Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторным работам №12,14**

Дисциплина: основы алгоритмизации и программирования

Тема: “Методы поиска: линейный, интерполяционный, прямой поиск подстроки в строке, Бойера-Мура, Кнута-Морриса-Пратта"

**Вар.21**

Выполнил работу

студент группы ИВТ-20-2Б

Галинов О.Ю.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2020

**Цель задачи**

1) Получить практические навыки работы с поисками;

2) Получить практические навыки работы со списками;

3) Получить практические навыки работы с меню;  
Для достижения поставленной цели необходимо решить следующие задачи:

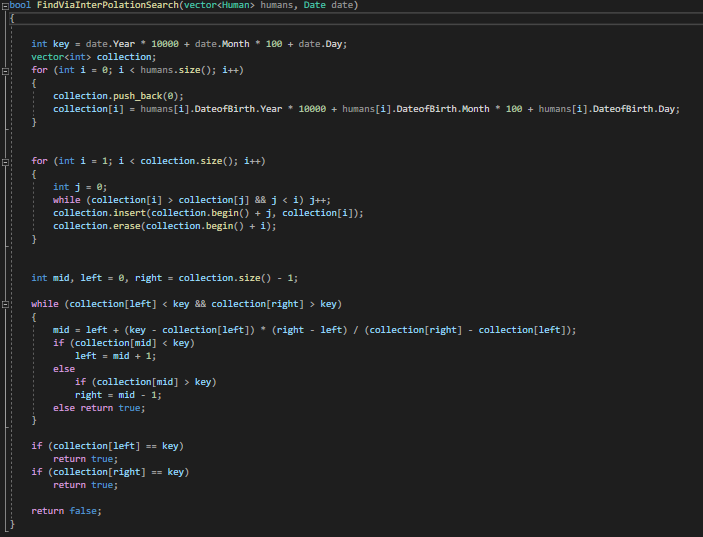
* Разработать структуру человека.
* Организовать ввод-вывод данных.
* Реализовать меню.
* Разработать функцию, заполняющую массив объектов структуры случайными числами.
* Реализовать функции поиска.
* Реализовать функцию, сохраняющую измененный массив в файл.
* Разработать программу.

