|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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-10-21 | Смольников А.Б. |
| Принял преподаватель | Филатов А.С. |

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

Москва 2022

# **Цель работы**

Получить навыки применения файловых потоков языка С++ по управлению текстовым и двоичным файлами.

# **Постановка задачи**

* 1. **Задача 1**

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 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.
   9. **Задача 2**
2. 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. Сохраните 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.

Таблица 1. Задания варианта №22

|  |  |
| --- | --- |
| Задание 1 | Создать новый файл из значений исходного, размещая на каждой строке заданное количество чисел, кроме, возможно, последней, где должны разместиться оставшиеся числа. |
| Задание 2 | Расписание занятий группы: номер группы, название дисциплины, номер пары, номер недели, номер дня недели, вид занятия, номер аудитории.   1. Сформировать в двоичном файле расписание заданной группы на заданный день недели. 2. Обновить расписание, найти дисциплины, которые стоят на одинаковых парах в одной аудитории, и определить для них новые аудитории |

# **Решение**

* 1. **Теоретическое введение**

В С++ открытый файл может быть представлен как последовательность считываемых или записываемых данных. При открытии файла используется поток ввода и вывода. Для работы с файлами используется библиотека <fstream>, в которую включены <ifstream> и <ofstream> - ввод и вывод соответственно.

В данной библиотеке предусмотрены режимы открытия файлов, которые устанавливают характер использования файлов. Для установки режима в классе ios\_base предусмотрены константы:

Таблица 2. Режимы открытия файлов

|  |  |
| --- | --- |
| Константа | Описание |
| ios::in | Открыть файл для чтения |
| ios::out | Открыть файл для записи |
| ios::ate | Открыть файл и переместить указатель в конец |
| ios::app | Открыть файл для дозаписи |
| ios::trunc | Удалить содержимое файла |
| ios::binary | Открыть файл в двоичном режиме |

Режимы открытия можно комбинировать при помощи логического ИЛИ “|”.

После открытия файла для проверки можно использовать метод file.is\_open() или же good(). Для прямого доступа к файлу используются методы seekg() и seekp(). Они позволяют переместить указатель. Можно определить текущую позицию файлового указателя, используя следующие функции: streampos tellg() - позиция для ввода и streampos tellp() - позиция для вывода.

* 1. **Функции задания №1**

Функция genetateFile() принимает на вход имя файла и генерирует файл со случайным количеством строк и случайным количеством чисел в строке при помощи Вихря Мерсенна

|  |
| --- |
| //генерирует файл  bool generateFile(std::string name){  std::ofstream file(name, std::ios::out);  if (file.is\_open()){  std::random\_device rd;  std::mt19937 gen(rd());  std::uniform\_int\_distribution<> dis(10, 10);  int n = dis(gen);  for(int i=0;i<n;i++){  for(int j=0;j<dis(gen);j++){  file << dis(gen) << " ";  }  file << std::endl;  }  file.close();  return true;  }  return false;  } |

Функция printFile() принимает на имя файла и выводит на экран весь файл.

|  |
| --- |
| //Функция вывода файла на экран  bool printFile(std::string name){  std::ifstream file(name, std::ios::in);  if (file.is\_open()){  std::string line;  while (getline(file, line)){  std::cout << line << std::endl;  }  file.close();  return true;  }  return false;  } |

Функция appendLine() принимает на вход строку – имя файла и строку – линию для добавления и добавляет в переданный файл строку.

|  |
| --- |
| //Функция добавления строки в конец файла  bool appendLine(std::string name, std::string line){  std::ofstream file(name, std::ios::app);  if (file.is\_open()){  file << line << std::endl;  file.close();  return true;  }  return false;  } |

Функция readNumber () принимает на вход строку – имя файла и целое число – позицию, возвращает целое число - искомое

|  |
| --- |
| //Функция чтения числа по его позиции  int readNumber(std::string name, int pos){  std::ifstream file(name, std::ios::in);  if (file.is\_open()){  std::string line;  int i = 0;  while (getline(file, line)){  std::stringstream ss(line);  int n;  while (ss >> n){  if (i == pos){  file.close();  return n;  }  i++;  }  }  file.close();  return NAN;  }  return NAN;  } |

Функция countNumbers() принимает на вход строку – имя файла и возвращает целое – цоличество чисел.

|  |
| --- |
| //Функция подсчета количества чисел  int countNumbers(std::string name){  std::ifstream file(name, std::ios::in);  if (file.is\_open()){  std::string line;  int i = 0;  while (getline(file, line)){  std::stringstream ss(line);  int n;  while (ss >> n){  i++;  }  }  file.close();  return i;  }  return NAN;  } |

Функция copyFormatted () принимает на вход строку – имя файла, строку – имя нового файла и целое число – ширину. Возвращает логическое значение – признак успешного форматирования и копирования.

|  |
| --- |
| int copyFormatted(std::string name, std::string cp\_name , int width){  std::ifstream file(name, std::ios::in);  if (file.is\_open()){  std::ofstream cp\_file(cp\_name, std::ios::out);  if (cp\_file.is\_open()){  std::string line;  int cnt = 0;  while (getline(file, line)){  std::stringstream ss(line);  int n;  while (ss >> n){  if(cnt == width){  cp\_file << std::endl;  cnt = 0;  }  cnt++;  cp\_file << n << " ";  }  }  cp\_file.close();  file.close();  return true;  }  file.close();  return false;  }  return false;  } |

* 1. **Функции задания №2**

Функция convertToBitFile() принимает на вход строку - имя файла и строку – имя бинарного файла, возвращает логическое значение – признак успешности конвертации

|  |
| --- |
| //Функция конвертации в бинарный файл  bool convertToBitFile(std::string fileName, std::string bitFileName){  std::ifstream file(fileName, std::ios::in);  if (!file.is\_open()){  throw std::runtime\_error("File not found");  }  std::ofstream bitFile(bitFileName, std::ios::binary | std::ios::out);  if (!bitFile.is\_open()){  throw std::runtime\_error("File not found");  }  bitFile << file.rdbuf();  bitFile.close();  file.close();  return true;  } |

