|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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 № 9** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Поиск записей в файле»** | |
|  | |
| Выполнил студент группы ИКБО-09-22 | Гришин А.В. |
| Принял преподаватель | Филатов А.С. |

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

Москва 2023

# **Цель работы**

Получить практический опыт по применению алгоритмов поиска в таблицах данных.

# **Постановка задачи**

1. Разработать программу генерации двоичного файла из записей (структура записи определена вариантом). Поле ключа записи в задании варианта подчеркнуто. Файл заполнять данными, используя для датчик псевдослучайных чисел. Ключи записей в файле уникальны.
2. Разработать программу поиска записи по ключу в бинарном файле с применением алгоритма линейного поиска.
3. Для оптимизации поиска в файле, разработать программу, создающую в оперативной памяти дополнительную структуру данных, содержащую ключ и ссылку (смещение) на запись в файле.
4. Провести анализ эффективности разработанного алгоритма поиска по сравнению с линейным.

Вариант №7. Поиск записей в файле.

|  |  |
| --- | --- |
| Бинарный поиск | Специализация вуза: код специальности – целое число, название вуза, год начала обучения |

# **Решение**

Выполняя первое упражнение, мы должны написать структуру, которая будет хранить в себе данные.

|  |
| --- |
| struct record {  int key;  char name[20];  int year;  }; |

После этого мы должны открыть файл и записать в него данные, все будет заполнять рандомно и в хаотичном порядке, при этом будем учитывать, что ключ всегда уникален.

|  |
| --- |
| char alphavite[27] = "abcdefghijklmnopqrstuvwxyz";  record\* mas = new record[N];  vector<int> keys(N);  record last;  int bytes = 0;  ofstream file("data.bin");  for (int i = 0; i < N; i++)  keys[i] = i + 1;  random\_device rd;  mt19937 g(rd());  shuffle(keys.begin(), keys.end(), g);  for (int i = 0; i < N; i++)  {  char temp1[20];  char temp2[20];  mas[i].key = keys[i];  for (int j = 0; j < 20; j++)  {  temp1[j] = alphavite[rand() % 26];  temp2[j] = alphavite[rand() % 26];  }  temp1[19] = '\0';  temp2[19] = '\0';  strcpy(mas[i].name, temp1);  mas[i].year = rand() % (2023 - 1980 + 1) + 1980;  last.key = mas[i].key;  strcpy(last.name, mas[i].name);  last.year = mas[i].year;  bytes += sizeof(mas[i]);  }  file.write((char\*)mas, bytes);  file.close(); |

Уникальности значений мы добиваемся благодаря методу shuffle, который перемешает значения в полностью заполненном векторе уникальными значениями.

|  |
| --- |
| while (key != 0) {  ifstream infile("data.bin", ios::binary);  if (!infile)  {  cout << "Ошибка открытия файла!" << endl;  return 1;  }  record temp;  bool found = false;  clock\_t start = clock();  while (infile.read((char\*)&temp, sizeof(temp)))  {  if (temp.key == key)  {  cout << "\nНайдена запись: key=" << temp.key << ", name=" << temp.name << ", value=" << temp.year;  found = true;  break;  }  }  clock\_t end = clock();  if (!found)  cout << "Запись с таким ключом не найдена!" << endl;  double duration\_ms = 1000.0 \* (end - start) / CLOCKS\_PER\_SEC;  cout << "\nВремя выполнения = " << duration\_ms << endl;  infile.close();  cout << "\nВведите ключ для поиска: ";  cin >> key;  } |

Для реализации второго задания мы должны написать алгоритм линейного поиска, который заключается в том, что мы проходимся по всем значениям нашего списка, до тех пор, пока не найдем наше значение.