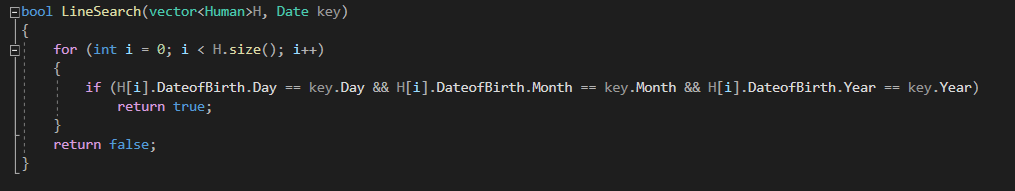
**Постановка задачи**

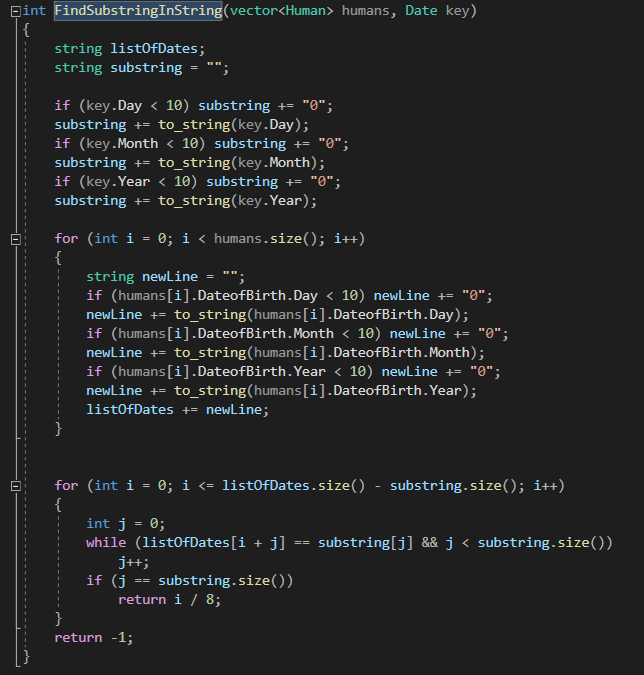
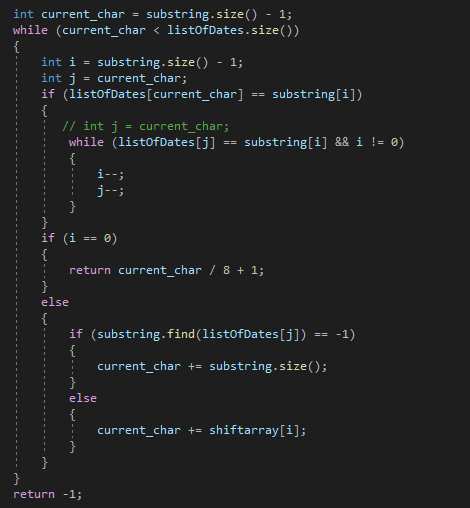
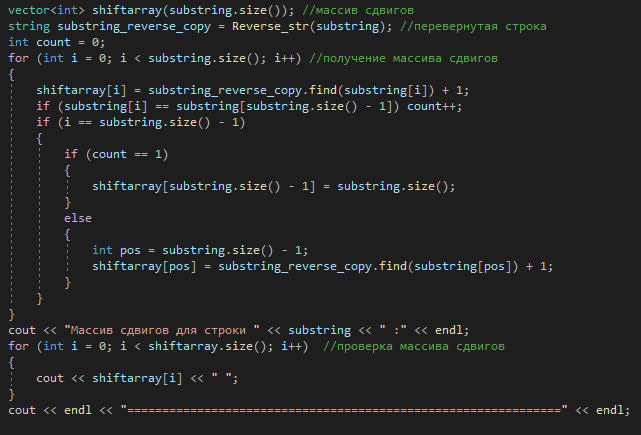
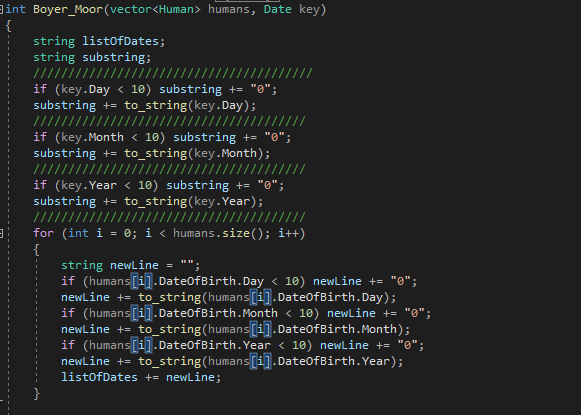
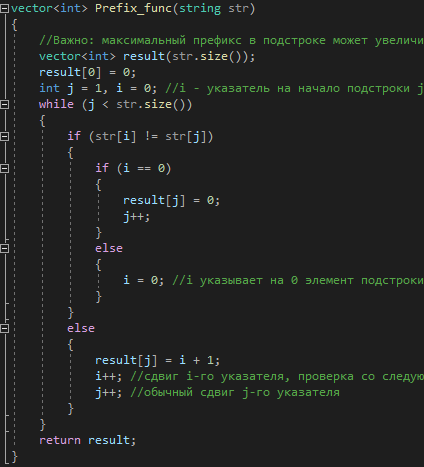
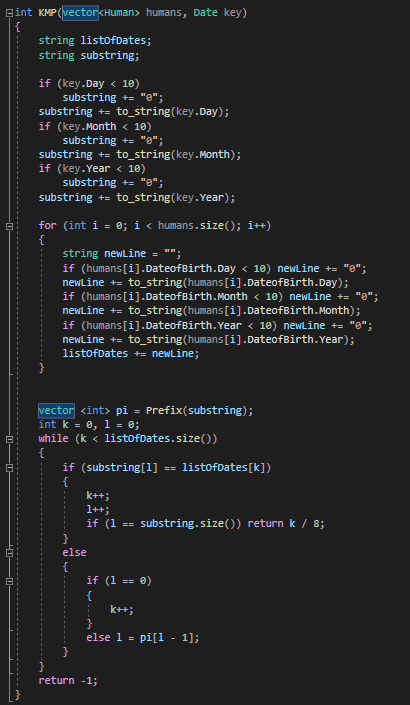
1. Создать динамический массив из записей (в соответствии с вариантом), содержащий не менее 100 элементов. Для заполнения элементов массива использовать ДСЧ.  
2. Предусмотреть сохранение массива в файл и загрузку массива из файла.  
3. Выполнить поиск элемента в массиве по ключу в соответствии с вариантом. Для поиска использовать метод линейного поиска, метод Прямого поиска подстроки в строке и интерполяционный метод.

