|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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 ИНБО-03-22 | Алтухов А.В. |
| Принял преподаватель | Филатов А.С. |

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

Москва 2023

# **Цель работы**

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

# **Постановка задачи 1**

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; файл может быть образцовым или введен пользователем;
      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.
      6. реализовать дополнительные операции, определенные вариантов
   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. Выполнить тестирование приложения в полном объеме.

Вариант №2. Условие задания:

Скопировать числа исходного файла, которые кратны 7, в новый файл.

# **Решение задачи 1**

Файловый поток (fstream) в C++ — это механизм для работы с файлами. Он позволяет открывать файлы для чтения и записи данных. Файловые потоки являются частью стандартной библиотеки C++, и они упрощают взаимодействие с внешними файлами. Для открытия файла нужно создать файловый поток fstream и вызвать для него метод open, в который передается название файла (если он лежит в одной папке с нашим cpp файлом) или путь к нему и указывается флаг, которые определяет режим открытия файла. Флаги бывают:

* **ios::in**: файл открывается для ввода (чтения). Может быть установлен только для объекта ifstream или fstream
* **ios::out**: файл открывается для вывода (записи). При этом старые данные удаляются. Может быть установлен только для объекта ofstream или fstream
* **ios::app**: файл открывается для дозаписи. Старые данные не удаляются.
* **ios::ate**: после открытия файла перемещает указатель в конец файла
* **ios::trunc**: файл усекается при открытии. Может быть установлен, если также установлен режим out
* **ios::binary**: файл открывается в бинарном режиме

Можно комбинировать несколько флагов, для этого необходимо между ними поставить “ | ”. Также, можно не прописывать флаги, если нам нужно лишь вести запись в файл или считывать с него данные. Для чтения файла можно вместо fstream использовать ifstream, а для записи ofstream.

Изначально файл создан, но во время выполнения программы он может быть удален, поэтому была написана функция createSampleFile, которая на входе получает название файла, который будет создан. Файл заполняется строками с числами. Функция возвращает целое число, которое указывает на результат выполнения операции.

|  |
| --- |
| int createSampleFile(string& filename) {  cout << "Ввести числа\n1-автоматически\n2-самому\n";  int n;  cin >> n;  ofstream file(filename, fstream::app);  if (!file.is\_open()) {  return -1;  }    if (n == 1) {  //Автоматическое заполнения файла  file << "10 21 38 40 56" << endl  << "63 70 80 91 100" << endl  << "112 120 133 140 150"<< endl;      cout << "Файл создан" << endl;    }  else {  //Ручное заполнение файла    string line;    cout << "Введите десятичные числа (разделенные пробелами, введите 'exit' для завершения):" << endl;  getline(cin, line);  if (file.good()) {  while (true) {  getline(cin, line);    if (line == "exit") { break; }  file << line<< endl;    }  }    cout << "Файл создан" << endl;  }  if (file.good()) {  file.close();  return 0;  }  else {  file.close();  return -2;  }  } |

Для вывода содержимого файла на экран была написана функция readFromFile, которая принимает в качестве аргумента название файла. Функция возвращает целое число, которое указывает на результат выполнения операции.

|  |
| --- |
| int readFromFile(string& filename) {  ifstream file(filename);  if (!file.is\_open()) {  return -1;  }  string line;    while (getline(file, line)) {  cout << line << endl;  }    if (file.good()) {  file.close();  return 0;  }  else {  file.close();  return -2;  }    } |

Функция writeToFile получает на вход название файла. Функция возвращает целое число, которое указывает на результат выполнения операции.

|  |
| --- |
| int writeToFile(string& filename) {    ofstream file(filename, fstream::app);    if (!file.is\_open()) {  return -1;  }  string data;  getline(cin, data);  getline(cin, data);  file << data<< endl;    if (file.good()) {  file.close();  return 0;  }  else {  file.close();  return -2;  }  } |

Для того, чтобы найти число в файле по его индексу была написана функция read\_number\_by\_index, которая на вход получает название файла и целочисленную переменную, которая отвечает за индекс искомого числа. Если число находится, то функция выводит его на экран, а иначе возвращает целое число, которое указывает на результат выполнения операции.

|  |
| --- |
| Int read\_number\_by\_index(string file\_name, int index) {  ifstream file(file\_name);  if (!file.is\_open()) {  return -1;  }  string line;  int count = 0;  int cnt = 0;  if (file.good()) {  while (getline(file, line)) {  istringstream iss(line);  int num;  while (iss >> num) {  count++;  if (count == index) {  cout << "Искомое число: " << num << endl;  cnt += 1;  break;  }  }  }  }  if(cnt==0){ cout << "Искомое число не найдено " << endl; }  if (file.good()) {  file.close();  return 0;  }  else {  file.close();  return -2;  }  } |

Для того чтобы посчитать количество чисел в файле была написана функция count\_numbers, которая на вход принимает только название файла. Если количество чисел посчиталось, то функция выводит это число на экран, иначе возвращает целое число, которое указывает на результат выполнения операции.

|  |
| --- |
| int count\_numbers(string file\_name) {  ifstream file(file\_name);  if (!file.is\_open()) {  return -1;  }  string line;  int count = 0;  while (getline(file, line)) {  istringstream iss(line);  int num;  while (iss >> num) {  count++;  }  }  cout << "Количество чисел: " << count<<endl;  if (file.good()) {  file.close();  return 0;  }  else {  file.close();  return -2;  }  } |

Чтобы выполнить задание моего варианта была написана функция create\_new\_file, которая получает в качестве аргумента название файла. Функция создает файл по варианту и возвращает целое число, которое указывает на результат выполнения операции.