Функция convertToTextFile() принимает на вход строку – имя бинарного файла и строку – имя текстового файла, возвращает логическую переменную – признак успешности конвертации

|  |
| --- |
| //Функция конвертации из бинарного файла в текстовый  bool convertToTextFile(std::string bitFileName, std::string fileName){  std::ifstream bitFile(bitFileName, std::ios::binary | std::ios::in);  if (!bitFile.is\_open()){  throw std::runtime\_error("File not found");  }  std::ofstream file(fileName, std::ios::out);  if (!file.is\_open()){  throw std::runtime\_error("File not found");  }  file<<bitFile.rdbuf();  bitFile.close();  file.close();  return true;  } |

Функция printBitFile() принимает на вход строку – имя файла и выводит содержимое файла на экран.

|  |
| --- |
| //Функция вывода бинарного файла  bool printBitFile(std::string bitFileName){  std::ifstream bitFile(bitFileName, std::ios::binary | std::ios::in);  if (!bitFile.is\_open()){  throw std::runtime\_error("File not found");  }  std::cout<<"номер группы | название дисциплины | номер пары | номер недели | номер дня недели | вид занятия | номер аудитории"<<std::endl;  std::cout<<bitFile.rdbuf();  bitFile.close();  return true;  } |

Функция directAccess() принимает на вход строку – имя файла, целое число – позицию и целое – длина записи (по умолчанию 32), выводит на экран искомую запись и возвращает логическое значение – признак успеха.

|  |
| --- |
| //Функция чтения числа по его позиции  bool directAccess(std::string bitFileName, int number, int length){  std::ifstream bitFile(bitFileName, std::ios::binary | std::ios::in);  if (!bitFile.is\_open()){  throw std::runtime\_error("File not found");  }  bitFile.seekg(0, std::ios::end);  int size = bitFile.tellg();  if ((length\*sizeof(char)+10)\*number > size){  bitFile.close();  return false;  }  bitFile.seekg((length\*sizeof(char)+10)\*number);  std::string s;  std::getline(bitFile, s);  std::cout<<s<<std::endl;  bitFile.close();  return true;  } |

Функция deleteByKey() принимает на вход строку – имя файла, строку – ключ и длину записи. Возвращает логическую переменную – признак успешности операции.

|  |
| --- |
| //Функция удаления записи по ключу  bool deleteByKey(std::string bitFileName, std::string key, int length){  std::ifstream bitFile(bitFileName, std::ios::binary | std::ios::in);  if (!bitFile.is\_open()){  throw std::runtime\_error("File not found");  }  std::ofstream tempFile("temp.bin", std::ios::out | std::ios::binary);  if (!tempFile.is\_open()){  bitFile.close();  throw std::runtime\_error("File not found");  }  bitFile.seekg(0, std::ios::end);  int size = bitFile.tellg();  bitFile.seekg(0, std::ios::beg);  bitFile.seekg(size - length-9);  std::string temp;  std::getline(bitFile, temp);  std::string tempNum = temp.substr(0, temp.find(' '));  temp = key+ temp.substr(key.size(), temp.size());  bitFile.seekg(0, std::ios::beg);  std::string s;  while (!bitFile.eof()){  std::getline(bitFile, s);  if(s.substr(0, key.size()) != key && s.substr(0, key.size()) != tempNum){  tempFile<<s<<std::endl;  } else if (s.substr(0, key.size()) == tempNum){  }else{  tempFile << temp << std::endl;  }  }  bitFile.close();  tempFile.close();  std::remove(bitFileName.c\_str());  if(std::rename("temp.bin", bitFileName.c\_str())==0){  return true;  } else{  return false;  }  } |

Функция splitter() принимает на вход строку – имя файла, строку –разделитель и целое число – длину. Возвращает указатель на массив строк – разделенная строка.

|  |
| --- |
| std::string\* splitter (std::string s, std::string delim = " ", int size =8){  std::string\* result = new std::string[size];  int i = 0;  while (s.find(delim) != std::string::npos){  result[i] = s.substr(0, s.find(delim));  s = s.substr(s.find(delim)+1, s.size());  i++;  }  result[i] = s;  return result;  } |
|  |

Функция generateByWeekDay() принимает на вход строку – имя файла, строку – группу и строку - день. Формирует в новом файле расписание группы на определенную неделю.

|  |
| --- |
| void generateByWeekDay(std::string bitFileName, std::string group, std::string day){  std::ifstream bitFile(bitFileName, std::ios::binary | std::ios::in);  if (!bitFile.is\_open()){  throw std::runtime\_error("File not found");  }  std::ofstream file("generated.bin", std::ios::out | std::ios::binary);  std::string s;  while (!bitFile.eof()){  std::getline(bitFile, s);  std::string\* temp = splitter(s);  if (temp[1] == group && temp[5] == day){  file<<s<<std::endl;  }  }  bitFile.close();  file.close();  } |

Функция updateSchedule() принимает на вход строку – имя файла, строку – имя нового файла. Создает новый файл с корректированным расписанием без коллизий.

