Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе № 17**

Дисциплина: «Основы теории алгоритмов и структуры данных»

Тема: Методы быстрого поиска. Хеширование

Вариант 21

Выполнил:

Студент группы РИС-20-1б

Ознобихин Елисей Андреевич

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

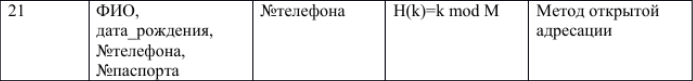
**Цель работы**

1. Изучить построение функции хеширования и алгоритмов хеширования данных
2. Научиться разрабатывать алгоритмы открытого и закрытого хеширования при

решении задач на языке C++.

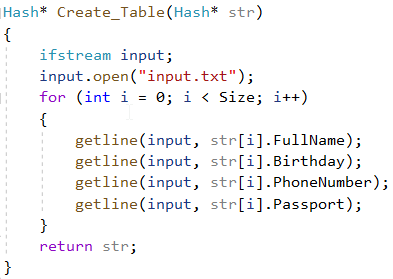
**Постановка задачи**

Выполнить поиск элемента в массиве по ключу в соответствии с вариантом. Для поиска использовать хеш-таблицу. Подсчитать количество коллизий хеш-таблицы.

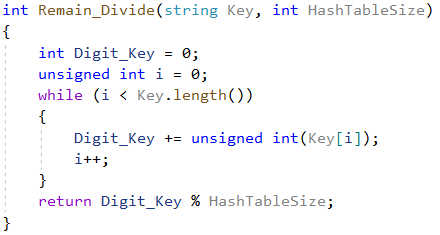


**Анализ задачи**

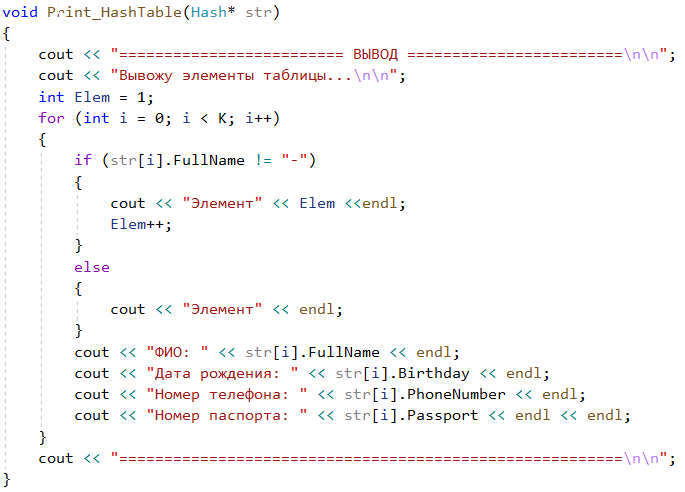
1. Для решения задачи необходимо:
   1. Реализовать функцию создания структуры данных, считываемых с файла.



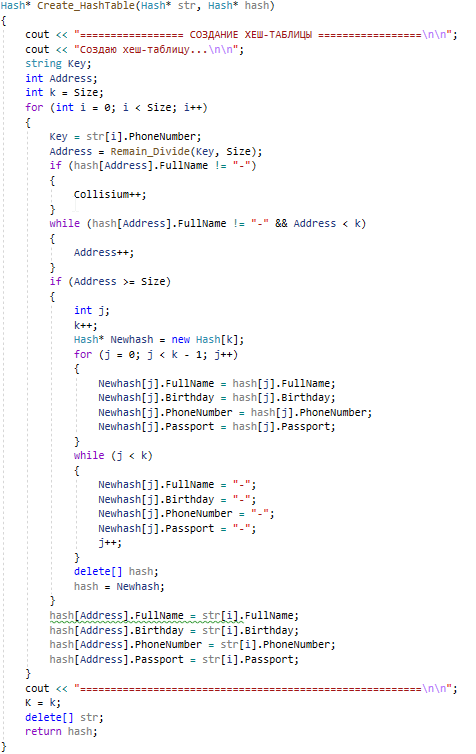
* 1. Разработать хеш-функцию типа остаток от деления.



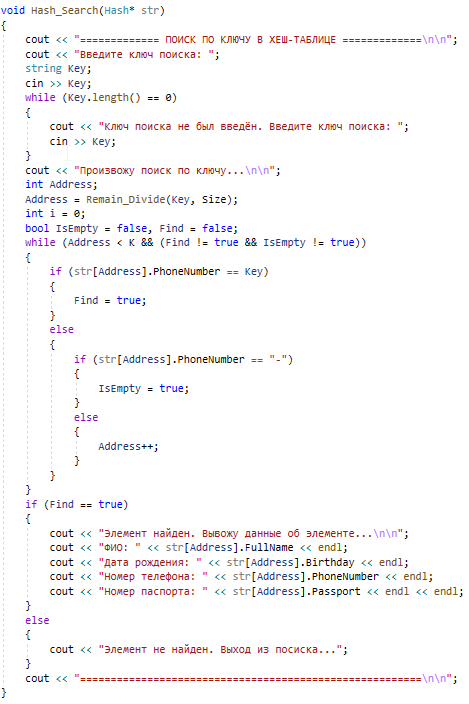
* 1. Разработать функцию вывода хеш-таблицы через консоль.