|  |
| --- |
| int create\_new\_file(string file\_name) {  ifstream file(file\_name);  if (!file.is\_open()) { return -1; }  //Создается вектор `numbers`, в котором будут храниться числа из исходного файла.  vector<int> numbers;  string line;  while (getline(file, line)) {  //Создается объект `istringstream` для каждой строки, который позволяет преобразовать строку в поток и считывать числа из нее с помощью оператора `>>`  istringstream iss(line);  int num;  while (iss >> num) {  numbers.push\_back(num);  }  }  vector<string> new\_numbers;  for (int num : numbers) {  if (num % 7 == 0) { new\_numbers.push\_back(to\_string(num)); }  }  ofstream new\_file("new\_" + file\_name);  for (string num\_str : new\_numbers) {  new\_file << num\_str << " ";  }  cout << endl;  if (file.good())  {  file.close();  return 0;  }  else {  file.close();  return -2;  }  } |

При запуске программы пользователя просят ввести название файла, который будет сразу создан, после чего он видит список операций.

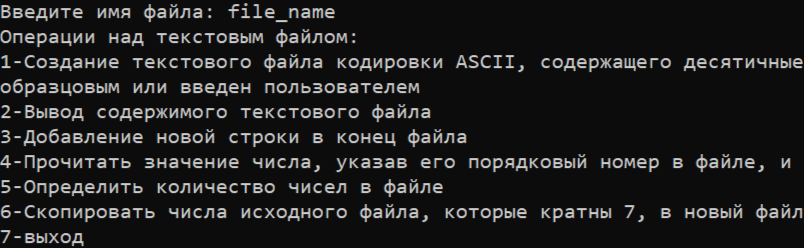


Рисунок 1. Интерфейс программы

# **Тестирование программы 1**

Протестируем работы программы. Начнем с первой операции. После вывода списка операций откроем папку, где лежит созданный файл и удалим его. Попробуем вызвать вторую операцию и убедимся, что файл удален (рис. 2). Вызовем первую операцию (рис. 3). Убедимся, что файл действительно создан, для этого посмотрим в папку (рис. 4).



Рисунок 2. Вызов операции 2 при удаленном файле



Рисунок 3. Вызов операции повторного создания файла

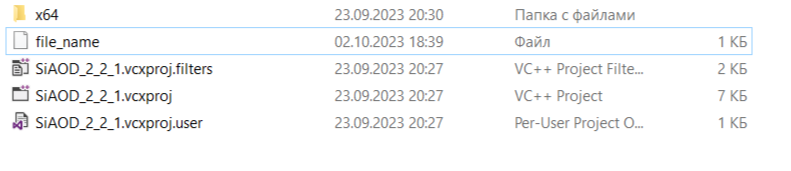


Рисунок 4. Содержимое папки с созданным файлом

Теперь попробуем вывести содержимое файла (рис. 5). Все успешно выводит.

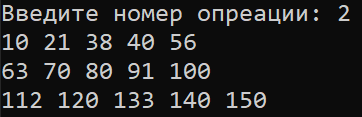


Рисунок 5. Вывод содержимого файла

Попробуем добавить новую строку в файл (рис. 6). Как можно увидеть, все исправно работает.

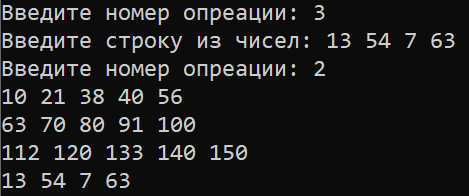


Рисунок 6. Добавление новой строки в конец файла

Теперь попробуем найти два числа в файле по индексу. Введем индекс 19 и увидим, что программа выводит число 63, которое мы недавно добавили в файл. И введем индекс 50, которого в файле нету, и убедимся, что программа отрабатывает подобный случай (рис. 7).

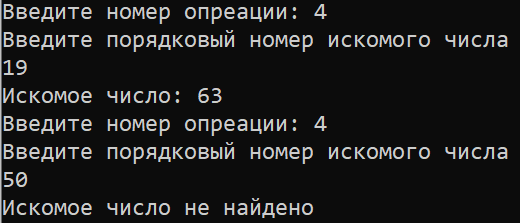


Рисунок 7. Поиск числа по заданному индексу

Теперь попробуем посчитать количество чисел в файле. Для наглядности сначала выведем содержимое файла на экран, после чего вызовем соответствующую функцию (рис. 8). В файле действительно 19 чисел.

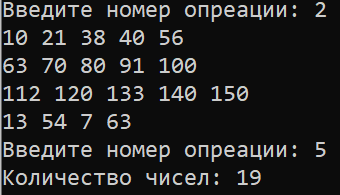


Рисунок 8. Подсчет количества чисел в файле

И наконец, вызовем последнюю операцию (рис. 9). Она не выводит содержимое файла, поэтому посмотрим его сами (рис. 10). Действительно, содержимое файла как прописано в варианте.

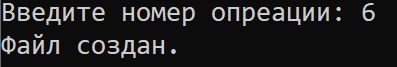


Рисунок 9. Создание нового файла

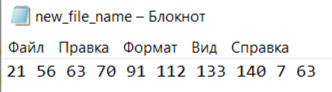


Рисунок 10. Содержимое нового файла

# **Постановка задачи 2**

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. Разработать приложение, демонстрирующее выполнение всех операций.
   8. Выполнить тестирование приложения, продемонстрировав выполнение всех операций.
   9. Составить отчет, отобразив в нем описание выполнения всех этапов разработки, тестирования и код всей программы со скриншотами результатов тестирования.