|  |
| --- |
| void updateSchedule(std::string bitFileName, std::string newFileName){  std::ifstream bitFile(bitFileName, std::ios::binary | std::ios::in);  if (!bitFile.is\_open()){  throw std::runtime\_error("File not found");  }  std::ofstream tempFile(newFileName, std::ios::out | std::ios::binary);  if (!tempFile.is\_open()){  bitFile.close();  throw std::runtime\_error("File not found");  }  customVector scheudele;  std::string s;  while (!bitFile.eof()){  std::getline(bitFile, s);  std::string\* temp = splitter(s);  // s = temp[0] + " " + temp[3] + " " + temp[4] + " " + temp[5] + " " + temp[7];  scheudele.push\_back(s);  }  scheudele.print();  for(int i=0;i<scheudele.size;i++){  std::string\* temp = splitter(scheudele.data[i]);  // for(int z=0;z<8;z++){  // tempFile<<"!"<<temp[z];  // }  // tempFile<<std::endl;  for(int j=i+1;j<scheudele.size;j++){  std::string\* temp2 = splitter(scheudele.data[j]);  if(temp[7]==temp2[7] && temp[3]==temp2[3] && temp[4]==temp2[4] && temp[5]==temp2[5]){  scheudele.data[i] = temp[0]+ " " +temp[1] + " " + temp[2] + " " + temp[3] + " " + temp[4] + " " + temp[5] + " " + temp[6] + " " + std::to\_string(200+ rand()%100);  }  }  }  for(int i=0;i<scheudele.size;i++){  tempFile<<scheudele.data[i]<<std::endl;  }  bitFile.close();  tempFile.close();  } |

* 1. **Интерфейс**

При запуске программы пользователю предоставляется выбор одного из двух заданий при помощи ввода чисел 1-2, после выбора предлагается подвыбор любой из необходимых функций. Любой другой введенный символ спровоцирует завершение программы.

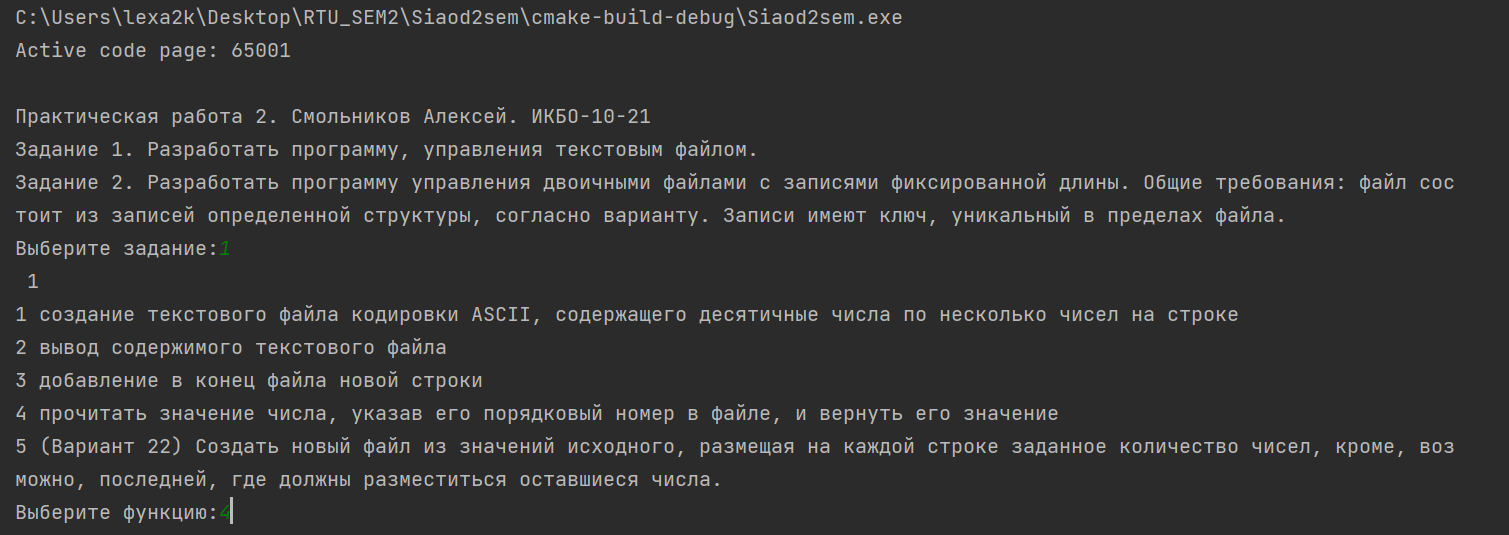


Рисунок 1. Интерфейс программы

# **Тестирование**

В силу простоты вычислений совместим тестовый прогон с рабочим, проверив каждое задание по одному разу.

Функция generateFile() генерирует случайным образом файл:

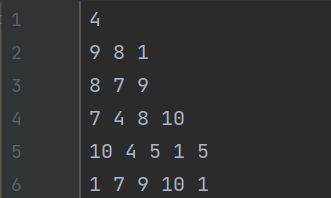


Рисунок 1. Случайно сгенерированный файл

Функция printFile() выводит сгенерированный файл в консоль:

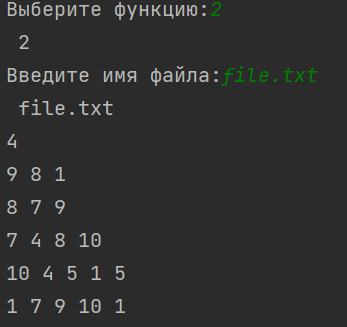


Рисунок 2. Вывод текстового файла

Функция appendLine() добавляет строчку в файл

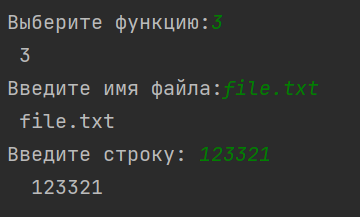


Рисунок 3. Ввод строки для добавления

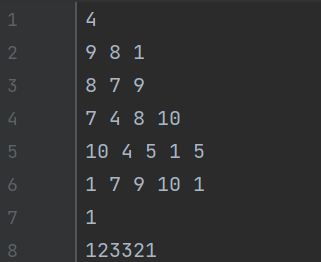


Рисунок 4. Файл после добавления строки

Функция readNumber() считывает из файла число по порядковому номеру (начиная с нуля), тогда проведем тестирование из файла на рисунке 4.

