strcpy(trafficViolation.place, line.c\_str());  textFile >> trafficViolation.article;  textFile.get();  textFile >> trafficViolation.punishment;  textFile.get();  binFile.write((char \*) (&trafficViolation), sizeof(trafficViolation));  binFile.clear();  }  textFile.close();  binFile.close();  }  } |

1. Нужно преобразовать данных из двоичного файла в текстовый. Воспользуемся циклом и будем считывать записи фиксированной длины, создавая объекты структуры. Далее выведем данные из полей объекта нарушения в текстовый файл.

|  |
| --- |
| void convertBinToText(string &fileName) {  ifstream binFile;  binFile.open(fileName + ".dat", ios::binary);  ofstream textFile;  textFile.open(fileName + "Text.txt");  if (binFile.is\_open() && textFile.is\_open()) {  TrafficViolation trafficViolation {};  bool isFirstString = true;  while (binFile.read((char \*) &trafficViolation, sizeof(TrafficViolation))) {  if (!isFirstString) {  textFile << '\n';  } else { isFirstString = false; }  textFile << trafficViolation.autoNumber;  textFile << '\n' << trafficViolation.ownerFIO;  textFile << '\n' << trafficViolation.model;  textFile << '\n' << trafficViolation.date;  textFile << '\n' << trafficViolation.place;  textFile << '\n' << trafficViolation.article;  textFile << '\n' << trafficViolation.punishment;  }  binFile.clear();  textFile.clear();  if (binFile.good() && textFile.good()) {  binFile.close();  textFile.close();  }  }  } |

1. Нужно вывести всех записи двоичного файла. Воспользуемся циклом и будем считывать записи фиксированной длины, создавая объекты структуры. Далее выведем данные из полей объекта нарушения в консоль.

|  |
| --- |
| void print(string &fileName) {  ifstream binFile;  binFile.open(fileName + ".dat", ios::binary);  if (binFile.is\_open()) {  while (binFile.good()) {  TrafficViolation trafficViolation;  binFile.read((char \*) &trafficViolation, sizeof(TrafficViolation));  cout << "------------" << endl;  cout << trafficViolation.autoNumber << endl;  cout << trafficViolation.ownerFIO << endl;  cout << trafficViolation.model << endl;  cout << trafficViolation.date << endl;  cout << trafficViolation.place << endl;  cout << trafficViolation.article << endl;  cout << trafficViolation.punishment << endl;  }  binFile.close();  }  } |

1. Нужно получить доступ к записи по ее порядковому номеру в файле. Используя механизм прямого доступа к записи в двоичном файле, считаем запись из нужного места в файле и выведем данные из её полей в консоль.

|  |
| --- |
| TrafficViolation getRecord(string &fileName, int &index) {  TrafficViolation trafficViolation;  ifstream binFile;  binFile.open(fileName + ".dat", ios::binary);  binFile.seekg(index \* sizeof(TrafficViolation), ios::beg);  binFile.read((char \*) (&trafficViolation), sizeof(TrafficViolation));  return trafficViolation;  } |

1. Нужно удалить записи с заданным значением ключа, выполнить путем замены на последнюю запись. За значение ключа возьмём номер автомобиля так как это уникальное значение. Используя цикл найдём нарушение с указанным номером авто и используя механизма прямого доступа поменяем его на последнее нарушение в файле.

|  |
| --- |
| void deleteRecordAndReplaceByLast(string &fileName, int &autoNumber) {  fstream binFile;  binFile.open(fileName + ".dat", ios::binary);  if (binFile.is\_open()) {  int counter = 0;  while (binFile.good()) {  TrafficViolation trafficViolation;  binFile.read((char \*) (&trafficViolation), sizeof(TrafficViolation));  if (trafficViolation.autoNumber == autoNumber) {  binFile.seekp(-1 \* sizeof(TrafficViolation), ios::end);  binFile.read((char \*) (&trafficViolation), sizeof(TrafficViolation));  binFile.seekp(counter \* sizeof(TrafficViolation), ios::beg);  binFile.write((char \*) (&trafficViolation), sizeof(TrafficViolation));  break;  }  counter++;  }  binFile.close();  }  } |

1. Нужно сформировать список нарушений по автомобилю заданного номера. Результат сохранить в новом двоичном файле с той же структурой записи, что и исходный файл. Используем цикл для чтения из двоичного файла и проверяем, что номер автомобиля совпадает с указанным пользователем и только при совпадении записываем нарушение в результирующий файл.