# **Решение задачи 2**

На случай удаления исходного файла была написана функция create\_begin\_file, которая создает этот исходный файл заново. Она ничего не принимает и ничего не возвращает.

|  |
| --- |
| void create\_begin\_file() {  fstream fs;  fs.open("begin.txt", ios::out);  fs << "1111111 Алтухов Ульяновск\n2222222 Патрикеев Москва\n3333333 Ефимов Балашиха\n4444444 Глазов Тверь\n5555555 Танушкин Мордовия\n6666666 Матюхин Уфа";  fs.close();  } |

Была написана вспомогательная функция set\_word, которая получается на вход ссылку на структуру bankAccount и слова, которые будут записаны в поле fullName и addres соответственно переданной структуры. Функция ничего не возвращает

|  |
| --- |
| void set\_word(bankAccount& rc, char fullName[10], char address[10]) {  for (int i = 0; i < 10; i++) {  rc.fullName[i] = fullName[i];  rc.address[i] = address[i];  }  } |

Также была написана вспомогательная функция clear, которая получает на вход ссылки на слова, которые представляют из себя массив char’ов. Функция каждый символ обнуляет и ничего не возвращает.

|  |
| --- |
| void clear(char (&fullName)[10], char(&address)[10]) {  for (int i = 0; i < 10; i++) {  fullName[i] = '\0';  address[i] = '\0';  }  } |

Для того, чтобы создать двоичный файл, была написана функция make\_dv, которая получает в качестве аргументов ссылку на файловый поток, из которого будет считываться содержимое, и имя, как будет называться двоичный файл. Функция ничего не возвращает.

|  |
| --- |
| void make\_dv(fstream& fs\_o, string filename) {  fstream fs\_n;  fs\_n.open(filename, ios::out | ios::binary);  vector<bankAccount>records;  int accountNumber;  char fullName[10];  char address[10];  while (!fs\_o.eof()) {  clear(fullName, address);  fs\_o >> accountNumber >> fullName >> address ;  bankAccount rc;  set\_word(rc, fullName, address);  rc.accountNumber = accountNumber;  records.push\_back(rc);  }  for (int i = 0; i < records.size(); i++)  fs\_n.write(reinterpret\_cast<char\*>(&records[i]), sizeof(bankAccount));  fs\_n.close();  } |

Для того, чтобы создать текстовый файл на основе двоичного, была написана функция make\_text, которая получает на вход ссылку на файловый поток двоичного файла и ссылку на название, как новый файл должен называться. Функция ничего не возвращает.

|  |
| --- |
| void make\_text(fstream& fs\_d, string& filename) {  fstream fs\_t;  fs\_t.open(filename, ios::out);  vector<bankAccount>records;  bankAccount rc;  while (true) {  fs\_d.read(reinterpret\_cast<char\*>(&rc), sizeof(bankAccount));  if (fs\_d.eof()) break;  records.push\_back(rc);  }  for (int i = 0; i < records.size(); i++) {  fs\_t << records[i].accountNumber << " " << records[i].fullName << " " << records[i].address << "\n";  }  fs\_t.close();} |

Для того, чтобы вывести на экран содержимое двоичного файла, была написана функция print\_binary, которая получает на вход только ссылку на файловый поток. Функция ничего не возвращает.

|  |
| --- |
| void print\_binary(fstream& fs) {  vector<bankAccount>records;  bankAccount rc;  while (true) {  fs.read(reinterpret\_cast<char\*>(&rc), sizeof(bankAccount));  if (fs.eof()) break;  records.push\_back(rc);  }  for (int i = 0; i < records.size(); i++)  cout << records[i].accountNumber << " " << records[i].fullName << " " << records[i].address << endl;  } |

Для того, чтобы достать из двоичного файла запись о структуре по ее индексу, была написана функция get\_record, которая получает на вход ссылку на файловый поток двоичного файла и номер позиции, которую нужно найти. Функция возвращает структуру bankAccount, которая либо соответствует найденной, либо пустая, если не удалось найти искомую.

|  |
| --- |
| bankAccount get\_bankAccount(fstream& fs, int num) {  vector<bankAccount>records;  bankAccount rc;  while (true) {  fs.read(reinterpret\_cast<char\*>(&rc), sizeof(bankAccount));  if (fs.eof()) break;  records.push\_back(rc);  }  bankAccount res = {0, "", ""};  if (num >= records.size()) return res;  res = records[num];  return res;  } |

Была написана вспомогательная функция equal, которая сравнивает два числа. Функция получает на вход два числа и возвращает true либо false.

|  |
| --- |
| bool equal(int w1, int w2) {  if (w1 != w2)  return false;  else if (w1 == '\0');  return true;  } |

Для удаления из двоичного файла записи с заданным словом была написана функция delete\_bankAccount. Она принимает на вход ссылку на файловый поток с двоичным файлом, номер счета, запись с которого будет удалена, и название самого файла, чтобы открыть его повторно, но уже с другими параметрами. Функция возвращает 0, если запись удалена, и -1, если запись не найдена.

|  |
| --- |
| int delete\_bankAccount(fstream& fs, int cnt, string filename2) {  vector<bankAccount>records;  bankAccount rc;  while (true) {  fs.read(reinterpret\_cast<char\*>(&rc), sizeof(bankAccount));  if (fs.eof()) break;  records.push\_back(rc);  }  bool metka = false;  for (int i = 0; i < records.size(); i++) {  if (equal(records[i].accountNumber, cnt)) {  records.erase(records.begin() + i);  metka = true;  break;  }  }  if (!metka) return -1;  fs.close();  fs.open(filename2, ios::out);  for (int i = 0; i < records.size(); i++)  fs.write(reinterpret\_cast<char\*>(&records[i]), sizeof(bankAccount));  return 0;  } |