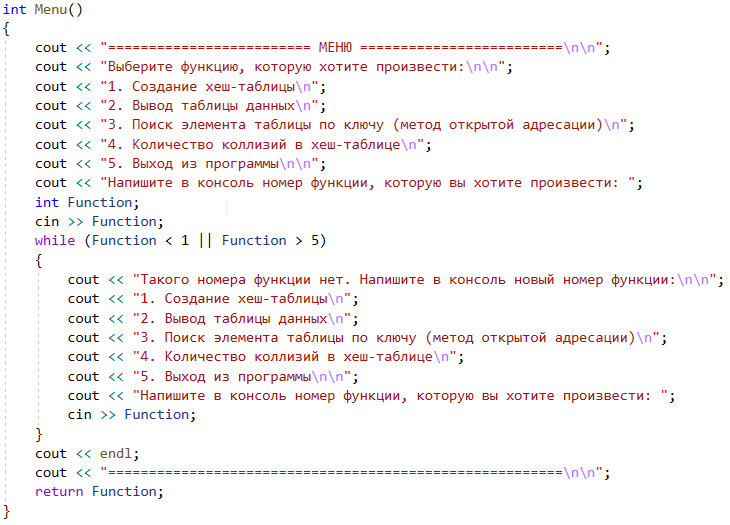
* 1. Разработать функцию создания хеш-таблицы.



* 1. Разработать функцию нахождения свободной клетки для постановки ферзя.



* 1. Разработать функцию, описывающую меню-интерфейс.



* 1. Реализовать применение этих функций в главной функции.



1. В ходе работы были использованы типы данных:
   1. Для функции Remain\_Divide используются следующие аргументы:
   2. Тип string: ключ-поле элемента.
   3. Тип int: размер хеш-таблицы.

Сама функция имеет тип указателя на int и возвращает вычисленное значение хеш-функции.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_9kgY0eEPLg.png

* 1. Для функции Create\_HashTable используются следующие аргументы:
  2. Указатель на Hash: таблица, с которой происходит считывание данных в хеш-таблицу.
  3. Указатель на Hash: пустая хеш-таблица.

Сама функция имеет тип указателя на Hash и возвращает заполненную хеш-таблицу.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_XQGIho5Gps.png

* 1. Для функции Create\_Table используются следующие аргументы:
  2. Указатель на Hash: пустая таблица данных.

Сама функция имеет тип Указателя на Hash и возвращает заполненную таблицу данных.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_ikhPwZ9B0P.png

* 1. Для функции Print\_HashTable используются следующие аргументы:
  2. Указатель на Hash: заполненная хеш-таблица.

Сама функция имеет тип void, поскольку при работе функции не нужно возвращать значение.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_QJKWCu3cbR.png

* 1. Для функции Shell\_Sort используются следующие аргументы:
  2. Указатель на Hash: хеш-таблица.

Сама функция имеет тип void, поскольку при работе функции не нужно возвращать значение.

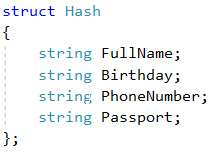
C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_eP1mKSYrH5.png

* 1. Для функции Menu не используются аргументы.

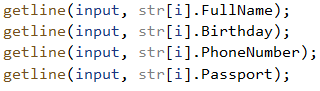
Сама функция имеет тип int и возвращает номер действия.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_MvujLbfriK.png

1. Для решения задачи данные были представлены в следующем виде:
   1. Для работы данные были записаны в структуру данных с четыремя полями для хранения данных.



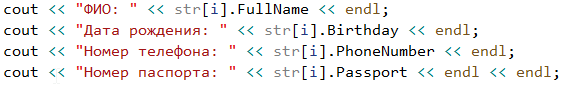
1. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Ввод данных реализован с помощью оператора cin и функции getline.



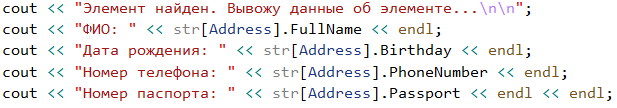
C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_SURU1bJcFF.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_w7bevpWuLY.png

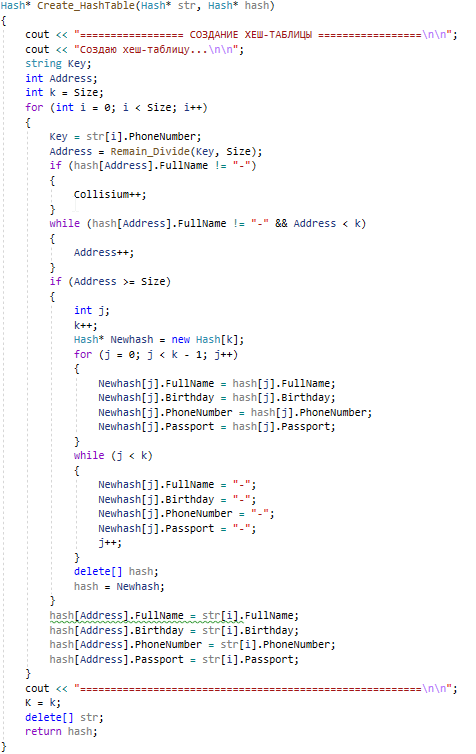
* 1. Вывод данных реализован с помощью оператора cout.

