Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

**«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПЕРМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра “Информационные технологии и автоматизированные системы”

**ОТЧЕТ**

**по лабораторной работе №17**

Дисциплина: “Основы теории алгоритмов и структуры данных”

Тема: Методы быстрого поиска. Хеширование

Вариант №17

Выполнил:

Студент группы РИС-20-2б

Пономарев Егор Витальевич

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

**Пермь, 2021**

**Цель работы**

1. Изучить построение функции хеширования и алгоритмов хеширования данных

2. Научиться разрабатывать алгоритмы открытого и закрытого хеширования при

решении задач на языке C++.

**Постановка задачи**

1. Создать динамический массив из записей (в соответствии с вариантом), содержащий не менее 100 элементов. Для заполнения элементов массива использовать ДСЧ.

2. Выполнить поиск элемента в массиве по ключу в соответствии с вариантом. Для поиска использовать хеш-таблицу.

3. Подсчитать количество коллизий при размере хеш-таблицы 40, 75 и 90

элементов.

**Анализ задачи**

**(1)**

**1.1** Объявить массив структур (База данных):

|  |
| --- |
| HashEl\*arr = new HashEl[arr\_len]; |

**1.2** Заполнить массив элементами и разрешение коллизий:

|  |
| --- |
| FillArray(arr, arr\_len, ElAmount); |

**1.3** Вывод массива структур и подсчета локальных и глобальных коллизий:

|  |
| --- |
| PrintDataBase(arr, arr\_len); |

**1.4** Проверка введенной даты рождения на нахождение в БД:

|  |
| --- |
| IsMatch(arr, arr\_len, birth\_date); |

**1.5** Очистить оперативную память, выделенную под массив:

|  |
| --- |
| delete[] arr; |

**(2)**

Типы данных у переменных:

**2.1** int main():

int ElAmount = 100; - кол-во элементов

int arr\_len = 40; - кол-во мест в массиве

HashEl\*arr - указатель на первый элемент массива структур

string birth\_date; - строка для поиска в БД

**2.2** FillArray():

HashEl\* arr - входной массив

int arr\_len - длина входного массива

int ElAmount - кол-во инициализированных элементов массива

int index; - индекс нового элемента массива, полученный с помощью хэш-функции

HashEl\* new\_elem - указатель на новый элемент в оперативной памяти для создания списка

string date - хранение даты рождения

int keyInt - преобразованный в целое число строковый ключ

**2.3** PrintDataBase():

list <HashEl> ::iterator it; - итератор по списку коллизий

int global\_colls = 0; - счетчик глобальных коллизий

int local\_colls = 0; - счетчик локальных коллизий

int cnt\_coll\_index = 0; - счетчик индексов с коллизиями

**2.4** IsMatch():

int cnt = 1; - счетчик кол-ва элементов массива для красивого вывода

int list\_counter = 1; - счетчик кол-ва элементов списка для красивого вывода

int index - индекс строки, которая ищется в БД, сформированная с помощью хэш-функции

bool nomatch - определитель ненахождения входной строки в БД

**(3)**

Данные представлены как массив структур HashEl\*arr:

|  |
| --- |
| HashEl\*arr = new HashEl[arr\_len]; |

**(4)**

**4.1** Ввод элементов в структуру:

Рандомизация полей элементов и запись в массив или список соответствуюoего индекса массива

|  |
| --- |
| HashEl\* new\_elem = new HashEl;  random\_FIO(new\_elem);  string date = random\_date(new\_elem);  random\_pass(new\_elem);  int keyInt = stringKeytointKey(date);  index = (HashFunc(keyInt, arr\_len)) % arr\_len;  cout << index << " index: " << new\_elem->FIO << endl;  if (arr[index].NoElem)  {  arr[index] = \*new\_elem;  arr[index].NoElem = false;  }  else  {  arr[index].lst.push\_back(\*new\_elem);  } |