Для того, чтобы найти аккаунт по номеру счета была написана функция findAccount, которая в качестве аргумента получает на вход ссылку на файловый поток и номер счёта. Если аккаунт был найден функция возвращает структуру bankAccount, представляющую данную запись, иначе возвращает пустую структуру bankAccount.

|  |
| --- |
| bankAccount find\_account(fstream& fs, int accountNumber) {  bankAccount rc; // Создаем экземпляр структуры bankAccount для хранения найденной записи  while (true) {  fs.read(reinterpret\_cast<char\*>(&rc), sizeof(bankAccount));  if (fs.eof()) break;  if (rc.accountNumber == accountNumber) {  return rc;  }  }    return bankAccount();  } |

Для того, чтобы в нашем двоичном файле обновить запись об аккаунте по номеру счета, была написана функция updateAccount. Она получает ссылку на вкетор структур, число, представляющее собой номер банковсого счета и два указателя на строку символов, соответствующим новому имени и адресу соответственно. Функция ничего не возвращает.

|  |
| --- |
| void new\_record(fstream& fs, char wordd[10], string filename2) {  vector<record>records;  record rc;  while (true) {  fs.read(reinterpret\_cast<char\*>(&rc), sizeof(record));  if (fs.eof()) break;  records.push\_back(rc);  }  bool metka = true;  for (int i = 0; i < records.size(); i++) {  if (equal(wordd, records[i].word)) {  records[i].n++;  metka = false;  break;  }  }  if (metka) {  record rc\_new = { "", 1 };  set\_word(rc\_new, wordd);  records.push\_back(rc\_new);  }  fs.close();  fs.open(filename2, ios::out | ios::binary);  for (int i = 0; i < records.size(); i++)  fs.write(reinterpret\_cast<char\*>(&records[i]), sizeof(record));  } |

Чтобы добавить сразу несколько записей о словах была написана функция update\_records, которая получает ссылку на файловый поток, строку со всеми словами, которые будут добавлены, и название файла. Функция возвращает true либо false.

|  |
| --- |
| bool updateAccount(vector<bankAccount>& records, int accountNumber, const char\* newFullName, const char\* newAddress) {  for (int i = 0; i < records.size(); i++) {  if (records[i].accountNumber == accountNumber) {  // Обновляем поле ФИО  for (int j = 0; j < sizeof(records[i].fullName); j++) {  if (newFullName[j] == '\0') {  records[i].fullName[j] = '\0';  break;  }  else {  records[i].fullName[j]=newFullName[j];  }  }  // Обновляем поле адреса  for (int j = 0; j < sizeof(records[i].address); j++) {  if (newAddress[j] == '\0') {  records[i].address[j] = '\0';\  break;  }  else {  records[i].address[j] = newAddress[j];  }  }  return true;  }  }  return false;  } |

При запуске программы пользователя просят ввести название файла, после чего появляется список всех возможных операций (рис. 11).

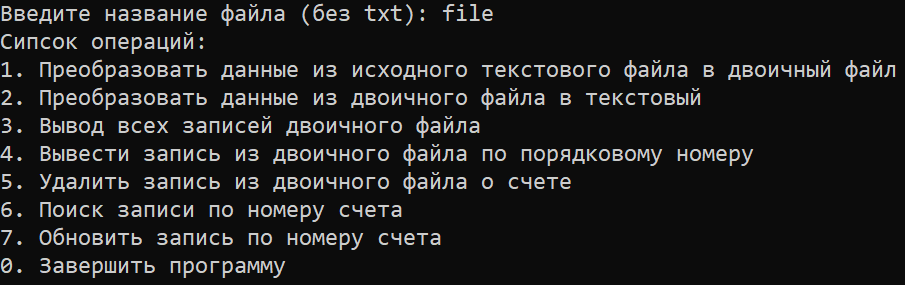


Рисунок 11. Пользовательское меню

# **Тестирование программы 2**

При запуске программу у нас создается только файл begin.txt, содержащий исходные данные. Попробуем его удалить и вызвать вторую операцию. Исходный файл не найден, поэтому вызовем первую операцию, которая создаст наш исходный файл (рис. 12). Проверим его наличие в папке (рис. 13).

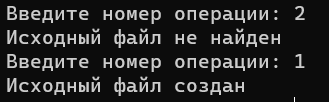


Рисунок 12. Вызов операции при удаленном исходном файле и его создание

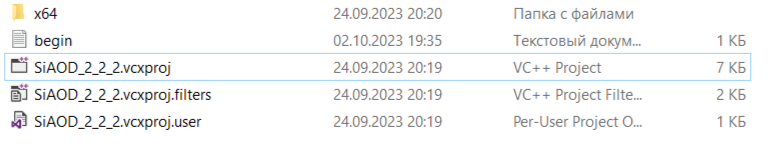


Рисунок 13. Папка проекта

Теперь попробуем создать двоичный файл на основе исходного (рис. 14). Файл действительно создан (рис. 15).



Рисунок 14. Создание двоичного файла

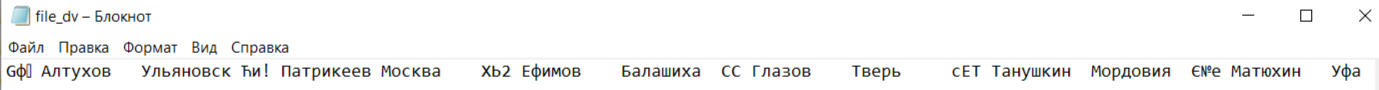


Рисунок 15. Двоичный файл

Попробуем теперь удалить двоичный файл и создать текстовый, на основе двоичного (рис. 16). Программа верно обрабатывает этот случай, поэтому создадим снова двоичный файл и вызовем эту же функцию (рис. 17). Посмотрим на созданный файл (рис. 18).