|  |
| --- |
| struct searchrecord  {  int key;  int pos;  };  void bubbleSort(searchrecord\* table, int n)  {  for (int i = 0; i < n - 1; i++)  {  for (int j = 0; j < n - i - 1; j++)  {  if (table[j].key > table[j + 1].key)  {  searchrecord temp = table[j];  table[j] = table[j + 1];  table[j + 1] = temp;  }  }  }  }  void createSearchTable(searchrecord\*& table, int N, FILE\* fp)  {  table = new searchrecord[N];  record temp;  int i = 1;  while (i <= N)  {  size\_t r = fread(&temp, sizeof(record), 1, fp);  table[i - 1].key = temp.key;  table[i - 1].pos = i - 1;  i++;  }  bubbleSort(table, N);  }  searchrecord binarySearchInTable(searchrecord\* table, int N, int key)  {  int left = 0;  int right = N - 1;  while (left <= right)  {  int mid = left + (right - left) / 2;  if (table[mid].key == key)  return table[mid];  else if (table[mid].key < key)  left = mid + 1;  else  right = mid - 1;  }  return table[0];  }  record getRecord(FILE\* fp, int pos)  {  fseek(fp, pos \* sizeof(record), SEEK\_SET);  record temp;  fread(&temp, sizeof(record), 1, fp);  return temp;  } |

Для выполнения задания 3 мы создаем структуру, которая хранит в себе ключ и позицию. Чтобы использовать бинарный поиск, мы должны иметь отсортированную структуру данных, отсортируем наши данные и после воспользуемся алгоритмом поиска, который заключается в делении массива на два и поиска элемента до тех пор, пока не найдем его.

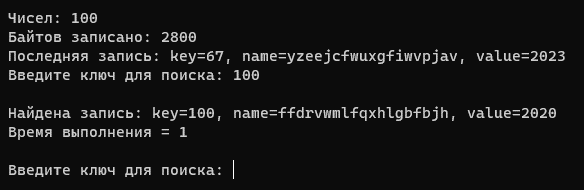


Рисунок 1. Интерфейс программы

# **Тестирование**

Протестируем выполнение программой поиска от 100 до 10000 значений и занесем их в таблицу.

