|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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 № 5.2** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Алгоритмы поиска в таблице (массиве). Применение алгоритмов поиска к поиску по ключу записей в файле.»** | |
|  | |
| Выполнил студент группы ИНБО-12-23 | Албахтин И.В. |
| Принял ассистент | Муравьёва Е.А. |

Москва 2024

**Оглавление**

[1. Цель работы 3](#_Toc177462904)

[2. Задание №1 3](#_Toc177462905)

[2.1 Постановка задачи 3](#_Toc177462906)

[2.2 Код программы 3](#_Toc177462907)

[2.3 Тестирование 5](#_Toc177462908)

[3. Задание №2 5](#_Toc177462909)

[3.1 Постановка задачи 5](#_Toc177462910)

[3.2 Код программы 5](#_Toc177462912)

[3.3 Тестирование 7](#_Toc177462913)

[3.4 Таблица результатов 7](#_Toc177462914)

[4. Задание №3 7](#_Toc177462915)

[4.1 Постановка задачи 7](#_Toc177462916)

[4.2 Код программы 8](#_Toc177462917)

[4.3 Тестирование 11](#_Toc177462918)

[4.4 Таблица результатов 11](#_Toc177462919)

[5. Вывод 11](#_Toc177462920)

# **Цель работы**

Получить практический опыт по применению алгоритмов поиска в таблицах данных.

# **Задание №1**

## **2.1 Постановка задачи**

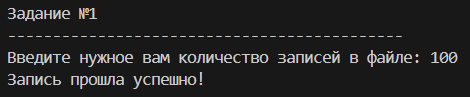
Создать двоичный файл из записей (структура записи определена вариантом – смотрите в конце файла). Поле ключа записи в задании варианта подчеркнуто. Заполнить файл данными, используя для поля ключа датчик случайных чисел. Ключи записей в файле уникальны.

## **2.2 Код программы**

*Листинг 1 – Исходный код программы для Задания 1*

|  |
| --- |
| #include <iostream>  #include <string>  #include <fstream>  #include <vector>  #include <chrono>  #include <algorithm>  using namespace std;  bool checkNum (string n){  for (int i = 0; i < n.size(); ++i) {  if (!(n[i]>='0' and n[i]<='9'))  return 0;  }  return 1;  }  int CheckInput (string n){  while (!(checkNum(n))) {  cout << "Введено некорректное значение, попробуйте еще раз: ";  cin >> n;  }  int N = stoi(n);  return N;  }  string CheckInput1 (string n){  while (!(checkNum(n) | n[0] == '8' | n[1] == 9 | size(n) == 11)) {  cout << "Введено некорректное значение, попробуйте еще раз: ";  cin >> n;  }  return n;  }  void inputToFile(string inputFileName, int count){  string checkCount;  string number;  string streat;  int indexOfStreat;  int indexOfNumber;  vector<string> streats = {"улица Бабаевская ", "Бабьегородский переулок", "улица Багратион", "Багратионовский проезд", "улица Багрицкого", "улица Баженова", "улица Бажова", "улица Базовая", "улица Базовская", "улица Байдукова", "улица Байкальская", "улица Балчуг"};    vector<string> numbers;  for (int i=0; i<count; i++){  numbers.push\_back(to\_string(i+100000000));  }  auto iter = numbers.cbegin(); // указатель на первый элемент  ofstream out;  int tmpCount = count;  out.open(inputFileName);  for (size\_t i = 0; i < count; i++)  {  indexOfNumber = (rand() % (tmpCount));  number = numbers[indexOfNumber];  numbers.erase(iter+indexOfNumber);  tmpCount--;  indexOfStreat = (rand() % (size(streats)));  streat = streats[indexOfStreat];  out << "89" << number << " " << streat << endl;  }  out.close();  cout << "Запись прошла успешно!" << endl;  }  int main() {  setlocale(LC\_ALL, "RUS");  srand(time(0));  string inputFileName = "PhoneNumbers.bin";  ofstream fout;  string checkCount;  if (!fout) {  cerr << "Ошибка открытия файла для записи!" << endl;  return -1;  }  cout << endl << "Задание №1" << endl << "--------------------------------------------" << endl;  cout << "Введите нужное вам количество записей в файле: ";  cin >> checkCount;  int count = CheckInput(checkCount);  inputToFile(inputFileName, count);  } |

## **2.3 Тестирование**



**Рисунок 1. Результаты тестирования программы №1**

# **Задание №2**

## **3.1 Постановка задачи**

## Поиск в файле с применением линейного поиска:

## 1. Разработать программу поиска записи по ключу в бинарном файле, созданном в первом задании, с применением алгоритма линейного поиска.