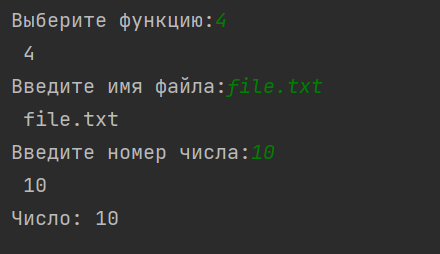


Рисунок 5. Результат работы функции

Функция copyFormatted() форматирует файл в соответствии с заданием 5 и записывает данные в новый файл:

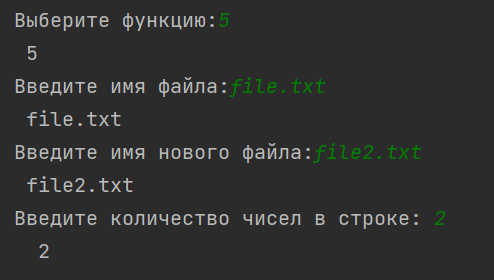


Рисунок 6. Интерфейс реформатирования файла

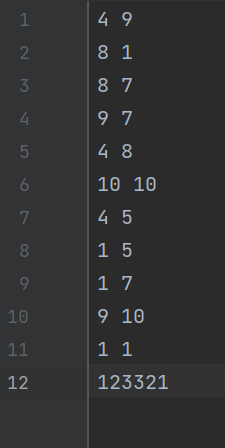


Рисунок 7. Результат работы функции задания 5

Рассмотрим структуру бинарного файла, используемого для тестирования программы (длина – 32 символа):

|  |
| --- |
| 001 ИКБО-10-21 Пр1 2 1 7 сем 201  002 ИКБО-01-21 Пр2 2 1 7 сем 201  003 ИКБО-02-21 Пр3 2 1 7 сем 203  004 ИКБО-03-21 пр4 2 1 4 сем 203  005 ИКБО-03-21 пр4 2 1 4 сем 204 |
|  |

Функция convertToBitFile() конвертирует текстовый файл в бинарный, если файл не найден, вызывается исключение:

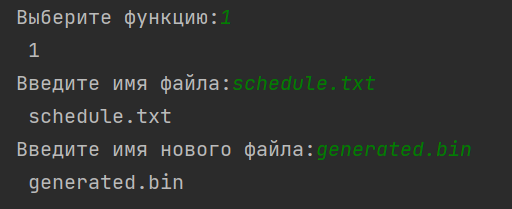


Рисунок 8. Интерфейс функции

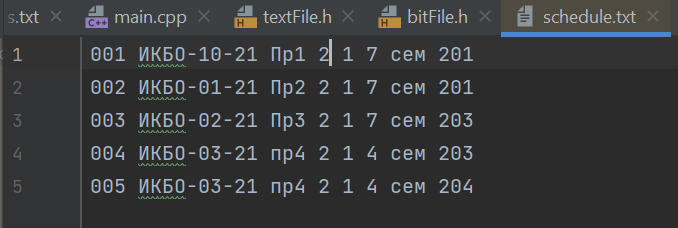


Рисунок 9. Текстовый файл

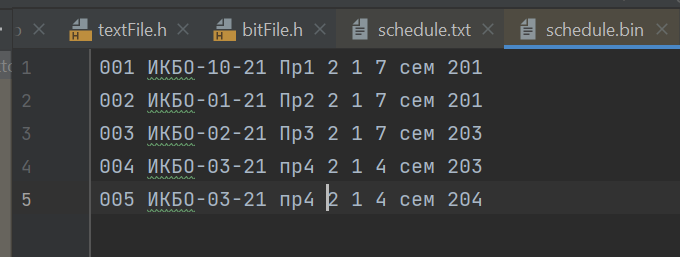


Рисунок 10. Бинарный файл

Функция convertToTextFile() выполняет функцию, обратную предыдущей:

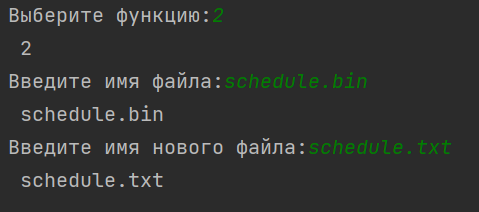


Рисунок 11. Интерфейс функции

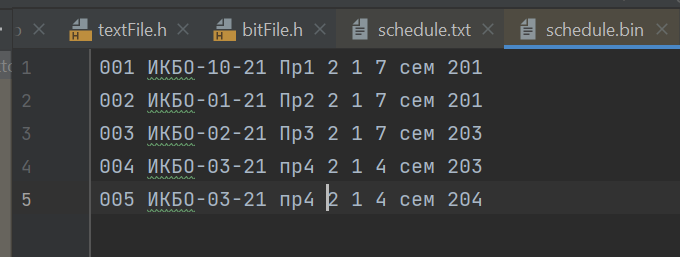


Рисунок 12. Бинарный файл

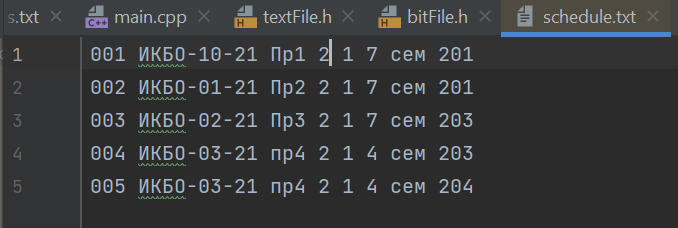


Рисунок 13. Текстовый файл

Функция printBitFile() считывает и выводит данные из бинарного файла, отсутствие файла вызывает исключение:

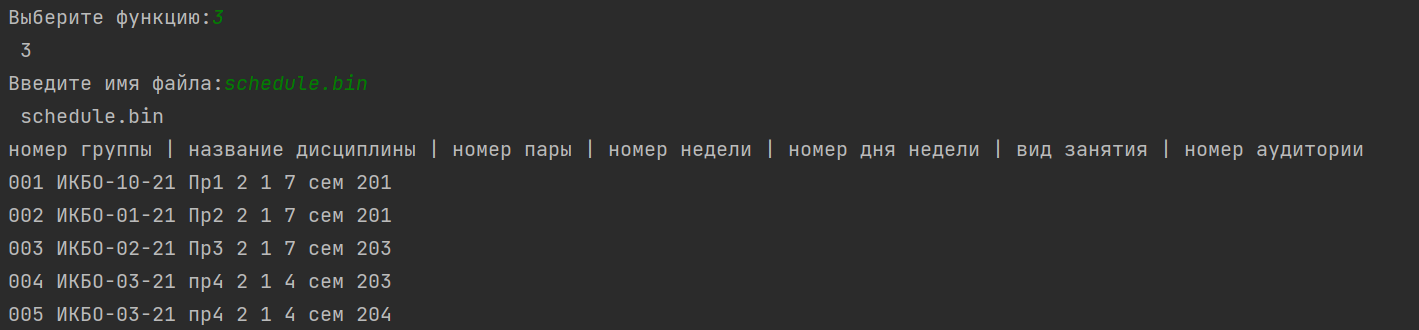


Рисунок 14. Результат работы функции printBitFile()

Функция directAccess() получает прямой доступ к файлу для получения записи:

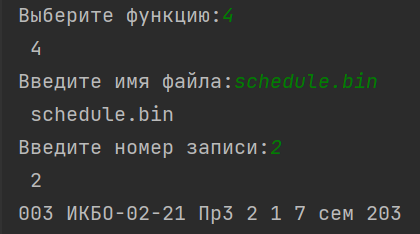


Рисунок 15. Результат работы функции directAccess()

Функция deleteByKey() удаляет запись по ключу по условию поставленной задачи:

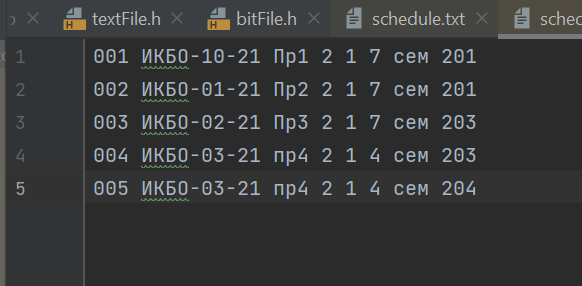


Рисунок 16. Файл до удаления

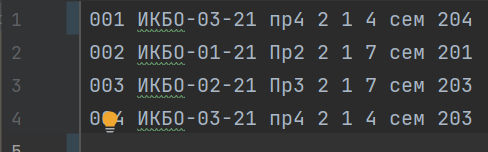


Рисунок 17. Файл после модификации

Функция generateByWeekDay() генерирует файл в соответствии с заданием. Неверный файл вызывает исключение.

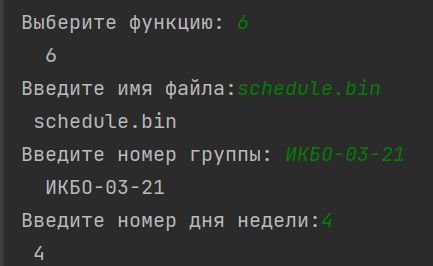


Рисунок 18. Интерфейс функции generateByWeekDay()

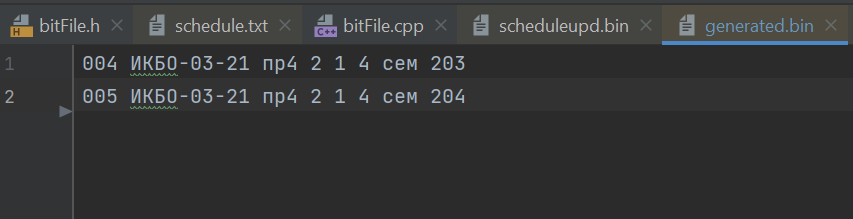


Рисунок 19. Итоговый файл

Функция updateSchedule() разрешает коллизии в расписании путем выбора случайной аудитории. Неверный файл вызовет исключение.

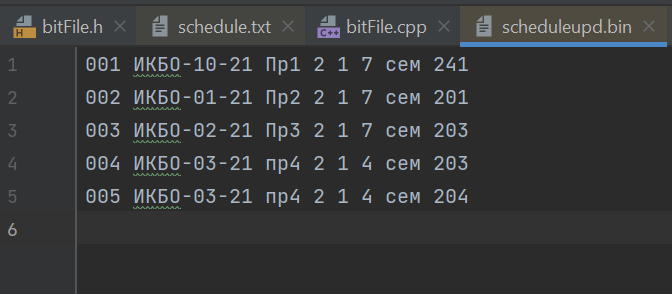


Рисунок 20. Файл после работы функции updateSchedule()

Из результатов выполнения программы видно:

1. Функции задания 1 работают корректно
2. Функции задания 2 работают корректно

# **Выводы**

В результате выполнения работы я:

1. Освоил файловые потоки в С++
2. Научился работать с текстовыми и бинарными файлами
3. Протестировал все функции и удостоверился в их правильности