ФИО, дата\_рождения, №телефона,  
ключ: дата\_рождения,

**Анализ задачи**

1. Для решения задачи необходимо:
   1. Разработать функцию меню.
   2. Разработать функцию, заполняющую массив с помощью ДСЧ.
   3. Разработать функции поиска по ключу.
   4. Реализовать структуру Human с полями FIO, DateOfBirth, PhoneNumber.
   5. Реализовать структуру Date, с полями Day, Month, Year. Поле DateOfBirth структуры Human будет типа Date.
   6. Разработать функцию записи массива в файл.
   7. Реализовать функцию сортировки массива для Интерполяционного поиска.
   8. Разработать функцию переворота строки для поиска Бойера-Мура.
   9. Разработать префикс-функцию для метода КМП.
2. В ходе работы были использованы следующие типы данных:
   1. Поля структуры Date – int.
   2. Поле FIO структуры Human типа string, PhoneNumber – long long.
   3. Для хранения объектов структуры Human используется класс vector.
   4. Функции поиска, кроме Интерполяционного и Линейного, возвращают тип данных int, функции Интерполяционного и линейного поиска возвращает тип данных bool.
   5. Для записи массива в файл используется класс ofstream.
   6. Функции поиска принимают в качестве параметров массив и ключ, типа Date.
3. Для решения задачи данные были представлены в следующем виде:
   1. Для хранения информации о человеке используется структура Human с 3 полями (см.п.1.6 и п.1.7).
4. Структуры, которые использовались при решении задачи:
   1. Структура Human имеет метод заполнения объекта (CreateData), которая заполняет поля объекта с помощью ДСЧ, для заполнения поля FIO используется 3 алфавита (имен, фамилий, отчеств). Номер телефона - случайное число. Аналогично, заполняются поле DateOfBirth.
   2. Структура Date имеет метод заполнения с помощью случайных чисел (день 1-31, месяц 1-12, год 1900-2021)
5. Для операций ввода и вывода использовались следующие операторы и функции:
   1. cin и cout в функциях Search, PrintArray.
6. Поставленные задачи будут решены следующими действиями:
   1. Функция FindViaInterPolationSearch преобразует массив объектов структуры в массив дат, приведенных к восьмизначному числу. Ключ преобразуется по такому же принципу. Для сортировки массива используется сортировка Шелла. С помощью цикла while, который выполняется до тех пор, пока значение левой границы меньше равно ключу и значение правой границы больше равно ключу. Внутри цикла рассчитывается индекс серединного элемента. Если значение серединного элемента меньше ключа, то левой границе присваивается инкрементированный индекс середины. Если же значение серединного элемента больше ключа, то правой границе присваивается декрементированный индекс середины. Если же значение mid совпадает с ключом, то функция возвращает true, в случае если элемента в массиве нет, то функция возвращает значение false.  
        
      
   2. Функция LineSearch С помощью цикла for осуществляет проход по всем элементам массива, в случае удовлетворения элемента массива ключу, функция возвращает true, иначе false.



* 1. Функция FindSubstringInString преобразует массив объектов структуры Human в строку дат, ключ так же преобразовывается в строку (важно отметить, что поле даты так же приводится к восьмизнаковой строке). Проход осуществляется с помощью цикла for, который выполняется до тех пор, пока индекс не станет равен длине строки без подстроки. Внутри цикла for используется цикл while, который считает количество совпавших символов, если количество совпавших символов равно длине подстроки, то возвращается значение i/8.  
     