|  |
| --- |
| void createFileWithViolationsByAutoNumber(  string &oldFileName,  string &newFileName,  int &autoNumber  ) {  ifstream binFile;  binFile.open(oldFileName + ".dat", ios::binary);  ofstream resultBinaryFile;  resultBinaryFile.open(newFileName + ".dat", ios::binary);  if (binFile.is\_open() && resultBinaryFile.is\_open()) {  TrafficViolation trafficViolation;  while (binFile.read((char \*) (&trafficViolation), sizeof(TrafficViolation))) {  if (trafficViolation.autoNumber == autoNumber) {  resultBinaryFile.write((char \*) (&trafficViolation), sizeof(TrafficViolation));  }  }  binFile.clear();  resultBinaryFile.clear();  if (binFile.good()){  binFile.close();  }  if (resultBinaryFile.good()) {  resultBinaryFile.close();  }  }  } |

1. Нужно увеличить сумму штрафа вдвое по всем авто за указанную дату и по заданной статье. Воспользуемся циклом и будем считывать записи фиксированной длины, создавая объекты структуры нарушения. Далее сравниваем поля объекта структуры (дата и статья) с введёнными пользователем. При одновременном совпадении данных у этой записи увеличиваем значение штрафа в 2 раза

|  |
| --- |
| void doubleIncreasePunishmentByDateAndArticle(  string &fileName,  int &date,  int &article  ) {  ifstream binFile;  binFile.open(fileName + ".dat", ios::binary);  ofstream resultBinFile;  resultBinFile.open("DoublePunishment" + fileName + ".dat", ios::binary);  if (binFile.is\_open() && resultBinFile.is\_open()) {  TrafficViolation trafficViolation;  while (binFile.read((char \*) (&trafficViolation), sizeof(TrafficViolation))) {  if (trafficViolation.date == date && trafficViolation.article == article) {  trafficViolation.punishment \*= 2;  }  resultBinFile.write((char \*) (&trafficViolation), sizeof(TrafficViolation));  }  binFile.clear();  resultBinFile.clear();  if (binFile.good() && resultBinFile.good()) {  binFile.close();  resultBinFile.close();  }  }  } |

# 2.4 Тестирование

На старте программы появляется главное меню, где нужно выбрать номер интересующего задания (Рисунок 2.1).

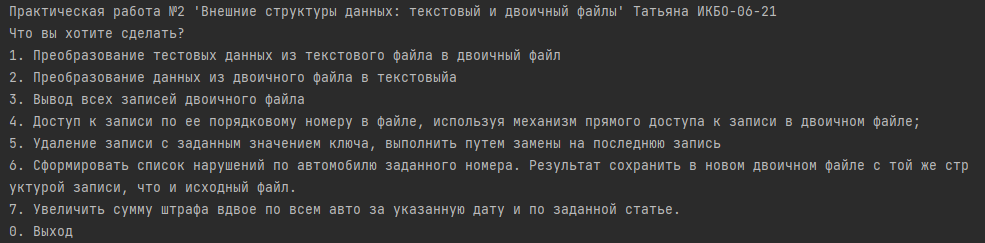


Рисунок 2.1. Главное меню

Подготовим текстовый файл для тестирования (Рисунок 2.2).

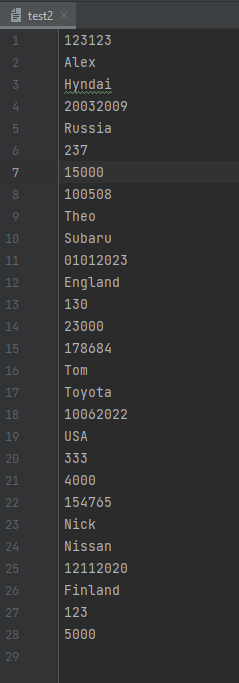


Рисунок 2.2. Тестовые текстовые данные файла

Протестируем преобразование данных из текстового в двоичный файл (Рисунок 2.3) и вывод всех записей двоичного файла (Рисунок 2.4).