# **Исходный код программы**

main.cpp

|  |
| --- |
|  |
| //  // Created by lexa2k on 16.09.2022.  //  //https://drive.google.com/drive/folders/17AVqcu5P2GEqUStdI9vWSBUOAJ1k4Vz2  #include <iostream>  #include <string>  #include "textFile.h"  #include "bitFile.h"  using namespace std;  //cmake\_minimum\_required(VERSION 3.23)  //project(Siaod2sem)  //  //set(CMAKE\_CXX\_STANDARD 23)  //  //add\_executable(Siaod2sem Task2/main.cpp)  int main(){  system("chcp 65001");  int task=-1;  while ( task !=0 ){  cout<<endl;  cout<<"Практическая работа 2. Смольников Алексей. ИКБО-10-21"<<endl;  cout<<"Задание 1. Разработать программу, управления текстовым файлом."<<endl;  cout<<"Задание 2. 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."<<endl;  cout<<"Выберите задание: ";  cin>>task;  switch (task) {  case 1: {  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 в конец файла новой строки"<<endl;  cout<<"4 прочитать значение числа, указав его порядковый номер в файле, и вернуть его значение"<<endl;  cout<<"5 (Вариант 22) Создать новый файл из значений исходного, размещая на каждой строке заданное количество чисел, кроме, возможно, последней, где должны разместиться оставшиеся числа."<<endl;  cout<<"Выберите функцию: ";  int func;  cin>>func;  switch (func) {  case 1:{  cout<<"Введите имя файла: ";  string fileName;  cin>>fileName;  generateFile(fileName);  break;  }  case 2:{  cout<<"Введите имя файла: ";  string fileName;  cin>>fileName;  printFile(fileName);  break;  }  case 3:{  cout<<"Введите имя файла: ";  string fileName;  cin>>fileName;  cout<<"Введите строку: ";  string str;  cin>>str;  appendLine(fileName, str);  break;  }  case 4:{  cout<<"Введите имя файла: ";  string fileName;  cin>>fileName;  cout<<"Введите номер числа: ";  int num;  cin>>num;  cout<<"Число: "<<readNumber(fileName, num)<<endl;  break;  }  case 5:{  cout<<"Введите имя файла: ";  string fileName;  cin>>fileName;  cout<<"Введите имя нового файла: ";  string newFileName;  cin>>newFileName;  cout<<"Введите количество чисел в строке: ";  int count;  cin>>count;  copyFormatted(fileName, newFileName, count);  break;  }  default:{  cout<<"Неверный ввод"<<endl;  break;  }  }  break;  }  case 2: {  cout<<"Расписание занятий группы: номер группы, название дисциплины, номер пары, номер недели, номер дня недели, вид занятия, номер аудитории."<<endl;  cout<<"1 преобразование тестовых данных из текстового файла в двоичный файл;"<<endl;  cout<<"2 преобразование данных из двоичного файла в текстовый;"<<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 в двоичном файле расписание заданной группы на заданный день недели."<<endl;  cout<<"7 Обновить расписание, найти дисциплины, которые стоят на одинаковых парах в одной аудитории, и определить для них новые аудитории.."<<endl;  cout<<"Выберите функцию: ";  int func;  cin>>func;  switch (func) {  case 1:{  cout<<"Введите имя файла: ";  string fileName;  cin>>fileName;  cout<<"Введите имя нового файла: ";  string newFileName;  cin>>newFileName;  convertToBitFile(fileName, newFileName);  break;  }  case 2:{  cout<<"Введите имя файла: ";  string fileName;  cin>>fileName;  cout<<"Введите имя нового файла: ";  string newFileName;  cin>>newFileName;  convertToTextFile(fileName, newFileName);  break;  }  case 3:{  cout<<"Введите имя файла: ";  string fileName;  cin>>fileName;  printBitFile(fileName);  break;  }  case 4:{  cout<<"Введите имя файла: ";  string fileName;  cin>>fileName;  cout<<"Введите номер записи: ";  int num;  cin>>num;  directAccess(fileName, num);  break;  }  case 5:{  cout<<"Введите имя файла: ";  string fileName;  cin>>fileName;  cout<<"Введите ключ: ";  string key;  cin>>key;  deleteByKey(fileName, key);  break;  }  case 6:{  cout<<"Введите имя файла: ";  string fileName;  cin>>fileName;  cout<<"Введите номер группы: ";  string group;  cin>>group;  cout<<"Введите номер дня недели: ";  string day;  cin>>day;  generateByWeekDay(fileName, group, day);  break;  }  case 7:{  cout<<"Введите имя файла: ";  string fileName;  cin>>fileName;  cout<<"Введите имя нового файла: ";  string newFileName;  cin>>newFileName;  updateSchedule(fileName, newFileName);  break;  }  default:{  cout<<"Неверный ввод"<<endl;  break;  }  }  }  }  }  return 0;  } |

textFile.h

|  |
| --- |
| //  // Created by lexa2k on 16.09.2022.  //  #ifndef SIAOD2SEM\_TEXTFILE\_H  #define SIAOD2SEM\_TEXTFILE\_H  #include <string>  #include <fstream>  bool generateFile(std::string name);  bool printFile(std::string name);  bool appendLine(std::string name, std::string line = "");  int readNumber(std::string name, int pos);  int countNumbers(std::string name);  int copyFormatted(std::string name, std::string cp\_name , int width);  #endif //SIAOD2SEM\_TEXTFILE\_H |

bitFile.h

|  |
| --- |
| //  // Created by lexa2k on 17.09.2022.  //  #ifndef SIAOD2SEM\_BITFILE\_H  #define SIAOD2SEM\_BITFILE\_H  #include <string>  //  //struct Record {  // std::string group;  // std::string disciplineName;  // std::string pairNumber;  // std::string weekNumber;  // std::string dayNumber;  // std::string lessonType;  // std::string roomNumber;  //  // void cutRecord();  // void printRecord();  // std::string createNote();  //};  struct customVector{  int size;  std::string\* data;  int capacity;  customVector();  void push\_back(std::string s);  void print();  ~customVector();  };  bool convertToBitFile(std::string fileName, std::string bitFileName);  bool convertToBitFile(std::string fileName, std::string bitFileName);  bool convertToTextFile(std::string bitFileName, std::string fileName);  bool printBitFile(std::string bitFileName);  bool directAccess(std::string bitFileName, int number, int length=32);  bool deleteByKey(std::string bitFileName, std::string key, int length=32);  void generateByWeekDay(std::string bitFileName, std::string group, std::string day);  std::string\* splitter (std::string s, std::string delim, int size);  //Обновить расписание, найти дисциплины, которые стоят на одинаковых парах в одной аудитории, и определить для них новые аудитории.  void updateSchedule(std::string bitFileName, std::string newFileName);  #endif //SIAOD2SEM\_BITFILE\_H |