  2. Функция BMSearch аналогично функции FindSubstringInString преобразует исходный массив и ключ в строки. Так же для создания массива сдвигов используется функия Reverse\_str, которая возвращает перевернутую строку. Массив сдвигов заполняется с помощью цикла for, i-ый элемент массива равен индексу i символа подстроки в перевернутой подстроке. Если же последний элемент подстроки встречается 1 раз, то в массив смещений для этого символа добавляется значение размера подстроки, иначе значение первого вхождения в эту подстроку. Для наглядности пользователю выводится исходная подстрока и массив сдвигов. Проход по элементам строк осуществляется с помощью цикла while, условием которого является проверка на выход за границы строки. Проверка символов подстроки в строке осуществляется с помощью индексов: current\_char, i, j (current\_char и j отвечают за строку, i - за подстроку). Внутри цикла проверяется сходство символом строки и подстроки, в случае совпадения, циклом while индексы i и j декрементируются, условием цикла является listOfDates[j] == substring[i] && i != 0. Если значение i равно нулю, то пользователю возвращается значение current\_char/8 +1, иначе проверяется наличие несовпавшего символа в алфавите подстроки. Если такой символ есть в алфавите, то current\_char сдвигается на количество равное i-ому элементу в массиве сдвигов, иначе сдвиг осуществляется на длину подстроки. Если элемента в массиве нет, то функция возвращает значение -1.  
     
  3. Функция KMP аналогично пунктам 6.4 и 6.3 осуществляет преобразование ключа и массива в строки. Для заполнения массива префиксов реализована функция Prefix\_func, которая принимает в качестве параметра подстроку и возвращает массив префиксов. Массив префиксов заполняется с помощью цикла for, 0 индексу массива префиксов соответствует значение 0, так же важно отметить, **что максимальный префикс в подстроке может увеличиваться только на 1, но изменяться на неограниченное количество (может стать нулевым)**. Для подсчета длины префиксов используются два указателя (i, j). J отвечает за проход по подстроке, i - за подсчет длины префикса. Заполнение осуществляется с помощью цикла while, условием которого проверяется выход за границы индексом j. В случае несовпадения i и j элементов подстроки, проверяется значения индекса i, если он нулевой, то в массив префиксов с индексом j добавляется нулевое значение, j инкрементируется. Если же i ненулевое значение, то i обнуляется. В случае совпадения i и j элементов подстроки, j индексу массива префиксов присваивается значение длины предыдущего префикса + 1, i и j инкрементируются.   
       
       
     Для нахождения подстроки в строке используются два указателя на элементы строки и подстроки (k и l соответственно). Проход осуществляется с помощью цикла while, который проверят выход k за границы строки. Внутри цикла проверяется совпадение символов строки и подстроки. Если же они совпали, то индексы инкрементируются (так же проверяется значение l, если оно равно длине подстроки, то функция возвращает k/8). В ином случае проверяется значение l, если оно нулевое, то k инкрементируется, иначе l указывает на индекс подстроки равный значению массива pi с индексом [l-1]. Если элемента в массиве нет, функция возвращает -1.  
     

**Код программы**

#include <iostream>

#include <string>

#include<fstream>

#include<vector>

using namespace std;

struct Date

{

int Day, Month, Year;

};

struct Human

{

string FIO;

long long PhoneNumber;

Date DateofBirth;

void Print()

{

cout << "=============================================" << endl;

cout << "ФИО: " << FIO << endl;

cout << "Дата рождения: " << DateofBirth.Day << "." << DateofBirth.Month << "." << DateofBirth.Year << endl;

cout << "Номер телефона: " << PhoneNumber << endl;

}

void CreateData()

{

string names[] = { "Геннадий", "Максим", "Вячеслав", "Алексей", "Дмитрий"};

string surnames[]= { "Иванов", "Максимов", "Петров", "Алексеев", "Сидоров"};

string otchestvoes[] = {"Игоревич", "Александрович", "Юрьевич", "Николаевич", "Сергеевич" };

FIO = surnames[rand() % 5] + " " + names[rand() % 5] + " " + otchestvoes[rand() % 5];

DateofBirth.Day = 1 + rand() % 31;

DateofBirth.Month = 1 + rand() % 12;

DateofBirth.Year = 1900 + rand() % 122;

PhoneNumber = 89000000000 + rand()\*30000 + rand() % 10000;

}

};

vector<Human> CreateArray(int size = 100)

{

vector<Human> res;

srand(time(0));

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

{

Human newEl;

newEl.CreateData();

res.push\_back(newEl);

}

cout << "Структура создана" << endl;

return res;

}

void PrintArray(vector<Human> humans)

{

for (int i = 0; i < humans.size(); i++)

{

humans[i].Print();

cout << "Номер человека " << i + 1 << endl << endl;

}

}

void SaveToFile(vector<Human> humans)

{

string path = "database.txt";

ofstream out = ofstream();

out.open(path);

for (int i = 0; i < humans.size(); i++)