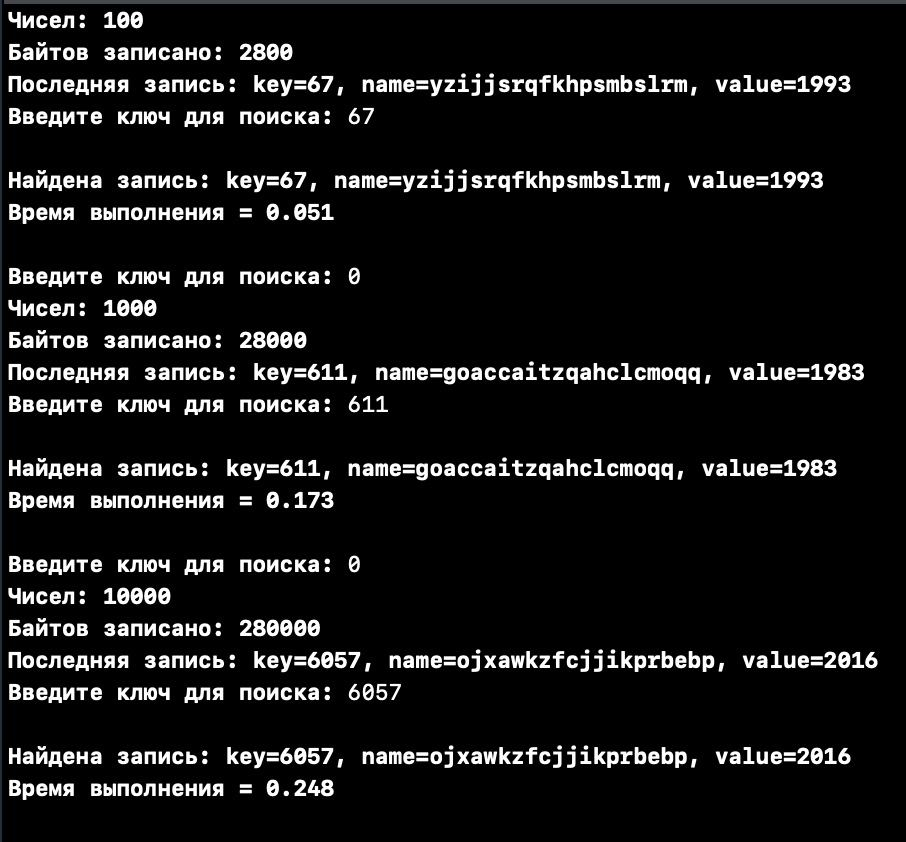


Рисунок 2. Тестирование программы для 2 задания

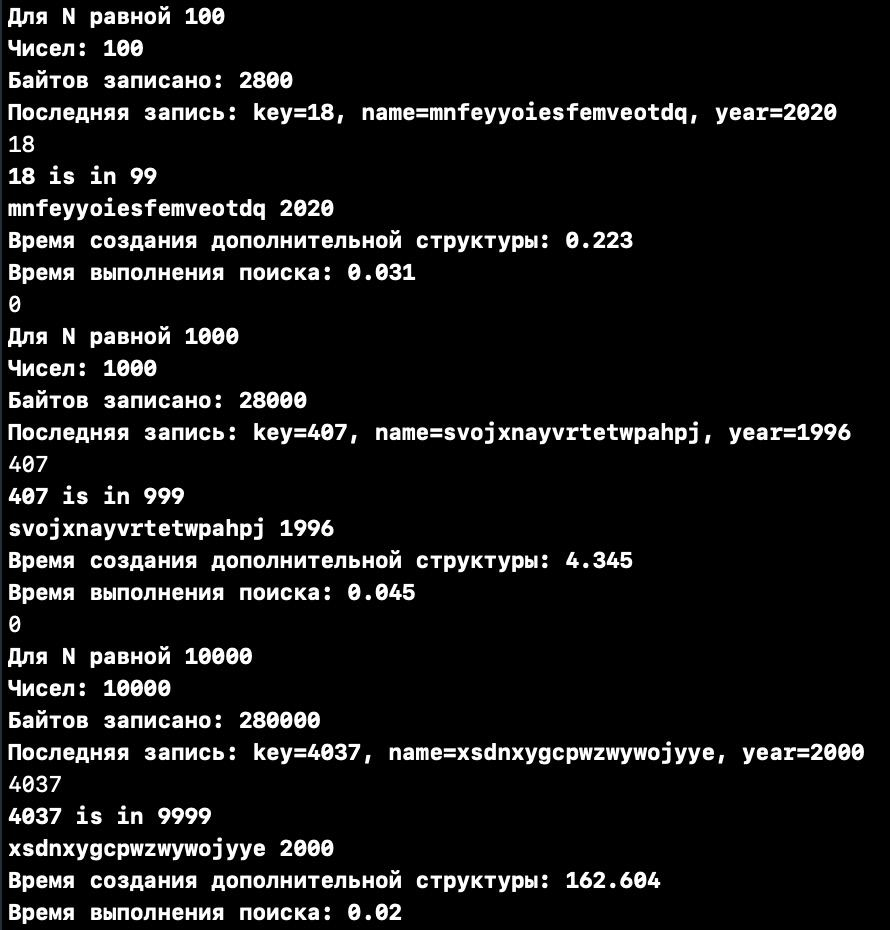


Рисунок 3. Тестирование программы для 3 задания

Программа показала корректную работу, занесем данные в таблицу.

Таблица 1. Таблица результатов для поиска последнего элемента с помощью алгоритма линейного поиска

|  |  |
| --- | --- |
| N | t, ms |
| 100 | 0.051 |
| 1000 | 0.173 |
| 10000 | 0.248 |

Таблица 2. Таблица результатов для поиска последнего элемента с помощью алгоритма бинарного поиска

|  |  |  |
| --- | --- | --- |
| N | t,ms (поиск) | t,ms (структура) |
| 100 | 0.031 | 0.223 |
| 1000 | 0.045 | 4.345 |
| 10000 | 0.02 | 162.604 |

Алгоритм бинарного поиска ищет быстрее, чем линейный поиск, по при этом, ему нужно будет изначально создать структуру, в которой он сможет быстрее находить значения. С точки зрения эффективности лучше всего использовать бинарный поиск.