## 2. Провести практическую оценку времени выполнения поиска на файле объемом 100, 1000, 10 000 записей.

## 3. Составить таблицу с указанием результатов замера времени.

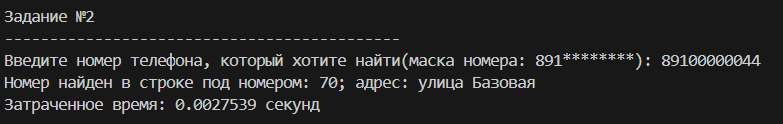
## **3.2 Код программы**

*Листинг 2 – Исходный код программы для Задания 2*

|  |
| --- |
| #include <iostream>  #include <string>  #include <fstream>  #include <vector>  #include <chrono>  #include <algorithm>  using namespace std;  bool checkNum (string n){  for (int i = 0; i < n.size(); ++i) {  if (!(n[i]>='0' and n[i]<='9'))  return 0;  }  return 1;  }  string CheckInput1 (string n){  while (!(checkNum(n) | n[0] == '8' | n[1] == 9 | size(n) == 11)) {  cout << "Введено некорректное значение, попробуйте еще раз: ";  cin >> n;  }  return n;  }  string findNumber(string name, string num) {  int lineNumber = 1;  string tmp = "89";  string line;  string adres;  ifstream fout;  fout.open(name);  if (fout.is\_open()){  cout << "Номер найден в строках под номером: ";  while(getline(fout, line)) {  for (int i = 2; i < 11; i++){  if (num[i] == line[i]) tmp +=num[i];  else{ break; }  }  if (num == tmp){  adres = line.substr(12, size(line));  cout << lineNumber << "; адрес: " << adres;  return adres;  }  tmp = "89";  lineNumber++;  }  cout << endl << endl;  fout.close();  }  else {  cerr << "Не удалось открыть файл " << name << endl;  }  }  int main() {  setlocale(LC\_ALL, "RUS");  string inputFileName = "PhoneNumbers.bin";  string phone;  string checkPhone;  cout << endl << "Задание №2" << endl << "--------------------------------------------" << endl;  cout << "Введите номер телефона, который хотите найти(маска номера: 89\*\*\*\*\*\*\*\*\*): ";  cin >> checkPhone;  phone = CheckInput1(checkPhone);  auto start = chrono::high\_resolution\_clock::now();  string adres = findNumber(inputFileName, phone);  auto end = chrono::high\_resolution\_clock::now();  chrono::duration<double> duration = end - start;  cout << endl << "Затраченное время: " << duration.count() << " секунд" << endl;  } |

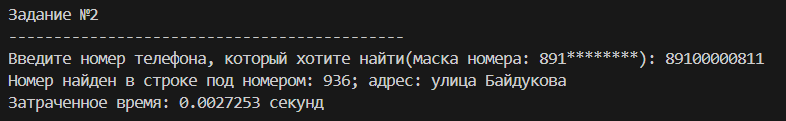
## **3.3 Тестирование программы**

### **3.3.1 Тестирование на 100 записях**



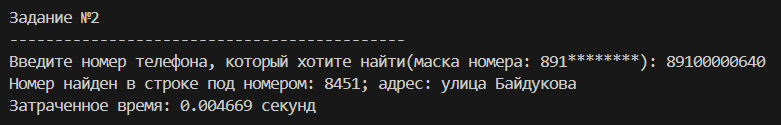
**Рисунок 2. Результаты тестирования программы №2 на 100 записях**

### **3.3.2 Тестирование на 1000 записях**



**Рисунок 3. Результаты тестирования программы №2 на 1000 записях**

### **3.3.3 Тестирование на 10 000 записях**



**Рисунок 4. Результаты тестирования программы №2 на 10 000 записях**

## **3.4 Таблица результатов**

|  |  |
| --- | --- |
| **Количество записей, шт** | **Время поиска записей, мс** |
| 100 | 2.7539 |
| 1000 | 2.7253 |
| 10000 | 4.669 |

# **Задание №3**

## **4.1 Постановка задачи**

Поиск записи в файле с применением дополнительной структуры данных, сформированной в оперативной памяти.

1. Для оптимизации поиска в файле создать в оперативной памяти структур данных – таблицу, содержащую ключ и ссылку (смещение) на запись в файле.

2. Разработать функцию, которая принимает на вход ключ и ищет в таблице элемент, содержащий ключ поиска, а возвращает ссылку на запись в файле. Алгоритм поиска определен в варианте.