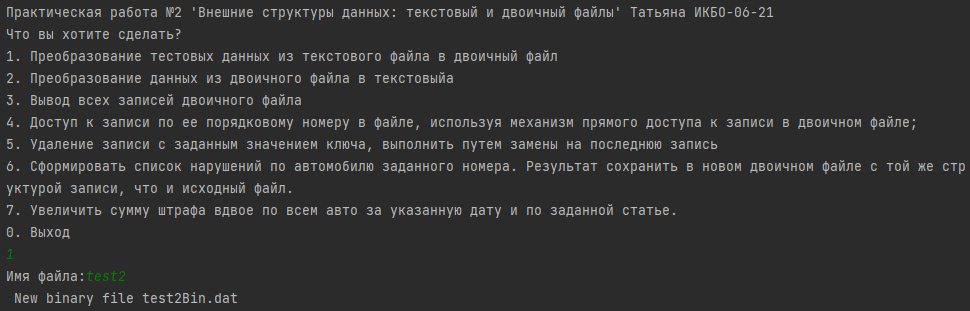


Рисунок 2.3. Тестирование преобразования в двоичный файл

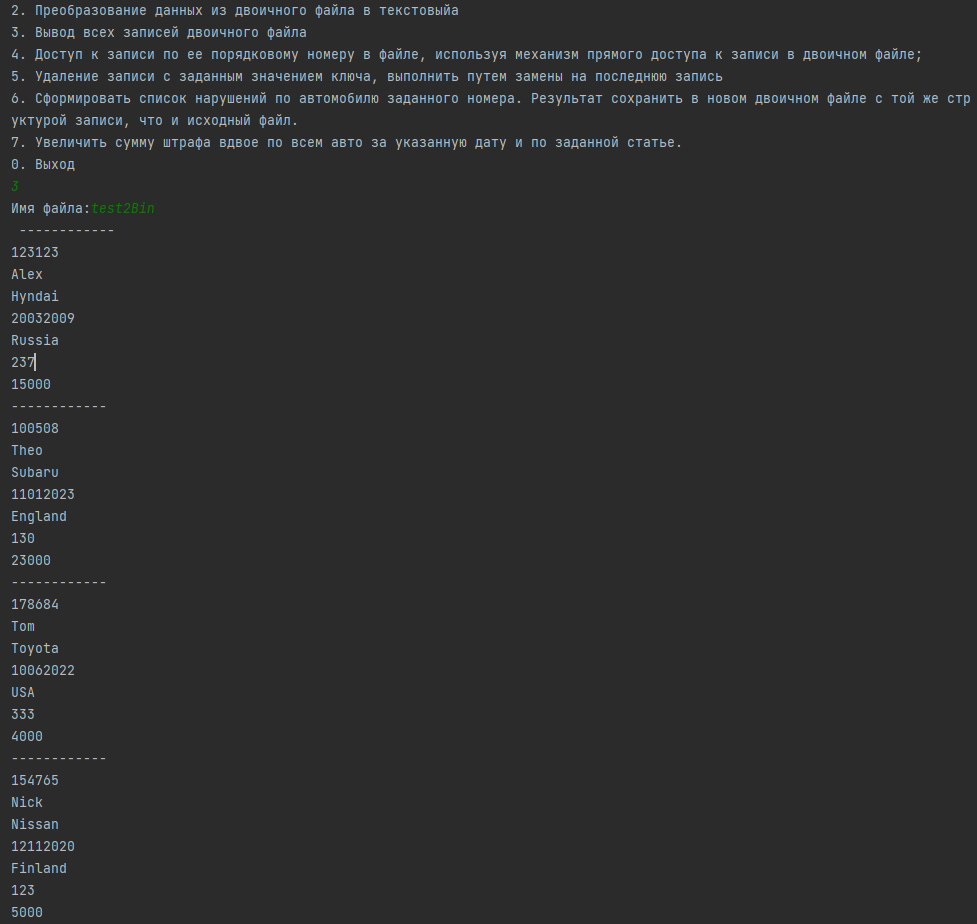


Рисунок 2.4. Тестирование вывода записей

Протестируем преобразование данных из двоичного файла в текстовый файл (Рисунок 2.5). Результат в виде текстового файла (Рисунок 2.6).