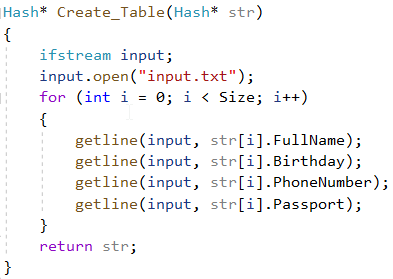


C:\Users\Асус\Documents\ShareX\Screenshots\2021-03\devenv_SvyWw3J4lt.png

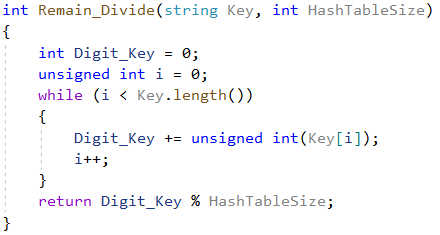


1. Поставленные задачи решены следующими действиями:
   1. Для реализации создания хеш-таблицы были реализованы функции Create\_Table и Create\_HashTable. Функция Create\_Table считывает данные об элементах с файла и записывает в структуру, функция Create\_HashTable считывает дааные с ранее заполенной структуры данных и записывает их в хеш-таблицу согласно значению вычисленного индекса с помощью хеш-функции.

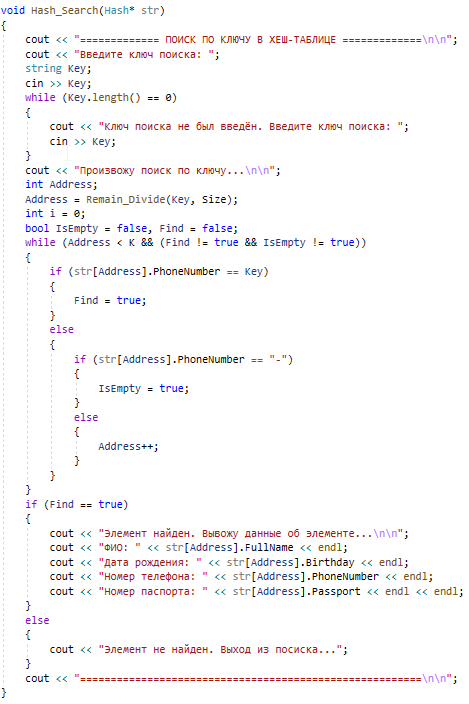




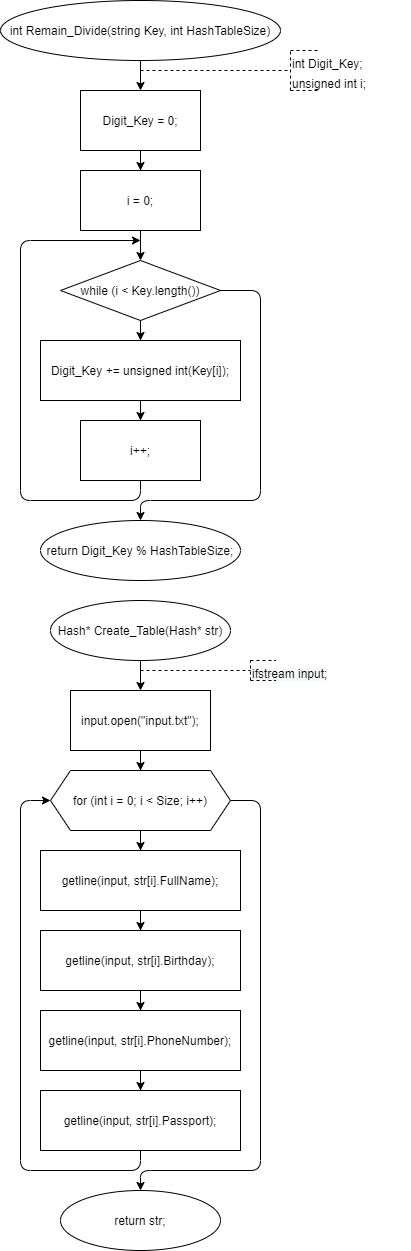
* 1. Для реализации хеш-функции использовалась дополнительная переменная Digit\_Key, которая хранит числовое значение. Значение данной переменной вычисляется путём сложения кода ASCII каждого символа в ключе.