3. Разработать функцию, которая принимает ссылку на запись в файле, считывает ее, применяя механизм прямого доступа к записям файла. Возвращает прочитанную запись как результат.

4. Провести практическую оценку времени выполнения поиска на файле объемом 100, 1000, 10 000 записей.

5. Составить таблицу с указанием результатов замера времени.

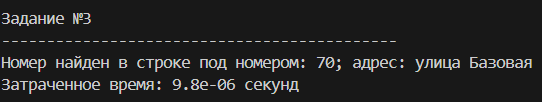
## **4.2 Код программы**

*Листинг 3 – Исходный код программы для Задания 3*

|  |
| --- |
| #include <iostream>  #include <string>  #include <fstream>  #include <vector>  #include <chrono>  #include <algorithm>  #include <cmath>  using namespace std;  struct stroki {  private:      string phone\_;              // номер      string adress\_;             // адрес      int final\_position;         // позиция курсора в файле (строка)  public:      stroki() : phone\_(""), adress\_(""), final\_position(0) {}      stroki(string phone, string adress, int this\_) : phone\_(phone), adress\_(adress), final\_position(this\_) {}      // геттеры с константными квалификаторами      string get\_phone() const { return phone\_; }      string get\_adress() const { return adress\_; }      int get\_final\_position() const { return final\_position; }      void print() const {          cout << "Phone: " << phone\_ << ", Address: " << adress\_ << ", Position: " << final\_position << endl;      }  };  bool checkNum (string n){      for (int i = 0; i < n.size(); ++i) {          if (!(n[i]>='0' and n[i]<='9'))              return 0;      }      return 1;  }  int CheckInput (string n){      while (!(checkNum(n))) {          cout << "Введено некорректное значение, попробуйте еще раз: ";          cin >> n;      }      int N = stoi(n);      return N;  }  string CheckInput1 (string n){      while (!(checkNum(n) | n[0] == '8' | n[1] == '9' | n[1] == '1' | size(n) == 11)) {          cout << "Введено некорректное значение, попробуйте еще раз: ";          cin >> n;      }      return n;  }  void inputToFile(string inputFileName, int count){      string checkCount;      string number;      string streat;      int indexOfStreat;      int indexOfNumber;      vector<string> streats = {"улица Бабаевская ", "Бабьегородский переулок", "улица Багратион", "Багратионовский проезд", "улица Багрицкого", "улица Баженова", "улица Бажова", "улица Базовая", "улица Базовская", "улица Байдукова", "улица Байкальская", "улица Балчуг"};        vector<string> numbers;      for (int i=0; i<count; i++){          numbers.push\_back(to\_string(i+100000000));      }      auto iter = numbers.cbegin(); // указатель на первый элемент      ofstream out;      int tmpCount = count;      out.open(inputFileName);      for (size\_t i = 0; i < count; i++)      {          indexOfNumber = (rand() % (tmpCount));          number = numbers[indexOfNumber];          numbers.erase(iter+indexOfNumber);          tmpCount--;          indexOfStreat = (rand() % (size(streats)));          streat = streats[indexOfStreat];          out << "89" << number << " " << streat << endl;      }      out.close();      cout << "Запись прошла успешно!" << endl;  }  void findNumber(string fileName, string key, int n) {      int lineNumber = 1;      string tmp = "891";      string line;      ifstream fout;      string adres;      fout.open(fileName);      if (fout.is\_open()){          while(getline(fout, line)) {              for (int i = 3; i < 11; i++){                  if (key[i] == line[i]) tmp +=key[i];                  else{ break; }              }              if (key == tmp){                  adres = line.substr(12, size(line));                  cout << "Номер найден в строке под номером: " << lineNumber <<  "; адрес: " << adres;              }              tmp = "891";              lineNumber++;          }      fout.close();      }      else {          cerr << "Не удалось открыть файл " << fileName << endl;      }  }  void get\_stroki(int count, vector<stroki>& Stroki) {      ifstream File("PhoneNumbers.bin", ios::binary);      int cnt\_search = 1;      string adress\_ = "";      if (File.is\_open()) {          string get\_;          while (getline(File, get\_)) {              string tmp = "891";              string phone\_ = get\_.substr(3, 8); //забираем номер телефона              adress\_ = get\_.substr(12); // получаем адрес (остальную часть строки)              if (cnt\_search < count) {                  stroki obj(tmp + phone\_, adress\_, cnt\_search); // создаем объект                  Stroki[cnt\_search - 1] = obj; // кладем объект в вектор (учтите, что индексация начинается с 0)              } else {                  break;              }              cnt\_search++;          }      }      File.close();  }  int binary\_search\_with\_delta(const vector<stroki>& arr, string target) {      int n = arr.size();      // Вычисление таблицы смещений (можно оптимизировать)      vector<int> delta(n);      for (int i = 1; i < n; ++i) {          delta[i] = (n + static\_cast<int>(pow(2, i - 1))) / static\_cast<int>(pow(2, i));      }      int i = delta[1];      int j = 2;      while (true) {          if (target < arr[i].get\_phone()) { // Искомое меньше              if (delta[j] == 0) {                  return -1; // Элемент не найден              }              i = i - delta[j];              j++;          } else if (target > arr[i].get\_phone()) { // Искомое больше              if (delta[j] == 0) {                  return -1; // Элемент не найден              }              i = i + delta[j];              j++;          } else {              return i; // Элемент найден          }      }  }  int main() {      setlocale(LC\_ALL, "RUS");      srand(time(0));      string inputFileName = "PhoneNumbers.bin";      string checkPhone;      ofstream fout;      string checkCount;      if (!fout) {          cerr << "Ошибка открытия файла для записи!" << endl;          return -1;      }      cout << endl << "Задание №1" << endl << "--------------------------------------------" << endl;      cout << "Введите нужное вам количество записей в файле: ";      cin >> checkCount;      int count = CheckInput(checkCount);      inputToFile(inputFileName, count);        cout << endl << "Задание №2" << endl << "--------------------------------------------" << endl;      string phone;      cout << "Введите номер телефона, который хотите найти(маска номера: 891\*\*\*\*\*\*\*\*): ";      cin >> checkPhone;      phone = CheckInput1(checkPhone);      auto start = chrono::high\_resolution\_clock::now();      findNumber(inputFileName, phone, count);      auto end = chrono::high\_resolution\_clock::now();      chrono::duration<double> duration = end - start;      cout << endl << "Затраченное время: " << duration.count() << " секунд" << endl;        cout << endl << "Задание №3" << endl << "--------------------------------------------" << endl;      vector<stroki> Stroki(count, stroki{"", "", 0}); // Инициализация с пустыми значениями      get\_stroki(count, Stroki);      sort(Stroki.begin(), Stroki.end(), [](const stroki& a, const stroki& b) {          return a.get\_phone() < b.get\_phone();      });      auto start1 = chrono::high\_resolution\_clock::now();      // Выполняем бинарный поиск      int result = binary\_search\_with\_delta(Stroki, phone);      auto end1 = chrono::high\_resolution\_clock::now();      if (result != -1 && result < Stroki.size()) {          // Если номер найден и индекс корректен, выводим его данные          cout << "Номер найден в строке под номером: " << Stroki[result].get\_final\_position() << "; адрес: " << Stroki[result].get\_adress() << endl;      } else {          // Если номер не найден или индекс неверный          cout << "Номер телефона не найден" << endl;      }      chrono::duration<double> duratio = end1 - start1;      cout << "Затраченное время: " << duratio.count() << " секунд" << endl;      main();  } |