{

out << humans[i].FIO << endl;

out << humans[i].DateofBirth.Day << "." << humans[i].DateofBirth.Month << "." << humans[i].DateofBirth.Year << endl;

out << humans[i].PhoneNumber << endl;

}

out.close();

cout << "Ваша структура была скопирована в database.txt" << endl;

}

bool FindViaInterPolationSearch(vector<Human> humans, Date date)

{

int key = date.Year \* 10000 + date.Month \* 100 + date.Day;

vector<int> collection;

for (int i = 0; i < humans.size(); i++)

{

collection.push\_back(0);

collection[i] = humans[i].DateofBirth.Year \* 10000 + humans[i].DateofBirth.Month \* 100 + humans[i].DateofBirth.Day;

}

for (int i = 1; i < collection.size(); i++)

{

int j = 0;

while (collection[i] > collection[j] && j < i) j++;

collection.insert(collection.begin() + j, collection[i]);

collection.erase(collection.begin() + i);

}

int mid, left = 0, right = collection.size() - 1;

while (collection[left] < key && collection[right] > key)

{

mid = left + (key - collection[left]) \* (right - left) / (collection[right] - collection[left]);

if (collection[mid] < key)

left = mid + 1;

else

if (collection[mid] > key)

right = mid - 1;

else return true;

}

if (collection[left] == key)

return true;

if (collection[right] == key)

return true;

return false;

}

int FindSubstringInString(vector<Human> humans, Date key)

{

string listOfDates;

string substring = "";

if (key.Day < 10) substring += "0";

substring += to\_string(key.Day);

if (key.Month < 10) substring += "0";

substring += to\_string(key.Month);

if (key.Year < 10) substring += "0";

substring += to\_string(key.Year);

for (int i = 0; i < humans.size(); i++)

{

string newLine = "";

if (humans[i].DateofBirth.Day < 10) newLine += "0";

newLine += to\_string(humans[i].DateofBirth.Day);

if (humans[i].DateofBirth.Month < 10) newLine += "0";

newLine += to\_string(humans[i].DateofBirth.Month);

if (humans[i].DateofBirth.Year < 10) newLine += "0";

newLine += to\_string(humans[i].DateofBirth.Year);

listOfDates += newLine;

}

for (int i = 0; i <= listOfDates.size() - substring.size(); i++)

{

int j = 0;

while (listOfDates[i + j] == substring[j] && j < substring.size())

j++;

if (j == substring.size())

return i / 8;

}

return -1;

}

bool LineSearch(vector<Human>H, Date key)

{

for (int i = 0; i < H.size(); i++)

{

if (H[i].DateofBirth.Day == key.Day && H[i].DateofBirth.Month == key.Month && H[i].DateofBirth.Year == key.Year)

return true;

}

return false;

}

vector<int> Prefix(string str)

{

vector<int> result(str.size());

result[0] = 0;

int j = 1, i = 0;

while (j < str.size())

{

if (str[i] != str[j])

{

if (i == 0)

{

result[j] = 0;

j++;

}

else

{

i = 0;

}

}

else

{

result[j] = i + 1;

i++;

j++;

}

}

return result;

}

int KMP(vector<Human> humans, Date key)

{

string listOfDates;

string substring;

if (key.Day < 10)

substring += "0";

substring += to\_string(key.Day);

if (key.Month < 10)

substring += "0";

substring += to\_string(key.Month);

if (key.Year < 10)

substring += "0";

substring += to\_string(key.Year);

for (int i = 0; i < humans.size(); i++)

{

string newLine = "";

if (humans[i].DateofBirth.Day < 10) newLine += "0";

newLine += to\_string(humans[i].DateofBirth.Day);

if (humans[i].DateofBirth.Month < 10) newLine += "0";

newLine += to\_string(humans[i].DateofBirth.Month);

if (humans[i].DateofBirth.Year < 10) newLine += "0";

newLine += to\_string(humans[i].DateofBirth.Year);

listOfDates += newLine;

}

vector <int> pi = Prefix(substring);

int k = 0, l = 0;

while (k < listOfDates.size())

{

if (substring[l] == listOfDates[k])

{

k++;

l++;

if (l == substring.size()) return k / 8;

}

else

{

if (l == 0)

{

k++;

}

else l = pi[l - 1];

}

}

return -1;

}

string Reverse\_str(string str)

{

string result;

for (int i = str.size() - 1; i >= 0; i--)

{

result.push\_back(str[i]);