**4.2** Вывод элементов структуры:

Вывод либо элемента массива на структуру, либо элемента со списком его коллизий:

|  |
| --- |
| for (int i = 0; i < arr\_len; i++)  {  if (arr[i].lst.size() != 0)  {  cout << i << " index: " << arr[i].birth << " " << arr[i].FIO << " " << arr[i].pass << "\t\tcollision on " << i << " index with:" << "\n";  for (it = arr[i].lst.begin(); it != arr[i].lst.end(); it++)  {  cout << "\t\t\t\t\t\t" << (\*it).birth << " " << (\*it).FIO << " " << (\*it).pass << endl;  local\_colls++;  }  if (local\_colls != 0) cout << "\n\t\t\t\t\t\tindex " << i << " has " << local\_colls << " local collisions\n\n";  }  else cout << i << " index: " << arr[i].birth << " " << arr[i].FIO << " " << arr[i].pass << endl; |

**(5)**

Использована структура HashEl со следующими полями:

|  |
| --- |
| struct HashEl {  string FIO;  string birth;  int pass;  HashEl\* next;  bool NoElem;  list<HashEl> lst; }; |

**(6)**

**6.1** Резервирование памяти под массив структур:

|  |
| --- |
| HashEl\*arr = new HashEl[arr\_len]; |

**6.2** Заполнение структурами массив структур:

|  |
| --- |
| FillArray(arr, arr\_len, ElAmount); |

Создается новый элемент, его поля заполняются рандомно и если на его индексе нет элемента, то он записывается туда, иначе - в список коллизий этого индекса:

|  |
| --- |
| int index;  for (int i = 0; i < ElAmount; i++)  {  HashEl\* new\_elem = new HashEl;  random\_FIO(new\_elem);  string date = random\_date(new\_elem);  random\_pass(new\_elem);  int keyInt = stringKeytointKey(date);  index = (HashFunc(keyInt, arr\_len)) % arr\_len;  cout << index << " index: " << new\_elem->FIO << endl;  if (arr[index].NoElem)  {  arr[index] = \*new\_elem;  arr[index].NoElem = false;  }  else  {  arr[index].lst.push\_back(\*new\_elem);  }    delete new\_elem;  } |

**6.3** Вывод всей БД на экран:

|  |
| --- |
| PrintDataBase(arr, arr\_len); |

Объявляется итератор по списку. На каждом индексе идет проверка на наличие у поля “список” элемента массива. Если оно имеется, то выводится элемент массива и его список коллизий, представленный списком, иначе - выводится только элемент на соответствующем индексе массива. Также считаются локальные коллизии индексов массива, то есть количество элементов на индексе массива, попавших в коллизионный список. На каждой итерации цикла ( на каждом элементе ) эти локальные коллизии прибавляются к глобальным коллизиям за все элементы БД:

|  |
| --- |
| for (int i = 0; i < arr\_len; i++)  {  if (arr[i].lst.size() != 0)  {  cout << i << " index: " << arr[i].birth << " " << arr[i].FIO << " " << arr[i].pass << "\t\tcollision on " << i << " index with:" << "\n";  for (it = arr[i].lst.begin(); it != arr[i].lst.end(); it++)  {  cout << "\t\t\t\t\t\t" << (\*it).birth << " " << (\*it).FIO << " " << (\*it).pass << endl;  local\_colls++;  }  if (local\_colls != 0) cout << "\n\t\t\t\t\t\tindex " << i << " has " << local\_colls << " local collisions\n\n";  }  else cout << i << " index: " << arr[i].birth << " " << arr[i].FIO << " " << arr[i].pass << endl;  global\_colls += local\_colls;  if (local\_colls != 0) cnt\_coll\_index++;  local\_colls = 0;   } |

**6.4** Поиск людей в БД по введенной дате рождения:

Если на индексе массива элемент без списка, то он выводится как единственный:

|  |
| --- |
| if (to\_find == arr[index].birth && arr[index].lst.size() == 0)  {  cout << "(the only person) (" << index << " index) " << arr[index].birth << " " << arr[index].FIO << " " << arr[index].pass << endl;  } |