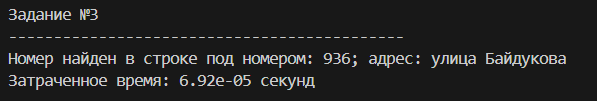
## **4.3 Тестирование программы**

### **4.3.1 Тестирование на 100 записях**



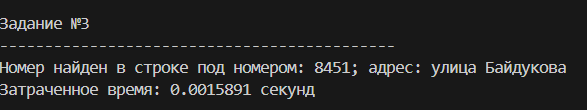
**Рисунок 5. Результаты тестирования программы №3 на 100 записях**

### **4.3.2 Тестирование на 1000 записях**



**Рисунок 6. Результаты тестирования программы №3 на 1000 записях**

### **4.3.3 Тестирование на 10 000 записях**



**Рисунок 7. Результаты тестирования программы №3 на 10 000 записях**

## **4.4 Таблица результатов**

|  |  |
| --- | --- |
| **Количество записей, шт** | **Время поиска записей, мс** |
| 100 | 0.0080 |
| 1000 | 0.00092 |
| 10000 | 1.5891 |

# **5. Вывод**

В результате выполнения работы я освоил приёмы создания программ для поиска записей с заданным ключом в двоичном файле с применением различных алгоритмов, поиска в файле с применением линейного поиска, а также поиска записи в файле с применением дополнительной структуры данных, сформированной в оперативной памяти.