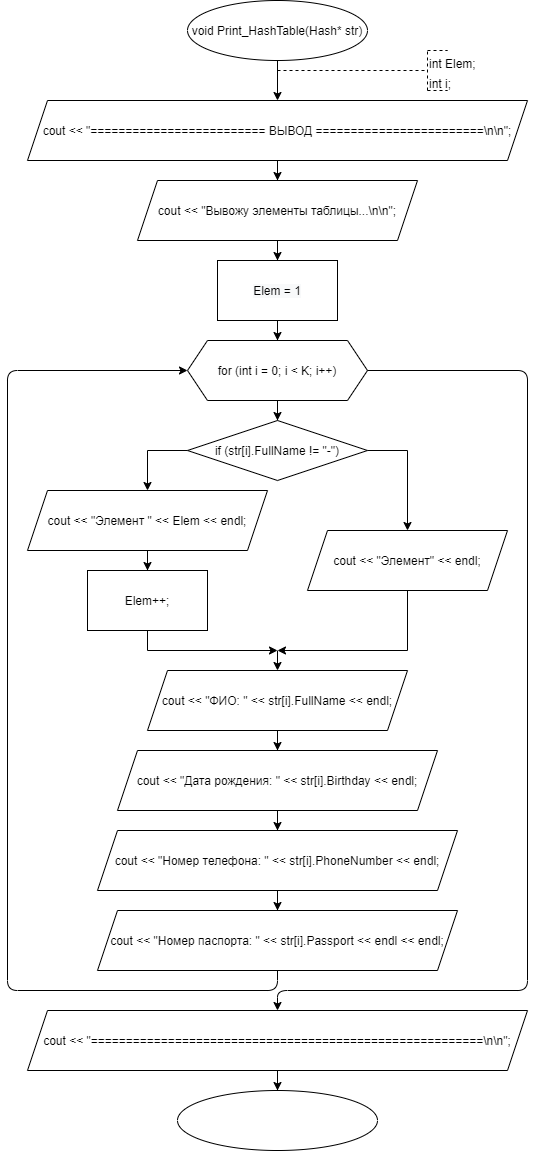


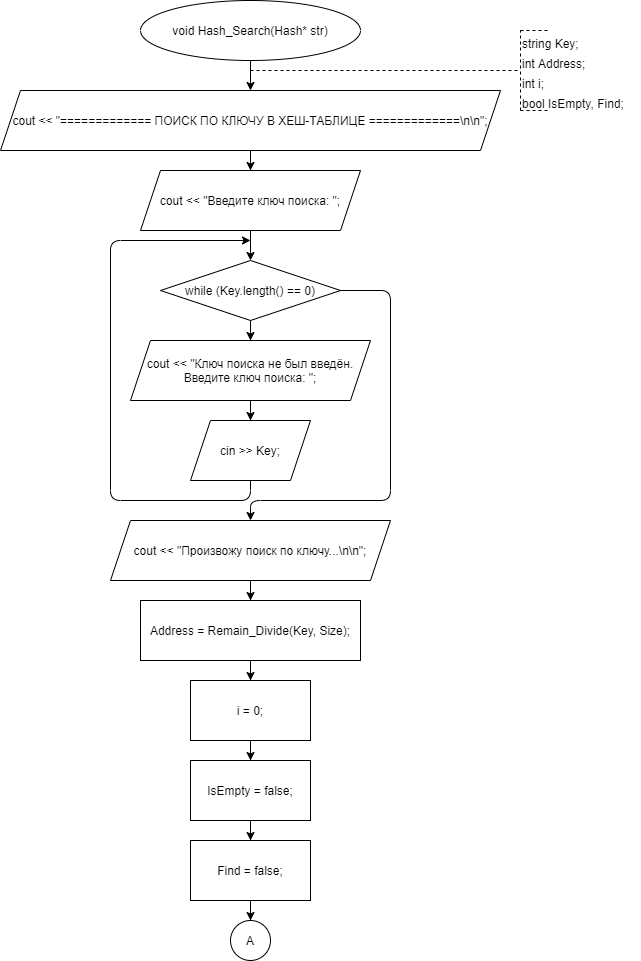
* 1. При работе функции Hash\_Search происходит вычисление индекса ключа с помощью хеш-функции. Далее выполняется цикл, в котором происходит поиск элемента по значению хеш-функции. Если по вычисленному индексу стоит заполненный элемент, поле которого не совпадает с ключом, происходит перебор всех следующих за данным элементов, пока не найдется искомый или не найдется пустой элемент.