Если элемент на индексе массива равен входной строке или не равен, то в первом случае он выводится и идет поиск в соответствующем ему списке. Во втором случае поиск идет сразу в его списке. Если совпадений не произошло, то выводится, что элемент в БД не найден:

|  |
| --- |
| else if (to\_find == arr[index].birth || to\_find != arr[index].birth)  {  bool nomatch = true;  if (to\_find == arr[index].birth)  {  nomatch = false;  cout << cnt++ << ") (" << index << " index) " << arr[index].birth << " " << arr[index].FIO << " " << arr[index].pass << endl;  }   for (it = arr[index].lst.begin(); it != arr[index].lst.end(); it++)  {  if ((\*it).birth == to\_find)  {  nomatch = false;  cout << cnt++ << ") (" << list\_counter << " element in the LIST of the " << index << " index of array) " << (\*it).birth << " " << (\*it).FIO << " " << (\*it).pass << endl;  }  list\_counter++;  }  if (nomatch == true) cout << "No match :(\n";  list\_counter = 0;  } |

**6.5** Перевод строки в целое число для вычисления ее хэш-функции:

Результат получается из полиномиальной суммы произведения кода символа строки и некой константы:

|  |
| --- |
| int stringKeytointKey(string key) {  int res = 0;  const int c = 15;  for (int i = 0; i < key.length(); i++)  {  res += ((int)key[i] \* pow(c, i)) / 1000000;  }    return res; } |

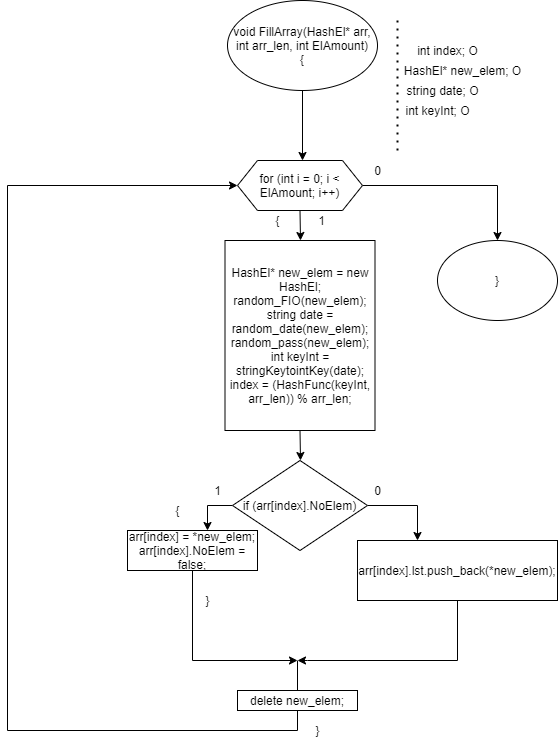
**6.6** Вычисление хэш-функции:

Вычисление хэш функции по заданной в варианте формуле:

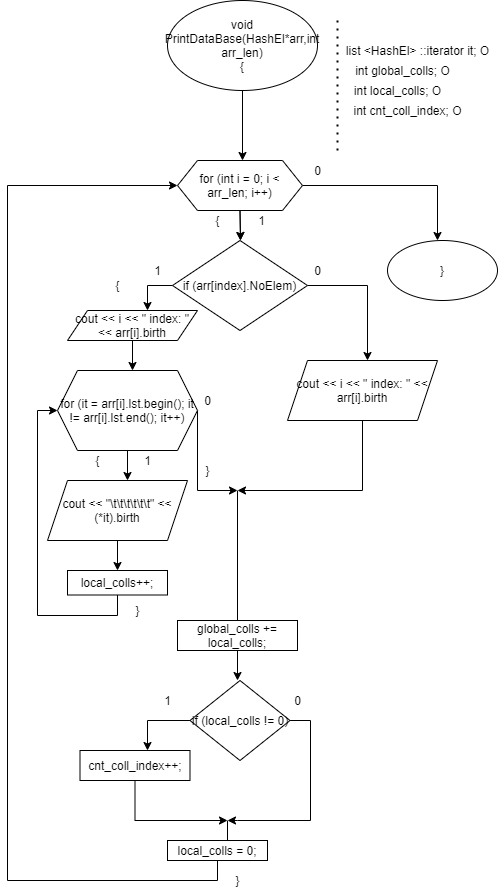
|  |
| --- |
| int HashFunc(int key, int arr\_size) {  float A = 0.153;  float k = key \* A;  return (int)(arr\_size\*(k - (int)(k)));   } |