textFile.cpp

|  |
| --- |
| // Created by lexa2k on 16.09.2022.  //  //bool generateFile(std::string name);  //bool printFile(std::string name);  //bool appendLine(std::string name, std::string line = "");  //bool readNumber(std::string name, int pos);  //bool countNumbers(std::string name);  #include <string>  #include <iostream>  #include <fstream>  #include <random>  #include <sstream>  //генерирует файл  bool generateFile(std::string name){  std::ofstream file(name, std::ios::out);  if (file.is\_open()){  std::random\_device rd;  std::mt19937 gen(rd());  std::uniform\_int\_distribution<> dis(10, 10);  int n = dis(gen);  for(int i=0;i<n;i++){  for(int j=0;j<dis(gen);j++){  file << dis(gen) << " ";  }  file << std::endl;  }  file.close();  return true;  }  return false;  }  bool printFile(std::string name){  std::ifstream file(name, std::ios::in);  if (file.is\_open()){  std::string line;  while (getline(file, line)){  std::cout << line << std::endl;  }  file.close();  return true;  }  return false;  }  bool appendLine(std::string name, std::string line){  std::ofstream file(name, std::ios::app);  if (file.is\_open()){  file << line << std::endl;  file.close();  return true;  }  return false;  }  int readNumber(std::string name, int pos){  std::ifstream file(name, std::ios::in);  if (file.is\_open()){  std::string line;  int i = 0;  while (getline(file, line)){  std::stringstream ss(line);  int n;  while (ss >> n){  if (i == pos){  file.close();  return n;  }  i++;  }  }  file.close();  return NAN;  }  return NAN;  }  int countNumbers(std::string name){  std::ifstream file(name, std::ios::in);  if (file.is\_open()){  std::string line;  int i = 0;  while (getline(file, line)){  std::stringstream ss(line);  int n;  while (ss >> n){  i++;  }  }  file.close();  return i;  }  return NAN;  }  int copyFormatted(std::string name, std::string cp\_name , int width){  std::ifstream file(name, std::ios::in);  if (file.is\_open()){  std::ofstream cp\_file(cp\_name, std::ios::out);  if (cp\_file.is\_open()){  std::string line;  int cnt = 0;  while (getline(file, line)){  std::stringstream ss(line);  int n;  while (ss >> n){  if(cnt == width){  cp\_file << std::endl;  cnt = 0;  }  cnt++;  cp\_file << n << " ";  }  }  cp\_file.close();  file.close();  return true;  }  file.close();  return false;  }  return false;  } |