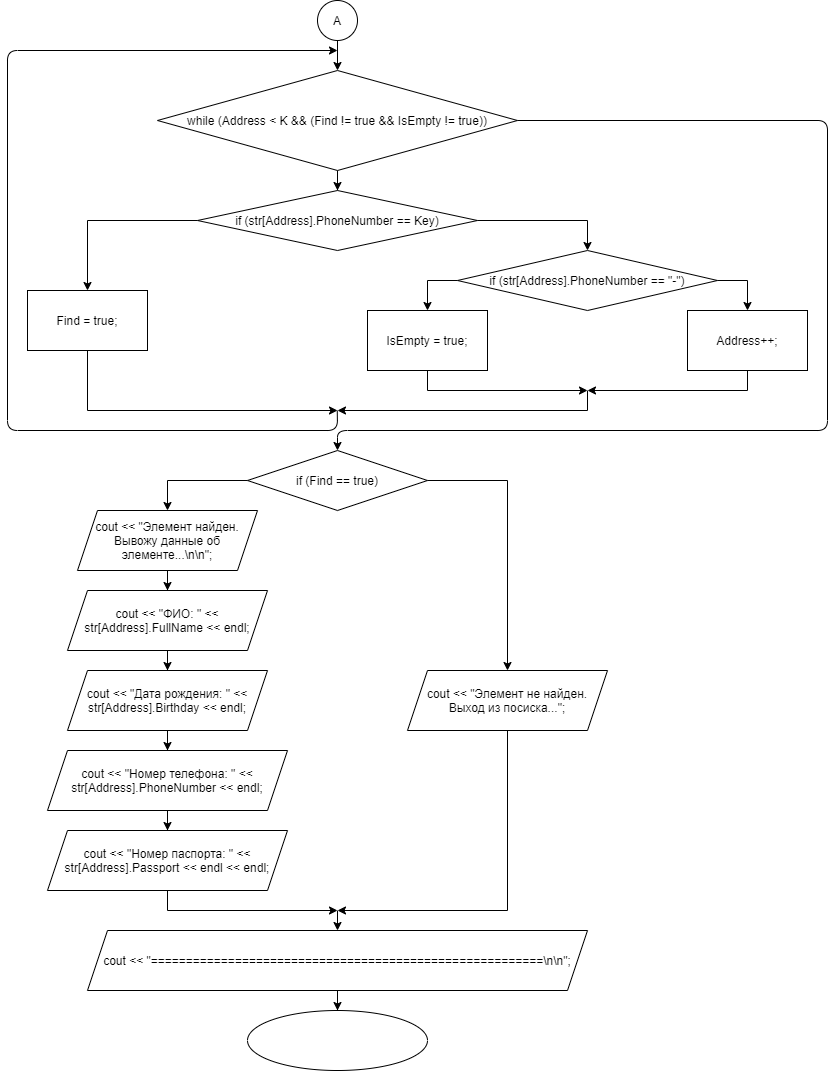


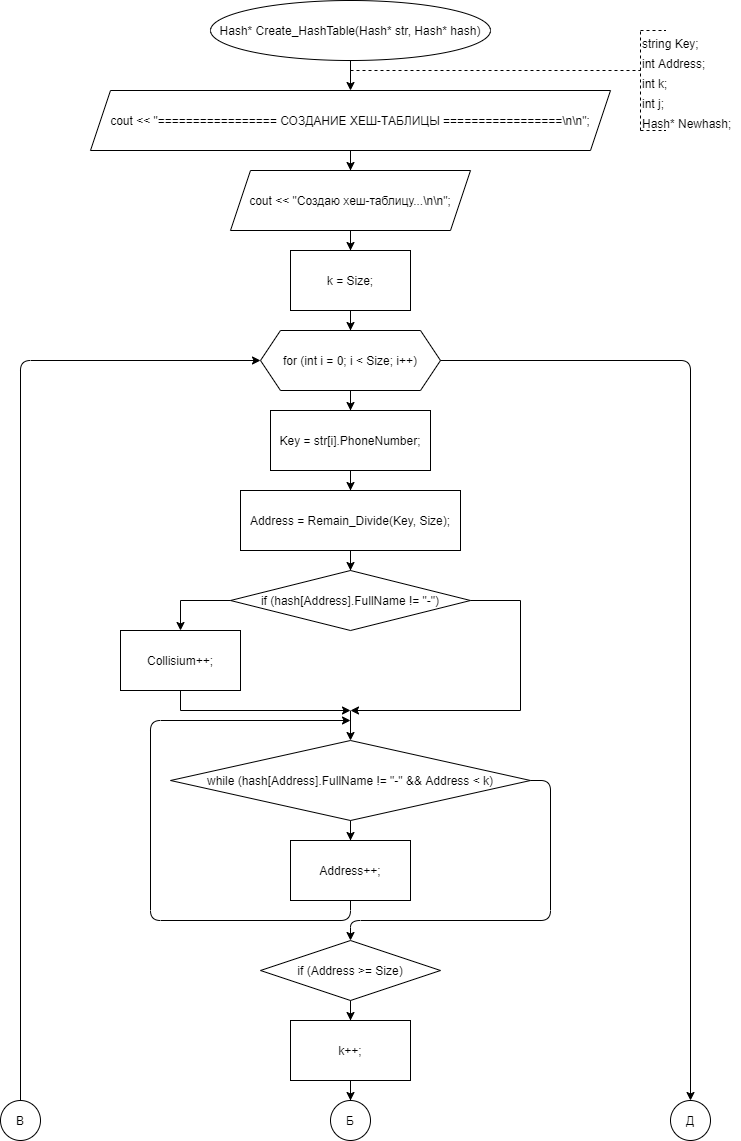
**Блок-схема программы**

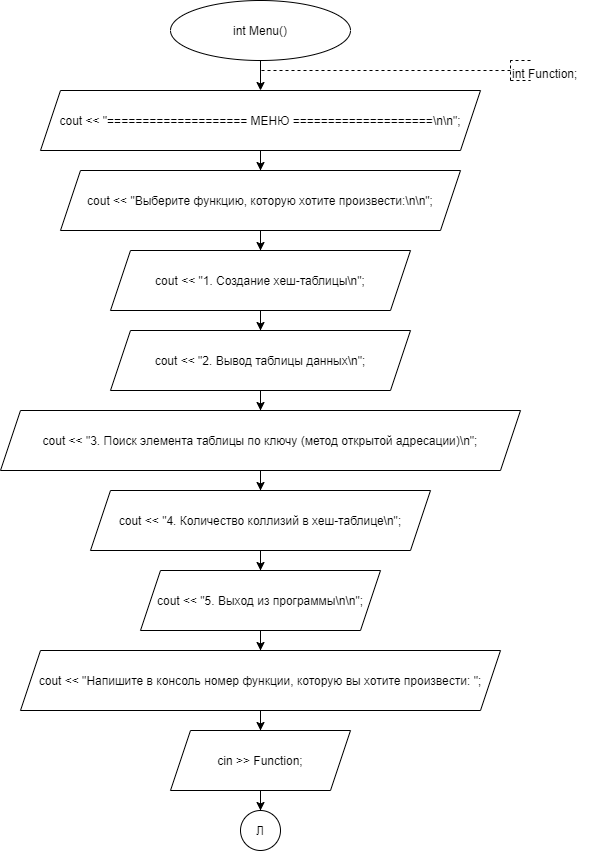


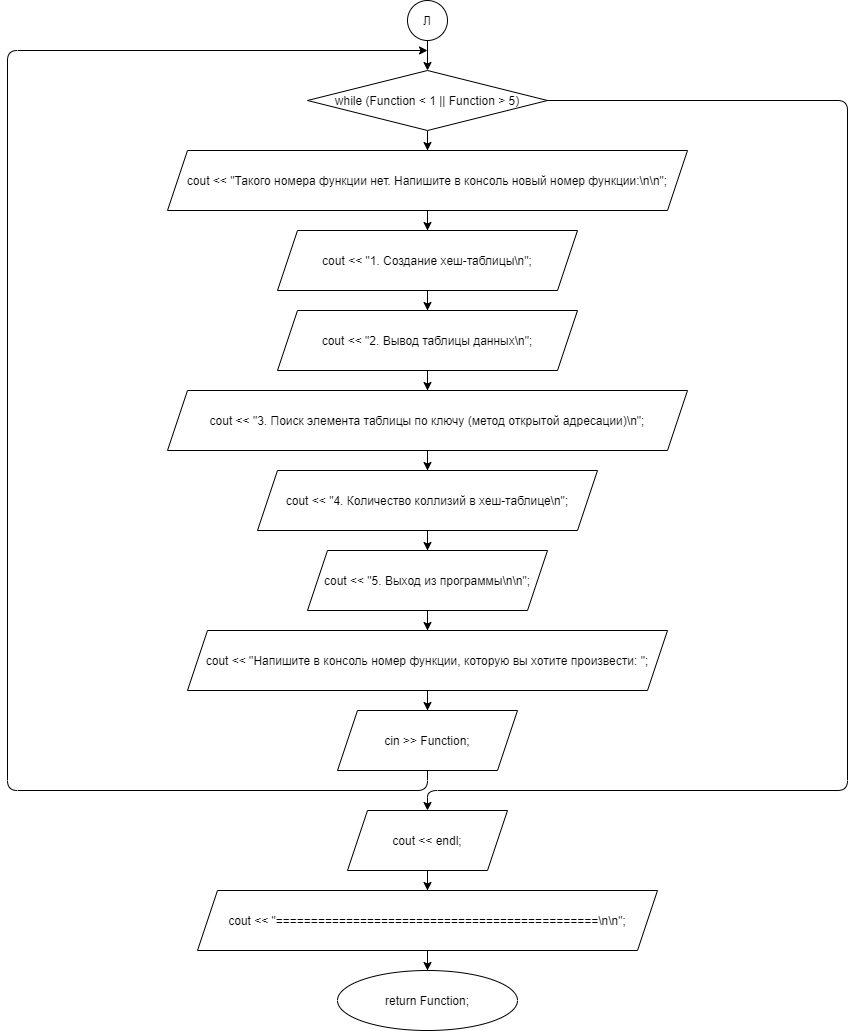


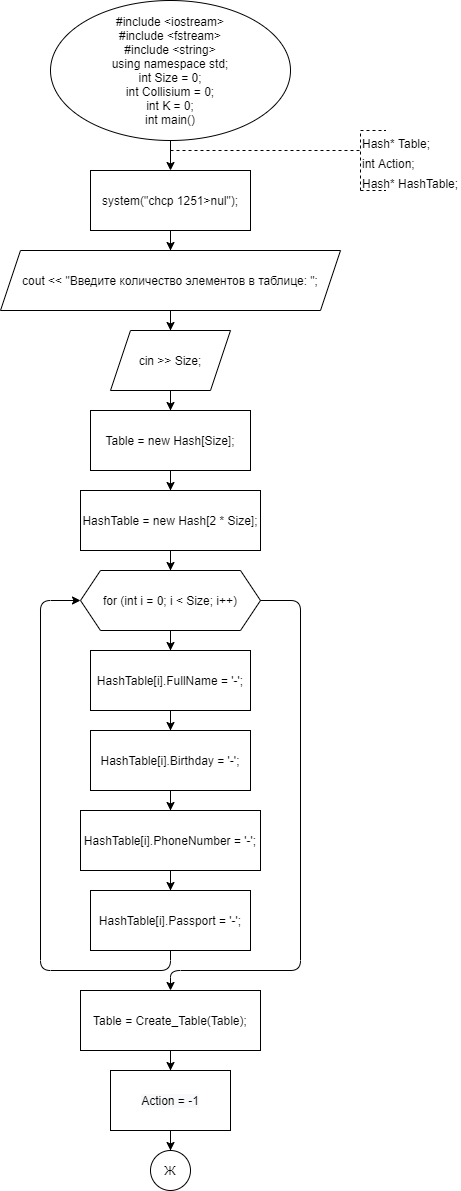


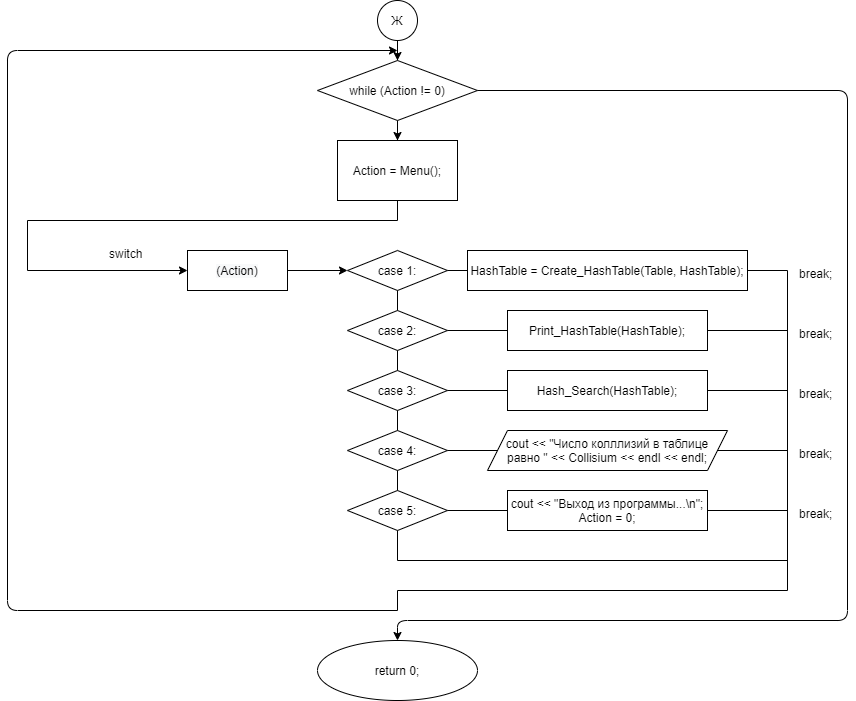












**Реализация задачи на языке С++**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int Size = 0;

int Collisium = 0;

int K = 0;

struct Hash

{

string FullName;

string Birthday;

string PhoneNumber;

string Passport;

};