**Блок-схемы**

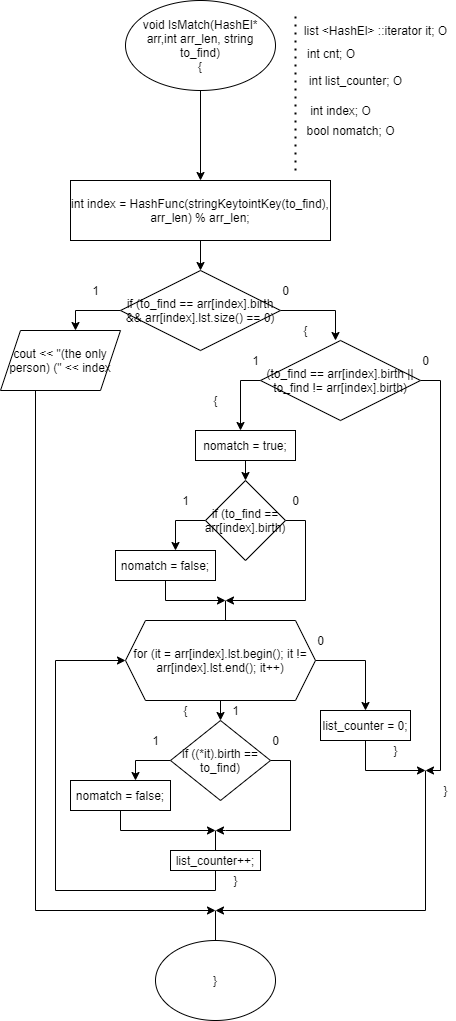
FillArray():



PrintDataBase():



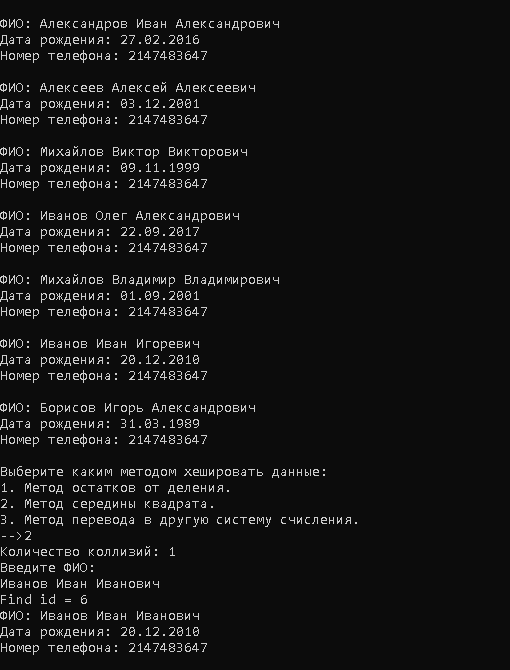
IsMatch():