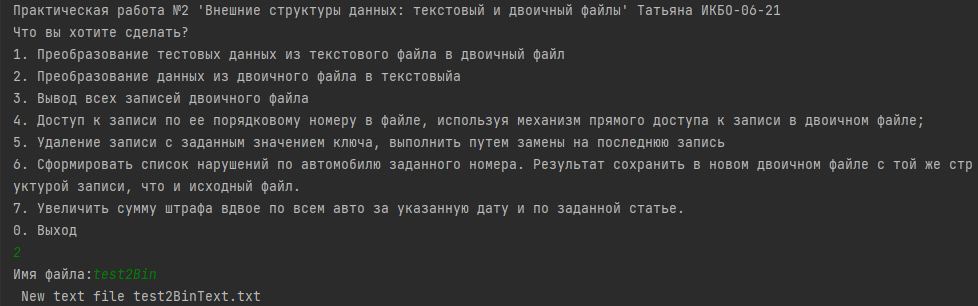


Рисунок 2.5. Тестирование преобразования данных из двоичного файла в текстовый

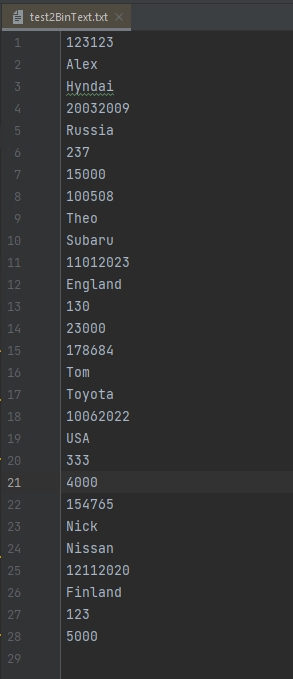


Рисунок 2.6. Результат преобразования данных из двоичного в текстовый формат

Протестируем доступ к записи по ее порядковому номеру в файле (Рисунок 2.7).

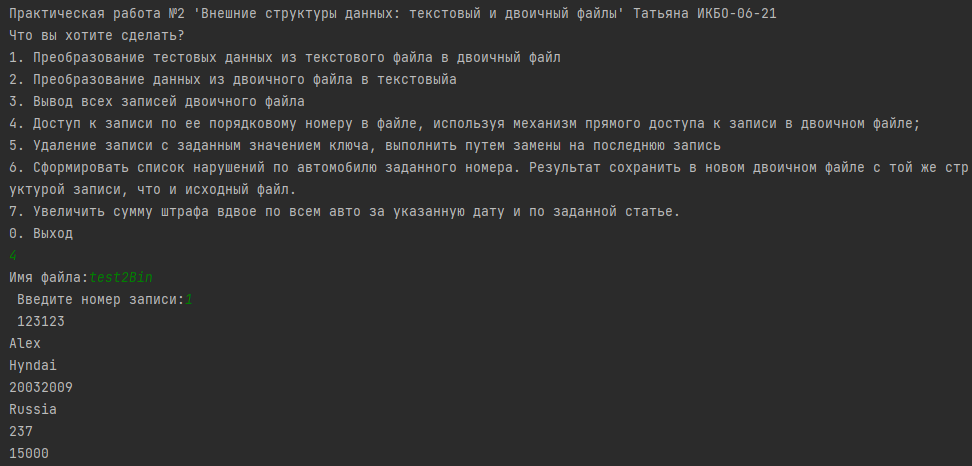


Рисунок 2.7. Результат получения доступа к записи по ее номеру

Протестируем удаление записи по заданному значению ключа (Рисунок 2.8). Результатом будет файл, полученный преобразованием двоичного файла в текстовый (Рисунок 2.9).