}

return result;

}

int BMSearch(vector<Human> humans, Date key)

{

string listOfDates;

string substring;

////////////////////////////////////////

if (key.Day < 10) substring += "0";

substring += to\_string(key.Day);

///////////////////////////////////////

if (key.Month < 10) substring += "0";

substring += to\_string(key.Month);

///////////////////////////////////////

if (key.Year < 10) substring += "0";

substring += to\_string(key.Year);

///////////////////////////////////////

for (int i = 0; i < humans.size(); i++)

{

string newLine = "";

if (humans[i].DateofBirth.Day < 10) newLine += "0";

newLine += to\_string(humans[i].DateofBirth.Day);

if (humans[i].DateofBirth.Month < 10) newLine += "0";

newLine += to\_string(humans[i].DateofBirth.Month);

if (humans[i].DateofBirth.Year < 10) newLine += "0";

newLine += to\_string(humans[i].DateofBirth.Year);

listOfDates += newLine;

}

vector<int> shiftarray(substring.size());

string substring\_reverse\_copy = Reverse\_str(substring);

int count = 0;

for (int i = 0; i < substring.size(); i++)

{

shiftarray[i] = substring\_reverse\_copy.find(substring[i]) + 1;

if (substring[i] == substring[substring.size() - 1]) count++;

if (i == substring.size() - 1)

{

if (count == 1)

{

shiftarray[substring.size() - 1] = substring.size();

}

else

{

int pos = substring.size() - 1;

shiftarray[pos] = substring\_reverse\_copy.find(substring[pos]) + 1;

}

}

}

cout << "Массив сдвигов для строки " << substring << " :" << endl;

for (int i = 0; i < shiftarray.size(); i++)

{

cout << shiftarray[i] << " ";

}

cout << endl << "==============================================================" << endl;

int current\_char = substring.size() - 1;

while (current\_char < listOfDates.size())

{

int i = substring.size() - 1;

int j = current\_char;

if (listOfDates[current\_char] == substring[i])

{

// int j = current\_char;

while (listOfDates[j] == substring[i] && i != 0)

{

i--;

j--;

}

}

if (i == 0)

{

return current\_char / 8 + 1;

}

else

{

if (substring.find(listOfDates[j]) == -1)

{

current\_char += substring.size();

}

else

{

current\_char += shiftarray[i];

}

}

}

return -1;

}

void Search(vector<Human> humans)

{

Date key;

cout << "Введите интересующую дату рождения" << endl;

cin >> key.Day >> key.Month >> key.Year;

bool f = FindViaInterPolationSearch(humans, key);

cout << endl << "Поиск через интерполяционный метод" << endl;

if (f)

cout << "Элемент найден" << endl;

else

cout << "Элемент не найден" << endl;

f = LineSearch(humans, key);

cout << endl << "Метод линейного поиска" << endl;

if (f)

cout << "Элемент найден" << endl;

else

cout << "Элемент не найден" << endl;

int ind = FindSubstringInString(humans, key);

cout << endl << "Метод поиска подстроки в строке" << endl;

if (ind != -1)

{

cout << "Элемент найден" << endl;

cout << "Инициалы интересующего элемента:" << endl << humans[ind].FIO << endl << endl;

}

else

cout << "Элемент не найден" << endl;

}

int main()

{

setlocale(LC\_ALL, "rus");

vector<Human>Humans;

char vvod = ' ';

while (vvod != '8')

{

cout << "1 - Создать структуру" << endl;

cout << "2 - Вывести структуру" << endl;

cout << "3 - Записать в файл" << endl;

cout << "4 - Выполнить поиск" << endl;

cin >> vvod;

switch (vvod)

{

case '1': Humans = CreateArray(); break;

case '2': PrintArray(Humans); break;

case '3': SaveToFile(Humans); break;

case '4': Search(Humans); break;

}

}

int pos = KMP(humans, key);

cout << "Сортировка методом Кнутта-Морриса-Пратта:" << endl;

if (pos == -1)

cout << "Нет такой даты рождения" << endl;

else

cout << "Есть такая дата рождения" << endl;

int BM = BMSearch(humans, key);

cout << "Сортировка методом Бойера-Мура:" << endl;

if (BM == -1)