int Remain\_Divide(string Key, int HashTableSize)

{

int Digit\_Key = 0;

unsigned int i = 0;

while (i < Key.length())

{

Digit\_Key += unsigned int(Key[i]);

i++;

}

return Digit\_Key % HashTableSize;

}

Hash\* Create\_HashTable(Hash\* str, Hash\* hash)

{

cout << "================= СОЗДАНИЕ ХЕШ-ТАБЛИЦЫ =================\n\n";

cout << "Создаю хеш-таблицу...\n\n";

string Key;

int Address;

int k = Size;

for (int i = 0; i < Size; i++)

{

Key = str[i].PhoneNumber;

Address = Remain\_Divide(Key, Size);

if (hash[Address].FullName != "-")

{

Collisium++;

}

while (hash[Address].FullName != "-" && Address < k)

{

Address++;

}

if (Address >= Size)

{

int j;

k++;

Hash\* Newhash = new Hash[k];

for (j = 0; j < k - 1; j++)

{

Newhash[j].FullName = hash[j].FullName;

Newhash[j].Birthday = hash[j].Birthday;

Newhash[j].PhoneNumber = hash[j].PhoneNumber;

Newhash[j].Passport = hash[j].Passport;

}

while (j < k)

{

Newhash[j].FullName = "-";

Newhash[j].Birthday = "-";

Newhash[j].PhoneNumber = "-";

Newhash[j].Passport = "-";

j++;

}

delete[] hash;

hash = Newhash;