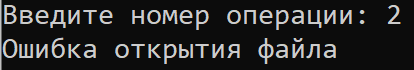


Рисунок 16. Вызов операции 2 при удаленном файле

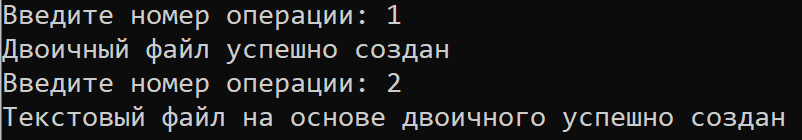


Рисунок 17. Создание текстового файла на основе двоичного

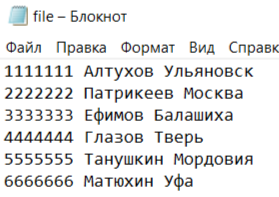


Рисунок 18. Созданный текстовый файл

Теперь будем тестировать операцию 3. Удалим двоичный файл, что посмотреть, обрабатывает ли программа этот случай. Далее заново создадим его и снова вызовем операцию 3, которая выводит содержимое двоичного файла на экран (рис. 19).

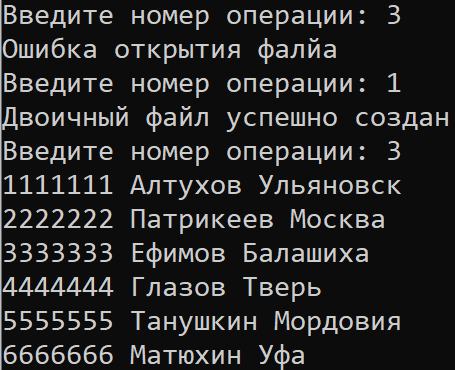


Рисунок 19. Вывод содержимого двоичного файла на экран

Проверим функцию, которая выводить запись из двоичного файла по порядковому номеру. Попробуем ввести существующий номер – 1. Программа должна вывести запись про Алтухова (рис. 20). Действительно, функция находит запись. Попробуем ввести номер 50 – которого нету в файле (рис. 21). Программа обработала этот случай.

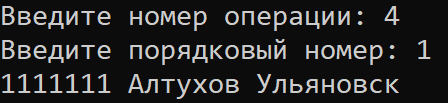


Рисунок 20. Поиск записи по существующему индексу

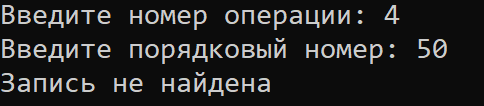


Рисунок 21. Поиск записи по несуществующему индексу

Попробуем удалить из файла запись, которой не существует (рис. 22). Теперь попробуем удалить запись об Алтухове и выведем на экран содержание двоичного файла (рис. 23). Программа срабатывает корректно.

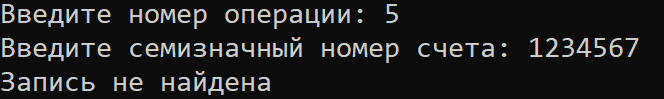


Рисунок 22. Удаление несуществующей записи

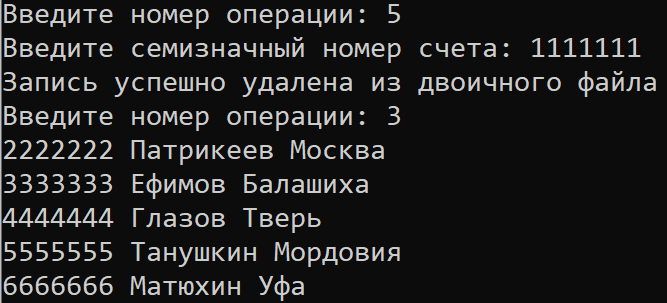


Рисунок 23. Удаление существующей записи

Попробуем найти запись по номеру счета. Сразу же выведем содержание записи, чтобы проверить, правильно ли программа нашла запись (рис. 24).

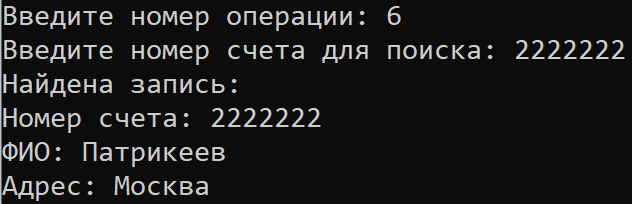


Рисунок 24. Поиск записи по номеру счета

Теперь попробуем обновить запись по номеру счета. Для наглядности выведем содержимое файла на экран до добавления записи и после (рис. 25). Программа успешно обновила заданную запись.

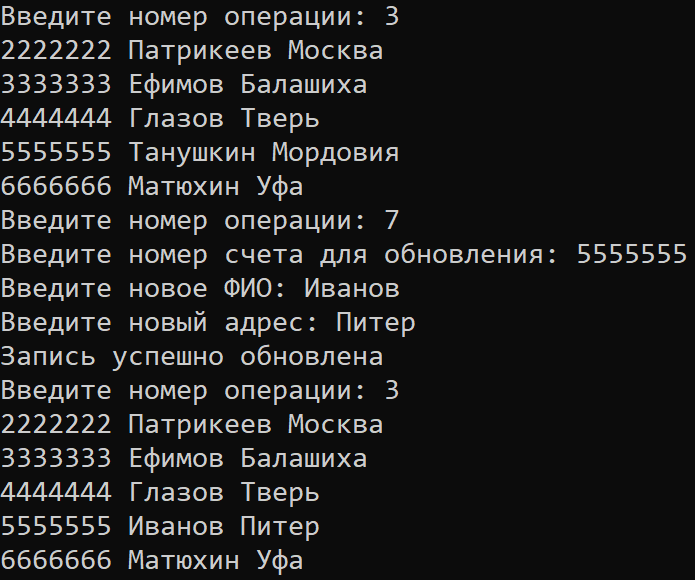


Рисунок 25. Обновление записи по номеру счета

Рассмотрим теперь случай, когда номер счета не найден (рис. 26). В таком случае, программа выведет на экран соответствующее сообщение.