# **Вывод**

В результате выполнения работы я освоил алгоритмы работы поиска.

# **Исходный код программы**

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <iostream>  #include <fstream>  #include <string>  #include <vector>  #include <random>  #include <ctime>  #include <cstdlib>  using namespace std;  const int N = 100;  struct record {  int key;  char name[20];  int year;  };  int main()  {  setlocale(LC\_ALL, "RUS");  srand(time(NULL));  char alphavite[27] = "abcdefghijklmnopqrstuvwxyz";  record\* mas = new record[N];  vector<int> keys(N);  record last;  int bytes = 0;  ofstream file("data.bin");  for (int i = 0; i < N; i++)  keys[i] = i + 1;  random\_device rd;  mt19937 g(rd());  shuffle(keys.begin(), keys.end(), g);  for (int i = 0; i < N; i++)  {  char temp1[20];  char temp2[20];  mas[i].key = keys[i];  for (int j = 0; j < 20; j++)  {  temp1[j] = alphavite[rand() % 26];  temp2[j] = alphavite[rand() % 26];  }  temp1[19] = '\0';  temp2[19] = '\0';  strcpy(mas[i].name, temp1);  mas[i].year = rand() % (2023 - 1980 + 1) + 1980;  last.key = mas[i].key;  strcpy(last.name, mas[i].name);  last.year = mas[i].year;  bytes += sizeof(mas[i]);  }  file.write((char\*)mas, bytes);  file.close();  cout << "Чисел: " << N << endl;  cout << "Байтов записано: " << bytes << endl;  cout << "Последняя запись: key=" << last.key << ", name=" << last.name << ", value=" << last.year << endl;  return 0;  } |

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <iostream>  #include <fstream>  #include <string>  #include <vector>  #include <random>  #include <chrono>  #include <ctime>  #include <cstdlib>  using namespace std;  struct record {  int key;  char name[20];  int year;  };  int main()  {  setlocale(LC\_ALL, "RUS");  srand(time(NULL));  for (int N = 100; N <= 10000; N \*= 10)  {  char alphavite[27] = "abcdefghijklmnopqrstuvwxyz";  record\* mas = new record[N];  vector<int> keys(N);  record last;  int bytes = 0;  ofstream file("data.bin", ios::binary);  for (int i = 0; i < N; i++)  keys[i] = i + 1;  random\_device rd;  mt19937 g(rd());  shuffle(keys.begin(), keys.end(), g);  for (int i = 0; i < N; i++)  {  char temp1[20];  char temp2[20];  mas[i].key = keys[i];  for (int j = 0; j < 20; j++)  {  temp1[j] = alphavite[rand() % 26];  temp2[j] = alphavite[rand() % 26];  }  temp1[19] = '\0';  temp2[19] = '\0';  strcpy(mas[i].name, temp1);  mas[i].year = rand() % (2023 - 1980 + 1) + 1980;  last.key = mas[i].key;  strcpy(last.name, mas[i].name);  last.year = mas[i].year;  bytes += sizeof(mas[i]);  }  file.write((char\*)mas, bytes);  file.close();  cout << "Чисел: " << N << endl;  cout << "Байтов записано: " << bytes << endl;  cout << "Последняя запись: key=" << last.key << ", name=" << last.name << ", value=" << last.year << endl;  int key;  cout << "Введите ключ для поиска: ";  cin >> key;  while (key != 0) {  ifstream infile("data.bin", ios::binary);  if (!infile)  {  cout << "Ошибка открытия файла!" << endl;  return 1;  }  record temp;  bool found = false;  clock\_t start = clock();  while (infile.read((char\*)&temp, sizeof(temp)))  {  if (temp.key == key)  {  cout << "\nНайдена запись: key=" << temp.key << ", name=" << temp.name << ", value=" << temp.year;  found = true;  break;  }  }  clock\_t end = clock();  if (!found)  cout << "Запись с таким ключом не найдена!" << endl;  double duration\_ms = 1000.0 \* (end - start) / CLOCKS\_PER\_SEC;  cout << "\nВремя выполнения = " << duration\_ms << endl;  infile.close();  cout << "\nВведите ключ для поиска: ";  cin >> key;  }  }  return 0;  } |