bitFile.cpp

|  |
| --- |
| //  // Created by lexa2k on 17.09.2022.  //Структура записи  //Расписание занятий группы: номер группы, название дисциплины, номер пары, номер недели, номер дня недели, вид занятия, номер аудитории.  //Доп. операция  //Сформировать в двоичном файле расписание заданной группы на заданный день недели.  //Обновить расписание, найти дисциплины, которые стоят на одинаковых парах в одной аудитории, и определить для них новые аудитории.  //001 ИКБО-10-21 Пр1 1 1 7 сем 223  //002 ИКБО-01-21 Пр2 2 1 7 сем 223  //003 ИКБО-02-21 Пр3 2 1 7 сем 220  //004 ИКБО-03-21 пр4 2 1 4 сем 220  //005 ИКБО-03-21 пр4 2 1 4 сем 220  #include "bitFile.h"  #include <fstream>  #include <iostream>  #include <string>  #include <iomanip>  #include <map>  //struct Record {  // std::string hash;  // std::string group;  // std::string disciplineName;  // std::string pairNumber;  // std::string weekNumber;  // std::string dayNumber;  // std::string lessonType;  // std::string roomNumber;  //  // void cutRecord(){  // this->hash = this->hash.substr(0);  // this->group = this->group.substr(0, this->group.find(','));  // this->disciplineName = this->disciplineName.substr(0, this->disciplineName.find(','));  // this->pairNumber = this->pairNumber.substr(0, this->pairNumber.find(','));  // this->weekNumber = this->weekNumber.substr(0, this->weekNumber.find(','));  // this->dayNumber = this->dayNumber.substr(0, this->dayNumber.find(','));  // this->lessonType = this->lessonType.substr(0, this->lessonType.find(','));  // this->roomNumber = this->roomNumber.substr(0, this->roomNumber.find(','));  // }  //  // void printRecord(){  // std::cout << this->group << " " << this->disciplineName << " " << this->pairNumber << " " << this->weekNumber << " " << this->dayNumber << " " << this->lessonType << " " << this->roomNumber << std::endl;  // }  //  // std::string createNote(){  // std::string note = this->group + " " + this->disciplineName + " " + this->pairNumber + " " + this->weekNumber + " " + this->dayNumber + " " + this->lessonType + " " + this->roomNumber + "\n";  // return note;  // }  //};  bool convertToBitFile(std::string fileName, std::string bitFileName){  std::ifstream file(fileName, std::ios::in);  if (!file.is\_open()){  throw std::runtime\_error("File not found");  }  std::ofstream bitFile(bitFileName, std::ios::binary | std::ios::out);  if (!bitFile.is\_open()){  throw std::runtime\_error("File not found");  }  bitFile << file.rdbuf();  bitFile.close();  file.close();  return true;  }  bool convertToTextFile(std::string bitFileName, std::string fileName){  std::ifstream bitFile(bitFileName, std::ios::binary | std::ios::in);  if (!bitFile.is\_open()){  throw std::runtime\_error("File not found");  }  std::ofstream file(fileName, std::ios::out);  if (!file.is\_open()){  throw std::runtime\_error("File not found");  }  file<<bitFile.rdbuf();  bitFile.close();  file.close();  return true;  }  bool printBitFile(std::string bitFileName){  std::ifstream bitFile(bitFileName, std::ios::binary | std::ios::in);  if (!bitFile.is\_open()){  throw std::runtime\_error("File not found");  }  std::cout<<"номер группы | название дисциплины | номер пары | номер недели | номер дня недели | вид занятия | номер аудитории"<<std::endl;  std::cout<<bitFile.rdbuf();  bitFile.close();  return true;  }  bool directAccess(std::string bitFileName, int number, int length){  std::ifstream bitFile(bitFileName, std::ios::binary | std::ios::in);  if (!bitFile.is\_open()){  throw std::runtime\_error("File not found");  }  bitFile.seekg(0, std::ios::end);  int size = bitFile.tellg();  if ((length\*sizeof(char)+10)\*number > size){  bitFile.close();  return false;  }  bitFile.seekg((length\*sizeof(char)+10)\*number);  std::string s;  std::getline(bitFile, s);  std::cout<<s<<std::endl;  bitFile.close();  return true;  }  bool deleteByKey(std::string bitFileName, std::string key, int length){  std::ifstream bitFile(bitFileName, std::ios::binary | std::ios::in);  if (!bitFile.is\_open()){  throw std::runtime\_error("File not found");  }  std::ofstream tempFile("temp.bin", std::ios::out | std::ios::binary);  if (!tempFile.is\_open()){  bitFile.close();  throw std::runtime\_error("File not found");  }  bitFile.seekg(0, std::ios::end);  int size = bitFile.tellg();  bitFile.seekg(0, std::ios::beg);  bitFile.seekg(size - length-9);  std::string temp;  std::getline(bitFile, temp);  std::string tempNum = temp.substr(0, temp.find(' '));  temp = key+ temp.substr(key.size(), temp.size());  bitFile.seekg(0, std::ios::beg);  std::string s;  while (!bitFile.eof()){  std::getline(bitFile, s);  if(s.substr(0, key.size()) != key && s.substr(0, key.size()) != tempNum){  tempFile<<s<<std::endl;  } else if (s.substr(0, key.size()) == tempNum){  }else{  tempFile << temp << std::endl;  }  }  bitFile.close();  tempFile.close();  std::remove(bitFileName.c\_str());  if(std::rename("temp.bin", bitFileName.c\_str())==0){  return true;  } else{  return false;  }  }  std::string\* splitter (std::string s, std::string delim = " ", int size =8){  std::string\* result = new std::string[size];  int i = 0;  while (s.find(delim) != std::string::npos){  result[i] = s.substr(0, s.find(delim));  s = s.substr(s.find(delim)+1, s.size());  i++;  }  result[i] = s;  return result;  }  void generateByWeekDay(std::string bitFileName, std::string group, std::string day){  std::ifstream bitFile(bitFileName, std::ios::binary | std::ios::in);  if (!bitFile.is\_open()){  throw std::runtime\_error("File not found");  }  std::ofstream file("generated.bin", std::ios::out | std::ios::binary);  std::string s;  while (!bitFile.eof()){  std::getline(bitFile, s);  std::string\* temp = splitter(s);  if (temp[1] == group && temp[5] == day){  file<<s<<std::endl;  }  }  bitFile.close();  file.close();  }  customVector::customVector() {  data = new std::string[1];  size = 0;  capacity = 1;  }  void customVector::push\_back(std::string s){  if (size == capacity){  std::string\* temp = new std::string[capacity\*2];  for (int i = 0; i < size; i++){  temp[i] = data[i];  }  delete[] data;  data = temp;  capacity \*= 2;  }  data[size] = s;  size++;  }  void customVector::print(){  for (int i = 0; i < size; i++){  std::cout<<data[i]<<std::endl;  }  }  customVector::~customVector(){  delete[] data;  }  void updateSchedule(std::string bitFileName, std::string newFileName){  std::ifstream bitFile(bitFileName, std::ios::binary | std::ios::in);  if (!bitFile.is\_open()){  throw std::runtime\_error("File not found");  }  std::ofstream tempFile(newFileName, std::ios::out | std::ios::binary);  if (!tempFile.is\_open()){  bitFile.close();  throw std::runtime\_error("File not found");  }  customVector scheudele;  std::string s;  while (!bitFile.eof()){  std::getline(bitFile, s);  std::string\* temp = splitter(s);  // s = temp[0] + " " + temp[3] + " " + temp[4] + " " + temp[5] + " " + temp[7];  scheudele.push\_back(s);  }  scheudele.print();  for(int i=0;i<scheudele.size;i++){  std::string\* temp = splitter(scheudele.data[i]);  // for(int z=0;z<8;z++){  // tempFile<<"!"<<temp[z];  // }  // tempFile<<std::endl;  for(int j=i+1;j<scheudele.size;j++){  std::string\* temp2 = splitter(scheudele.data[j]);  if(temp[7]==temp2[7] && temp[3]==temp2[3] && temp[4]==temp2[4] && temp[5]==temp2[5]){  scheudele.data[i] = temp[0]+ " " +temp[1] + " " + temp[2] + " " + temp[3] + " " + temp[4] + " " + temp[5] + " " + temp[6] + " " + std::to\_string(200+ rand()%100);  }  }  }  for(int i=0;i<scheudele.size;i++){  tempFile<<scheudele.data[i]<<std::endl;  }  bitFile.close();  tempFile.close();  } |