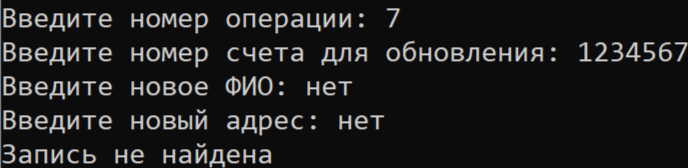


Рисунок 26. Обновление записи по номеру счета, который не найдем

# **Вывод**

В результате выполнения работы я освоил работу с файловыми потоками, реализовал текстовые и двоичные файлы. Также я научился записывать в них данные о структурах и изменять их.

# **Исходный код программы 1**

|  |
| --- |
| #include<iostream>  #include<fstream>  #include<string>  #include<vector>  #include <sstream>  #include <windows.h>    using namespace std;    // 1.3.3 добавление новой строки в конец файла  int writeToFile(string& filename) {    ofstream file(filename, fstream::app);    if (!file.is\_open()) {  return -1;  }  string data;  getline(cin, data);  getline(cin, data);  file << data<< endl;    if (file.good()) {  file.close();  return 0;  }  else {  file.close();  return -2;  }  }    // 1.3.2 вывод содержимого текстового файла  int readFromFile(string& filename) {  ifstream file(filename);  if (!file.is\_open()) {  return -1;  }  string line;    while (getline(file, line)) {  cout << line << endl;  }    if (file.good()) {  file.close();  return 0;  }  else {  file.close();  return -2;  }    }    //1.3.1 создание текстового файла кодировки ASCII  int createSampleFile(string& filename) {  cout << "Ввести числа\n1-автоматически\n2-самому\n";  int n;  cin >> n;  ofstream file(filename, fstream::app);  if (!file.is\_open()) {  return -1;  }    if (n == 1) {  //Автоматическое заполнения файла  file << "10 21 38 40 56" << endl;  file << "63 70 80 91 100" << endl;  file << "112 120 133 140 150"<< endl;      cout << "Файл создан" << endl;    }  else {  //Ручное заполнение файла    string line;    cout << "Введите десятичные числа (разделенные пробелами, введите 'exit' для завершения):" << endl;  getline(cin, line);  if (file.good()) {  while (true) {  getline(cin, line);    if (line == "exit") { break; }  file << line<< endl;    }  }    cout << "Файл создан" << endl;  }  if (file.good()) {  file.close();  return 0;  }  else {  file.close();  return -2;  }  }  //1.3.4 прочитать значение числа, указав его порядковый номер в файле, и вернуть его значение    int read\_number\_by\_index(string file\_name, int index) {  ifstream file(file\_name);  if (!file.is\_open()) {  return -1;  }  string line;  int count = 0;  int cnt = 0;  if (file.good()) {  while (getline(file, line)) {  istringstream iss(line);  int num;  while (iss >> num) {  count++;  if (count == index) {  cout << "Искомое число: " << num << endl;  cnt += 1;  break;  }  }  }  }  if(cnt==0){ cout << "Искомое число не найдено " << endl; }  if (file.good()) {  file.close();  return 0;  }  else {  file.close();  return -2;  }  }    //1.3.5 Определение количества чисел в файле    int count\_numbers(string file\_name) {  ifstream file(file\_name);  if (!file.is\_open()) {  return -1;  }  string line;  int count = 0;  while (getline(file, line)) {  istringstream iss(line);  int num;  while (iss >> num) {  count++;  }  }  cout << "Количество чисел: " << count<<endl;  if (file.good()) {  file.close();  return 0;  }  else {  file.close();  return -2;  }  }    //1.3.6 Скопировать числа исходного файла, которые кратны 7, в новый файл.  int create\_new\_file(string file\_name) {  ifstream file(file\_name);  if (!file.is\_open()) { return -1; }  //Создается вектор numbers, в котором будут храниться числа из исходного файла.  vector<int> numbers;  string line;  while (getline(file, line)) {  //Создается объект istringstream для каждой строки, который позволяет преобразовать строку в поток и считывать числа из нее с помощью оператора >>  istringstream iss(line);  int num;  while (iss >> num) {  numbers.push\_back(num);  }  }  vector<string> new\_numbers;  for (int num : numbers) {  if (num % 7 == 0) { new\_numbers.push\_back(to\_string(num)); }  }  ofstream new\_file("new\_" + file\_name);  for (string num\_str : new\_numbers) {  cout << num\_str << " ";  new\_file << num\_str << " ";  }  cout << endl;  if (file.good())  {  file.close();  return 0;  }  else {  file.close();  return -2;  }  }  int main()  {  SetConsoleCP(1251);  setlocale(LC\_ALL, "ru");  cout << "Введите имя файла: ";  string str\_file;  cin >> str\_file;    fstream file;  cout << "Операции над текстовым файлом:\n";  cout << "1-Cоздание текстового файла кодировки ASCII, содержащего десятичные числа по несколько чисел на строке; файл может быть образцовым или введен пользователем\n";  cout << "2-Вывод содержимого текстового файла\n";  cout << "3-Добавление новой строки в конец файла\n";  cout << "4-Прочитать значение числа, указав его порядковый номер в файле, и вернуть его значение\n";  cout << "5-Определить количество чисел в файле\n";  cout << "6-Скопировать числа исходного файла, которые кратны 7, в новый файл.\n";  cout << "7-выход\n";  int n;  cin >> n;  while (n != 7) {  switch (n)  {  case 1:  if (!file.is\_open()){  file.open(str\_file, ios::app);  }  createSampleFile(str\_file);  break;  case 2:  if (!file.is\_open()) {  file.open(str\_file, ios::in);  }  readFromFile(str\_file);  break;  case 3:  cout << "Введите строку из чисел: ";  if (!file.is\_open()) {  file.open(str\_file, ios::app);  }  writeToFile(str\_file);  break;  case 4:  if (!file.is\_open()) {  file.open(str\_file, ios::in);  }  int index;  cout << "Введите порядковый номер искомого числа\n";  cin >> index;  read\_number\_by\_index(str\_file, index);  break;  case 5:  if (!file.is\_open()) {  file.open(str\_file, ios::in);  }  count\_numbers(str\_file);  break;  case 6:  if (!file.is\_open()) {  file.open(str\_file, ios::in);  }  create\_new\_file(str\_file);  break;  default:  break;  }  cout << "Введите номер опреации над текстовым файлом: ";  cin >> n;  }  } |