}

hash[Address].FullName = str[i].FullName;

hash[Address].Birthday = str[i].Birthday;

hash[Address].PhoneNumber = str[i].PhoneNumber;

hash[Address].Passport = str[i].Passport;

}

cout << "========================================================\n\n";

K = k;

delete[] str;

return hash;

}

Hash\* Create\_Table(Hash\* str)

{

ifstream input;

input.open("input.txt");

for (int i = 0; i < Size; i++)

{

getline(input, str[i].FullName);

getline(input, str[i].Birthday);

getline(input, str[i].PhoneNumber);

getline(input, str[i].Passport);

}

return str;

}

void Print\_HashTable(Hash\* str)

{

cout << "========================= ВЫВОД ========================\n\n";

cout << "Вывожу элементы таблицы...\n\n";

int Elem = 1;

for (int i = 0; i < K; i++)

{

if (str[i].FullName != "-")

{

cout << "Элемент" << Elem <<endl;

Elem++;

}

else

{

cout << "Элемент" << endl;

}

cout << "ФИО: " << str[i].FullName << endl;

cout << "Дата рождения: " << str[i].Birthday << endl;

cout << "Номер телефона: " << str[i].PhoneNumber << endl;

cout << "Номер паспорта: " << str[i].Passport << endl << endl;

}

cout << "========================================================\n\n";

}

void Hash\_Search(Hash\* str)

{

cout << "============= ПОИСК ПО КЛЮЧУ В ХЕШ-ТАБЛИЦЕ =============\n\n";

cout << "Введите ключ поиска (№ телефона): ";

string Key;

cin >> Key;

while (Key.length() == 0)

{

cout << "Ключ поиска не был введён. Введите ключ поиска (№ телефона): ";

cin >> Key;

}

cout << "Произвожу поиск по ключу...\n\n";

int Address;

Address = Remain\_Divide(Key, Size);

int i = 0;

bool IsEmpty = false, Find = false;

while (Address < K && (Find != true && IsEmpty != true))

{

if (str[Address].PhoneNumber == Key)

{

Find = true;

}

else

{

if (str[Address].PhoneNumber == "-")

{

IsEmpty = true;

}

else

{

Address++;

}

}

}

if (Find == true)

{

cout << "Элемент найден. Вывожу данные об элементе...\n\n";

cout << "ФИО: " << str[Address].FullName << endl;

cout << "Дата рождения: " << str[Address].Birthday << endl;

cout << "Номер телефона: " << str[Address].PhoneNumber << endl;

cout << "Номер паспорта: " << str[Address].Passport << endl << endl;

}

else

{

cout << "Элемент не найден. Выход из посиска...";

}

cout << "========================================================\n\n";

}

int Menu()

{

cout << "========================= МЕНЮ =========================\n\n";

cout << "Выберите функцию, которую хотите произвести:\n\n";

cout << "1. Создание хеш-таблицы\n";

cout << "2. Вывод таблицы данных\n";

cout << "3. Поиск элемента таблицы по ключу (метод открытой адресации)\n";

cout << "4. Количество коллизий в хеш-таблице\n";

cout << "5. Выход из программы\n\n";

cout << "Напишите в консоль номер функции, которую вы хотите произвести: ";

int Function;

cin >> Function;

while (Function < 1 || Function > 5)

{

cout << "Такого номера функции нет. Напишите в консоль новый номер функции:\n\n";

cout << "1. Создание хеш-таблицы\n";

cout << "2. Вывод таблицы данных\n";

cout << "3. Поиск элемента таблицы по ключу (метод открытой адресации)\n";

cout << "4. Количество коллизий в хеш-таблице\n";

cout << "5. Выход из программы\n\n";

cout << "Напишите в консоль номер функции, которую вы хотите произвести: ";

cin >> Function;

}

cout << endl;

cout << "========================================================\n\n";

return Function;

}

int main()

{

system("chcp 1251>nul");

cout << "Введите количество элементов в таблице: ";

cin >> Size;

while (Size < 3 || Size > 20)

{

cout << "Число элементов в таблице не может быть меньше 3 или больше 20. Введите новое число элементов в таблице: ";

cin >> Size;

}

Hash\* Table = new Hash[Size];

Hash\* HashTable = new Hash[2 \* Size];

for (int i = 0; i < Size; i++)

{

HashTable[i].FullName = '-';

HashTable[i].Birthday = '-';

HashTable[i].PhoneNumber = '-';

HashTable[i].Passport = '-';

}

Table = Create\_Table(Table);

int Action = -1;

while (Action != 0)

{

Action = Menu();

switch (Action)

{

case 1:

{

HashTable = Create\_HashTable(Table, HashTable);

break;

}

case 2:

{

Print\_HashTable(HashTable);

break;

}

case 3:

{

Hash\_Search(HashTable);

break;

}

case 4:

{

cout << "Число колллизий в таблице равно " << Collisium << endl << endl;

break;

}

case 5:

{

cout << "Выход из программы...\n";

Action = 0;

break;

}

}

}

}

**Скриншоты результатов работы программы**