**Код на C++**

|  |
| --- |
| #include <iostream> #include <string> #include <time.h> #include <list> #include <iterator> using namespace std;  struct HashEl {  string FIO;  string birth;  int pass;  HashEl\* next;  bool NoElem;  list<HashEl> lst; };  string random\_date(HashEl\* arr) {  (\*arr).birth = to\_string(rand() % 2) + to\_string(rand() % 9 + 1) + ".";  if (rand() % 2 == 0) (\*arr).birth += "0" + to\_string(rand() % 9 + 1) + "." + to\_string(rand() % 121 + 1900);  else (\*arr).birth += "1" + to\_string(rand() % 3) + "." + to\_string(rand() % 121 + 1900);  return (\*arr).birth; }  void random\_FIO(HashEl\* arr) {  for (int j = 0; j < rand() % 4 + 2; j++)  {  if (j == 0) (\*arr).FIO += toupper((char)(rand() % 26 + 97));  else (\*arr).FIO += (char)(rand() % 26 + 97);  }  (\*arr).FIO += " ";  for (int j = 0; j < rand() % 3 + 7; j++)  {  if (j == 0) (\*arr).FIO += toupper((char)(rand() % 26 + 97));  else (\*arr).FIO += (char)(rand() % 26 + 97);  } }  void random\_pass(HashEl\* arr) {  (\*arr).pass = rand() % 800001 + 100000; }  int stringKeytointKey(string key) {  int res = 0;  const int c = 15;  for (int i = 0; i < key.length(); i++)  {  res += ((int)key[i] \* pow(c, i)) / 1000000;  }    return res; }  int HashFunc(int key, int arr\_size) {  float A = 0.153;  float k = key \* A;  return (int)(arr\_size\*(k - (int)(k)));   }  void IsMatch(HashEl\* arr,int arr\_len, string to\_find) {  list <HashEl> ::iterator it;  int cnt = 1; int list\_counter = 1;  int index = HashFunc(stringKeytointKey(to\_find), arr\_len) % arr\_len;  if (to\_find == arr[index].birth && arr[index].lst.size() == 0)  {  cout << "(the only person) (" << index << " index) " << arr[index].birth << " " << arr[index].FIO << " " << arr[index].pass << endl;  }  else if (to\_find == arr[index].birth || to\_find != arr[index].birth)  {  bool nomatch = true;  if (to\_find == arr[index].birth)  {  nomatch = false;  cout << cnt++ << ") (" << index << " index) " << arr[index].birth << " " << arr[index].FIO << " " << arr[index].pass << endl;  }   for (it = arr[index].lst.begin(); it != arr[index].lst.end(); it++)  {  if ((\*it).birth == to\_find)  {  nomatch = false;  cout << cnt++ << ") (" << list\_counter << " element in the LIST of the " << index << " index of array) " << (\*it).birth << " " << (\*it).FIO << " " << (\*it).pass << endl;  }  list\_counter++;  }  if (nomatch == true) cout << "No match :(\n";  list\_counter = 0;  } }  void PrintDataBase(HashEl\*arr,int arr\_len) {  cout << "\nThe whole array with its elements:\n======================================" << endl;  list <HashEl> ::iterator it;  int global\_colls = 0; int local\_colls = 0; int cnt\_coll\_index = 0;  for (int i = 0; i < arr\_len; i++)  {  if (arr[i].lst.size() != 0)  {  cout << i << " index: " << arr[i].birth << " " << arr[i].FIO << " " << arr[i].pass << "\t\tcollision on " << i << " index with:" << "\n";  for (it = arr[i].lst.begin(); it != arr[i].lst.end(); it++)  {  cout << "\t\t\t\t\t\t" << (\*it).birth << " " << (\*it).FIO << " " << (\*it).pass << endl;  local\_colls++;  }  if (local\_colls != 0) cout << "\n\t\t\t\t\t\tindex " << i << " has " << local\_colls << " local collisions\n\n";  }  else cout << i << " index: " << arr[i].birth << " " << arr[i].FIO << " " << arr[i].pass << endl;  global\_colls += local\_colls;  if (local\_colls != 0) cnt\_coll\_index++;  local\_colls = 0;   }  cout << "=====================================";  if (global\_colls != 0) cout << "\nquantity of all collisions from " << cnt\_coll\_index << " indexes is " << global\_colls << endl;  else cout << "\nquantity of all collisions is zero\n"; }  void FillArray(HashEl\* arr, int arr\_len, int ElAmount) {  cout << "Elements with their custom indexes:\n\n";  int index;  for (int i = 0; i < ElAmount; i++)  {  HashEl\* new\_elem = new HashEl;  random\_FIO(new\_elem);  string date = random\_date(new\_elem);  random\_pass(new\_elem);  int keyInt = stringKeytointKey(date);  index = (HashFunc(keyInt, arr\_len)) % arr\_len;  cout << index << " index: " << new\_elem->FIO << endl;  if (arr[index].NoElem)  {  arr[index] = \*new\_elem;  arr[index].NoElem = false;  }  else  {  arr[index].lst.push\_back(\*new\_elem);  }    delete new\_elem;  } }  int main() {  srand(time(NULL));   int ElAmount = 100;  int arr\_len = 40;  HashEl\*arr = new HashEl[arr\_len];   FillArray(arr, arr\_len, ElAmount);   // DataBase with collision counting  PrintDataBase(arr, arr\_len);     // string match  cout << "\n\nenter a birth date u'd like to find: ";  string birth\_date;  cin >> birth\_date;  IsMatch(arr, arr\_len, birth\_date);    delete[] arr;  return 0; } |

**Результаты выполнения**

****