# **Исходный код программы 2**

|  |
| --- |
| #include <iostream>  #include <locale.h>  #include <windows.h>  #include <string>  #include <fstream>  #include <vector>  using namespace std;  void create\_begin\_file() {  fstream fs;  fs.open("begin.txt", ios::out);  fs << "1111111 Алтухов Ульяновск\n2222222 Патрикеев Москва\n3333333 Ефимов Балашиха\n4444444 Глазов Тверь\n5555555 Танушкин Мордовия\n6666666 Матюхин Уфа";  fs.close();  }  struct bankAccount {  int accountNumber;  char fullName[10];  char address[10];  };  void set\_word(bankAccount& rc, char fullName[10], char address[10]) {  for (int i = 0; i < 10; i++) {  rc.fullName[i] = fullName[i];  rc.address[i] = address[i];  }  }  void clear(char (&fullName)[10], char(&address)[10]) {  for (int i = 0; i < 10; i++) {  fullName[i] = '\0';  address[i] = '\0';  }  }  // 1. функция которая преобразовывает данные из исходного текстового файла в двоичный файл  void make\_dv(fstream& fs\_o, string filename) {  fstream fs\_n;  fs\_n.open(filename, ios::out | ios::binary);  vector<bankAccount>records;  int accountNumber;  char fullName[10];  char address[10];  while (!fs\_o.eof()) {  clear(fullName, address);  fs\_o >> accountNumber >> fullName >> address ;  bankAccount rc;  set\_word(rc, fullName, address);  rc.accountNumber = accountNumber;  records.push\_back(rc);  }  for (int i = 0; i < records.size(); i++)  fs\_n.write(reinterpret\_cast<char\*>(&records[i]), sizeof(bankAccount));  fs\_n.close();  }  // 2. функция которая преобразовывает данные из двоичного файла в текстовый  void make\_text(fstream& fs\_d, string& filename) {  fstream fs\_t;  fs\_t.open(filename, ios::out);  vector<bankAccount>records;  bankAccount rc;  while (true) {  fs\_d.read(reinterpret\_cast<char\*>(&rc), sizeof(bankAccount));  if (fs\_d.eof()) break;  records.push\_back(rc);  }  for (int i = 0; i < records.size(); i++) {  fs\_t << records[i].accountNumber << " " << records[i].fullName << " " << records[i].address << "\n";  }  fs\_t.close();  }  // 3. функция которая выводит все записи двоичного файла  void print\_binary(fstream& fs) {  vector<bankAccount>records;  bankAccount rc;  while (true) {  fs.read(reinterpret\_cast<char\*>(&rc), sizeof(bankAccount));  if (fs.eof()) break;  records.push\_back(rc);  }  for (int i = 0; i < records.size(); i++)  cout << records[i].accountNumber << " " << records[i].fullName << " " << records[i].address << endl;  }  // 4. функция которая выводит запись из двоичного файла по порядковому номеру  bankAccount get\_bankAccount(fstream& fs, int num) {  vector<bankAccount>records;  bankAccount rc;  while (true) {  fs.read(reinterpret\_cast<char\*>(&rc), sizeof(bankAccount));  if (fs.eof()) break;  records.push\_back(rc);  }  bankAccount res = {0, "", ""};  if (num >= records.size()) return res;  res = records[num];  return res;  }  bool equal(int w1, int w2) {  if (w1 != w2)  return false;  else if (w1 == '\0');  return true;  }  // 5. функция которая удаляет запись из двоичного файла о счете  int delete\_bankAccount(fstream& fs, int cnt, string filename2) {  vector<bankAccount>records;  bankAccount rc;  while (true) {  fs.read(reinterpret\_cast<char\*>(&rc), sizeof(bankAccount));  if (fs.eof()) break;  records.push\_back(rc);  }  bool metka = false;  for (int i = 0; i < records.size(); i++) {  if (equal(records[i].accountNumber, cnt)) {  records.erase(records.begin() + i);  metka = true;  break;  }  }  if (!metka) return -1;  fs.close();  fs.open(filename2, ios::out);  for (int i = 0; i < records.size(); i++)  fs.write(reinterpret\_cast<char\*>(&records[i]), sizeof(bankAccount));  return 0;  }  // 6. Функция для поиска записи по номеру счета  bankAccount find\_account(fstream& fs, int accountNumber) {  bankAccount rc; // Создаем экземпляр структуры bankAccount для хранения найденной записи  while (true) {  fs.read(reinterpret\_cast<char\*>(&rc), sizeof(bankAccount));  if (fs.eof()) break;  if (rc.accountNumber == accountNumber) {  return rc;  }  }    return bankAccount();  }  // 7. Функция которая обновляет запись по номеру счета  bool updateAccount(vector<bankAccount>& records, int accountNumber, const char\* newFullName, const char\* newAddress) {  for (int i = 0; i < records.size(); i++) {  if (records[i].accountNumber == accountNumber) {  // Обновляем поле ФИО  for (int j = 0; j < sizeof(records[i].fullName); j++) {  if (newFullName[j] == '\0') {  