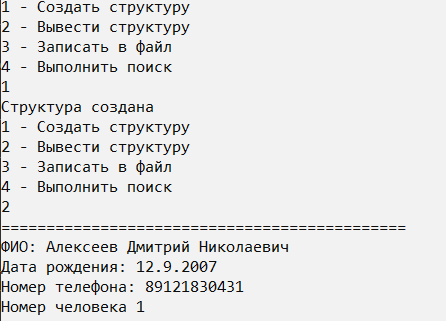
cout << "Нет такой даты рождения" << endl;

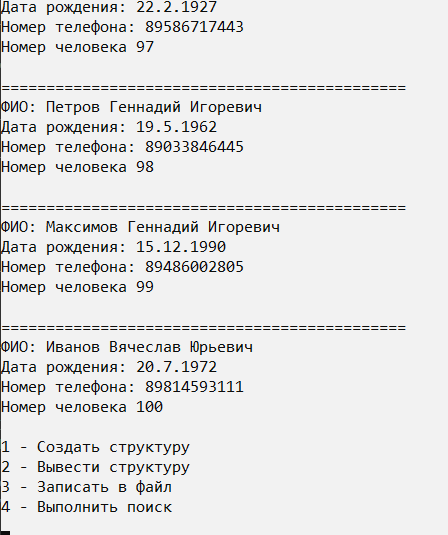
else

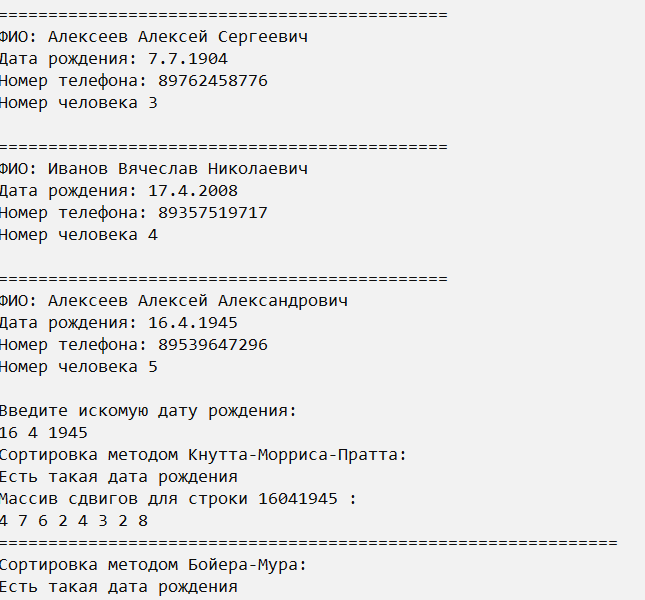
cout << "Есть такая дата рождения" << endl;

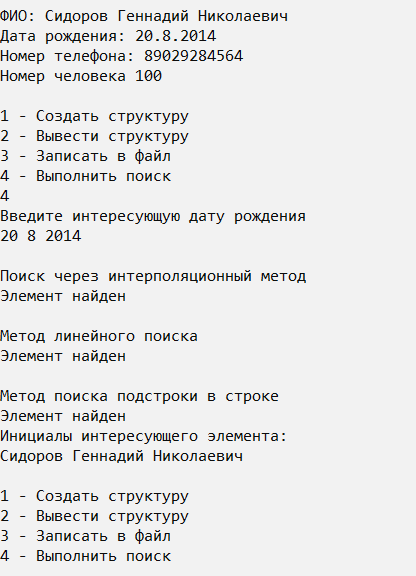
}

**Работа кода**

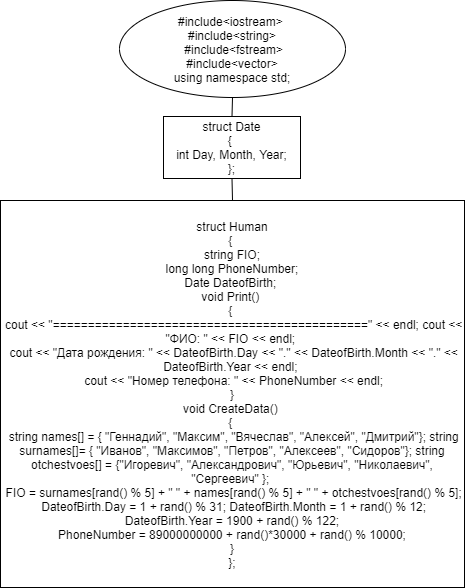
****

****

****

****

**Блок-схема**

****