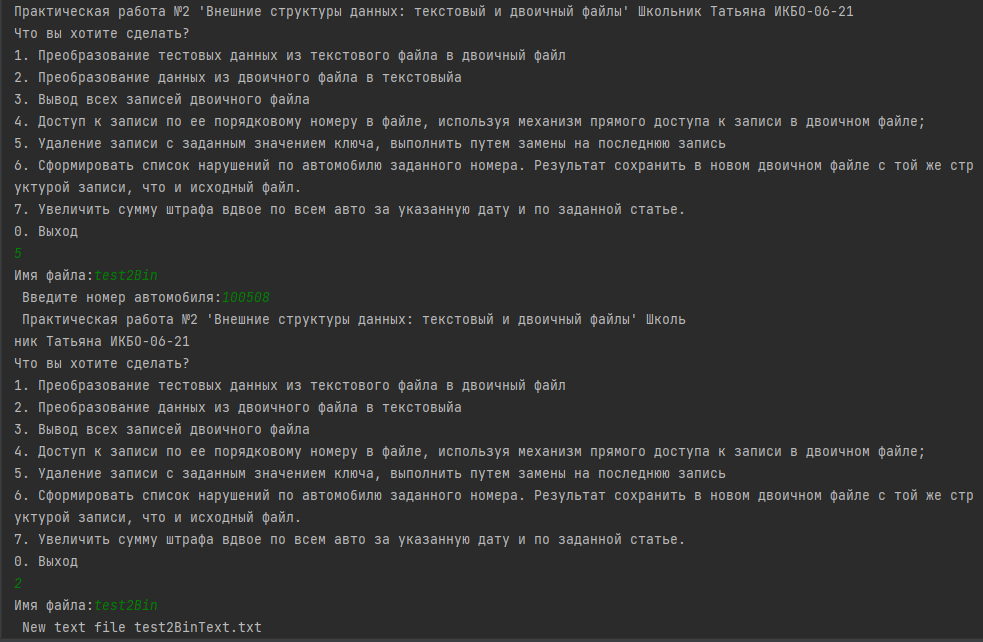


Рисунок 2.8. Тестирование удаления записи по ключу

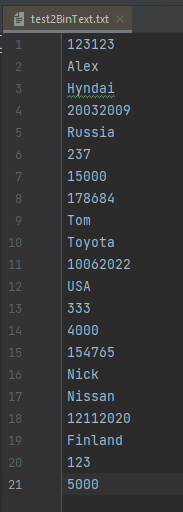


Рисунок 2.9. Результат удаления записи по ключу

Протестируем формирование списка нарушений по автомобилю заданного номера. (Рисунок 2.10).

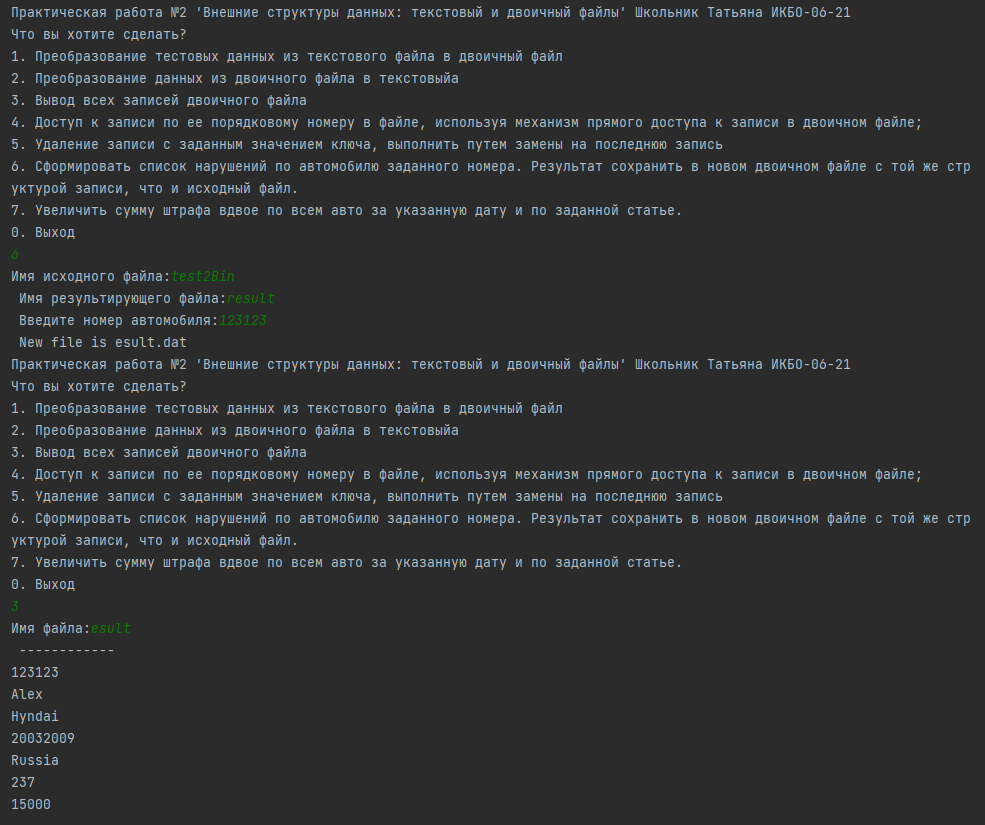


Рисунок 2.10. Тестирование формирования списка нарушений по автомобилю заданного номера

Протестируем увеличение суммы штрафа вдвое по всем авто за указанную дату и по заданной статье. Записи из преобразованного файла выведем в консоль (Рисунок 2.11).