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <iostream>  #include <fstream>  #include <string>  #include <vector>  #include <algorithm>  #include <random>  #include <ctime>  #include <cstdlib>  using namespace std;  struct record {  int key;  char name[20];  int year;  };  struct searchrecord  {  int key;  int pos;  };  void bubbleSort(searchrecord\* table, int n)  {  for (int i = 0; i < n - 1; i++)  {  for (int j = 0; j < n - i - 1; j++)  {  if (table[j].key > table[j + 1].key)  {  searchrecord temp = table[j];  table[j] = table[j + 1];  table[j + 1] = temp;  }  }  }  }  void createSearchTable(searchrecord\*& table, int N, FILE\* fp)  {  table = new searchrecord[N];  record temp;  int i = 1;  while (i <= N)  {  size\_t r = fread(&temp, sizeof(record), 1, fp);  table[i - 1].key = temp.key;  table[i - 1].pos = i - 1;  i++;  }  bubbleSort(table, N);  }  searchrecord binarySearchInTable(searchrecord\* table, int N, int key)  {  int left = 0;  int right = N - 1;  while (left <= right)  {  int mid = left + (right - left) / 2;  if (table[mid].key == key)  return table[mid];  else if (table[mid].key < key)  left = mid + 1;  else  right = mid - 1;  }  return table[0];  }  record getRecord(FILE\* fp, int pos)  {  fseek(fp, pos \* sizeof(record), SEEK\_SET);  record temp;  fread(&temp, sizeof(record), 1, fp);  return temp;  }  int main()  {  setlocale(LC\_ALL, "RUS");  srand(time(NULL));  for (int N = 100; N <= 10000; N \*= 10)  {  cout << "Для N равной " << N << endl;  char alphabet[27] = "abcdefghijklmnopqrstuvwxyz";  record\* mas = new record[N];  vector<int> keys(N);  record last;  int bytes = 0;  ofstream file("data.bin");  for (int i = 1; i < N; i++)  keys[i] = i;  random\_device rd;  mt19937 g(rd());  shuffle(keys.begin(), keys.end(), g);  for (int i = 0; i < N; i++)  {  char temp1[20];  char temp2[20];  mas[i].key = keys[i];  for (int j = 0; j < 19; j++)  {  temp1[j] = alphabet[rand() % 26];  temp2[j] = alphabet[rand() % 26];  }  temp1[19] = '\0';  temp2[19] = '\0';  strcpy(mas[i].name, temp1);  mas[i].year = rand() % (2023 - 1980 + 1) + 1980;  last.key = mas[i].key;  strcpy(last.name, mas[i].name);  last.year = mas[i].year;  bytes += sizeof(mas[i]);  }  file.write((char\*)mas, bytes);  file.close();  cout << "Чисел: " << N << endl;  cout << "Байтов записано: " << bytes << endl;  cout << "Последняя запись: key=" << last.key << ", name=" << last.name << ", year=" << last.year << endl;  FILE\* fp = fopen("data.bin", "rb");  int key;  cin >> key;  while (key != 0)  {  searchrecord\* table = NULL;  clock\_t start1 = clock();  createSearchTable(table, N, fp);  clock\_t end1 = clock();  double duration\_ms1 = 1000.0 \* (end1 - start1) / CLOCKS\_PER\_SEC;  clock\_t start2 = clock();  searchrecord sr = binarySearchInTable(table, N, key);  cout << sr.key << " is in " << sr.pos << endl;  record r = getRecord(fp, sr.pos);  cout << r.name << " " << r.year << endl;  clock\_t end2 = clock();  double duration\_ms2 = 1000.0 \* static\_cast<double>(end2 - start2) / CLOCKS\_PER\_SEC;  cout << "Время создания дополнительной структуры: " << duration\_ms1 << endl;  cout << "Время выполнения поиска: " << duration\_ms2 << endl;  delete[] table;  cin >> key;  }  fclose(fp);  }  return 0;  } |