records[i].fullName[j] = '\0'; // Завершаем строку  break;  }  else {  records[i].fullName[j] = newFullName[j];  }  }  // Обновляем поле адреса  for (int j = 0; j < sizeof(records[i].address); j++) {  if (newAddress[j] == '\0') {  records[i].address[j] = '\0'; // Завершаем строку  break;  }  else {  records[i].address[j] = newAddress[j];  }  }  return true;  }  }  return false;  }  int main() {  setlocale(LC\_ALL, "rus");  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  create\_begin\_file();  fstream fs;  string filename;  cout << "Введите название файла (без txt): ";  cin >> filename;  string filename2 = filename + "\_dv.txt";  filename += ".txt";  cout << "Сипсок операций:\n";  cout << "1. Преобразовать данные из исходного текстового файла в двоичный файл\n";  cout << "2. Преобразовать данные из двоичного файла в текстовый\n";  cout << "3. Вывод всех записей двоичного файла\n";  cout << "4. Вывести запись из двоичного файла по порядковому номеру\n";  cout << "5. Удалить запись из двоичного файла о счете\n";  cout << "6. Поиск записи по номеру счета\n";  cout << "7. Обновить запись по номеру счета\n";  cout << "0. Завершить программу\n";  int m = 1;  while (m != 0) {  cout << "Введите номер операции: ";  cin >> m;  switch (m) {  case 1: {  fs.open("begin.txt", ios::in);  if (!fs.is\_open()) {  cout << "Исходный файл не найден\n";  break;  }  make\_dv(fs, filename2);  fs.close();  cout << "Двоичный файл успешно создан" << endl;  break;  }  case 2: {  fs.open(filename2, ios::in | ios::binary);  if (!fs.is\_open()) {  cout << "Ошибка открытия файла\n";  break;  }  make\_text(fs, filename);  cout << "Текстовый файл на основе двоичного успешно создан" << endl;  fs.close();  break;  }  case 3: {  fs.open(filename2, ios::in | ios::binary);  if (!fs.is\_open()) {  cout << "Ошибка открытия фалйа\n";  break;  }  print\_binary(fs);  fs.close();  break;  }  case 4: {  fs.open(filename2, ios::in | ios::binary);  if (!fs.is\_open()) {  cout << "Ошибка открытия фалйа\n";  break;  }  int num;  cout << "Введите порядковый номер: ";  cin >> num;  bankAccount res = get\_bankAccount(fs, num-1);  if (res.accountNumber == 0)  cout << "Запись не найдена" << endl;  else  cout << res.accountNumber << " " << res.fullName << " " << res.address << endl;  fs.close();  break;  }  case 5: {  fs.open(filename2, ios::in | ios::binary);  if (!fs.is\_open()) {  cout << "Ошибка открытия фалйа\n";  break;  }  int cnt;  cout << "Введите семизначный номер счета: ";  cin >> cnt;  if (delete\_bankAccount(fs, cnt, filename2) == 0) {  cout << "Запись успешно удалена из двоичного файла" << endl;  }  else cout << "Запись не найдена" << endl;  fs.close();  break;  }    case 6: {  fs.open(filename2, ios::in | ios::binary);  if (!fs.is\_open()) {  cout << "Ошибка открытия фалйа\n";  break;  }  int accountNumber;  cout << "Введите номер счета для поиска: ";  cin >> accountNumber;  bankAccount foundAccount = find\_account(fs, accountNumber);  if (foundAccount.accountNumber != 0) {  cout << "Найдена запись:\n";  cout << "Номер счета: " << foundAccount.accountNumber << endl;  cout << "ФИО: " << foundAccount.fullName << endl;  cout << "Адрес: " << foundAccount.address << endl;  }  else {  cout << "Запись не найдена." << endl;  }  fs.close();  break;  }  case 7: {  fs.open(filename2, ios::in | ios::binary);  if (!fs.is\_open()) {  cout << "Ошибка открытия файла" << endl;  break;  }  int accountNumber;  cout << "Введите номер счета для обновления: ";  cin >> accountNumber;  char newFullName[10];  char newAddress[10];  cout << "Введите новое ФИО: ";  cin >> newFullName;  cout << "Введите новый адрес: ";  cin >> newAddress;  vector<bankAccount> records;  bankAccount rc;  // Перемещаем указатель файла в начало  fs.seekg(0, ios::beg);  // Читаем все записи из файла  while (true) {  fs.read(reinterpret\_cast<char\*>(&rc), sizeof(bankAccount));  if (fs.eof()) break;  records.push\_back(rc);  }  if (updateAccount(records, accountNumber, newFullName, newAddress)) {  // перезаписываем все записи  fs.close();  fs.open(filename2, ios::out);  for (int i = 0; i < records.size(); i++)  fs.write(reinterpret\_cast<char\*>(&records[i]), sizeof(bankAccount));  cout << "Запись успешно обновлена" << endl;  }  else {  cout << "Запись не найдена" << endl;  }  fs.close();  break;  }  case 0:  break;  default:  cout << "Введен некорректный номер операции\n";  break;  }  }  return 0;  } |