Рисунок 2.11. Тестирование увеличения суммы штрафа вдвое по всем авто за указанную дату и по заданной статье

# Вывод

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

# Исходный код программы

|  |
| --- |
| //  // Created by t.shkolnik  //  #include <iostream>  #include <windows.h>  #include <string>  #include <ctime>  #include <fstream>  #include "functions2.h"  using namespace std;  //Вводная функция, выбор задания  void onboarding() {  cout << "Практическая работа №2 'Внешние структуры данных: текстовый и двоичный файлы' Школьник Татьяна ИКБО-06-21" << endl;  cout << "Что вы хотите сделать?" << endl;  cout << "1. Преобразование тестовых данных из текстового файла в двоичный файл" << endl;  cout << "2. Преобразование данных из двоичного файла в текстовыйа" << endl;  cout << "3. Вывод всех записей двоичного файла" << endl;  cout << "4. Доступ к записи по ее порядковому номеру в файле, используя механизм прямого доступа к записи в двоичном файле;" << endl;  cout << "5. Удаление записи с заданным значением ключа, выполнить путем замены на последнюю запись" << endl;  cout << "6. Сформировать список нарушений по автомобилю заданного номера. "  "Результат сохранить в новом двоичном файле с той же структурой записи, что и исходный файл." << endl;  cout << "7. Увеличить сумму штрафа вдвое по всем авто за указанную дату и по заданной статье." << endl;  cout << "0. Выход" << endl;  }  void task1() {  string fileName;  cout << "Имя файла: ";  cin.get();  getline(cin, fileName);  convertTextToBin(fileName);  cout << "New binary file " + fileName + "Bin.dat" << endl;  }  void task2() {  string fileName;  cout << "Имя файла: ";  cin.get();  getline(cin, fileName);  convertBinToText(fileName);  cout << "New text file " + fileName + "Text.txt" << endl;  }  void task3() {  string fileName;  string stringThatAdd;  cout << "Имя файла: ";  cin.get();  getline(cin, fileName);  print(fileName);  }  void task4() {  int positionNum;  string fileName;  cout << "Имя файла: ";  cin.get();  getline(cin, fileName);  cout << "Введите номер записи: ";  cin >> positionNum;  if(positionNum < 1) {  cout << "Неправильный порядковый номер" << endl;  } else {  positionNum -= 1; // т.к. нужен индекс  TrafficViolation trafficViolation = getRecord(fileName, positionNum);  cout << trafficViolation.autoNumber << endl;  cout << trafficViolation.ownerFIO << endl;  cout << trafficViolation.model << endl;  cout << trafficViolation.date << endl;  cout << trafficViolation.place << endl;  cout << trafficViolation.article << endl;  cout << trafficViolation.punishment << endl;  }  }  void task5(){  string fileName;  int autoNumber;  cout << "Имя файла: ";  cin.get();  getline(cin, fileName);  cout << "Введите номер автомобиля: ";  cin >> autoNumber;  deleteRecordAndReplaceByLast(fileName, autoNumber);  }  void task6() {  string fileName;  string newFileName;  int autoNumber;  cout << "Имя исходного файла: ";  cin.get();  getline(cin, fileName);  cout << "Имя результирующего файла: ";  cin.get();  getline(cin, newFileName);  cout << "Введите номер автомобиля: ";  cin >> autoNumber;  createFileWithViolationsByAutoNumber(fileName, newFileName, autoNumber);  cout << "New file is " + newFileName + ".dat" << endl;  }  void task7() {  string fileName;  int date;  int article;  cout << "Имя исходного файла: ";  cin.get();  getline(cin, fileName);  cout << "Дата нарушения: ";  cin >> date;  cout << "Статья(КоАП): ";  cin >> article;  doubleIncreasePunishmentByDateAndArticle(fileName, date, article);  cout << "Updated file is DoublePunishment" + fileName + ".dat" << endl;  }  int main() {  SetConsoleOutputCP(CP\_UTF8);  srand(time(NULL));  int menu = 1;  while (menu != 0) {  onboarding();  cin >> menu;  switch (menu) {  case 1: task1(); break;  case 2: task2(); break;  case 3: task3(); break;  case 4: task4(); break;  case 5: task5(); break;  case 6: task6(); break;  case 7: task7(); break;  }  }  system("pause");  return 0;  } |