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 №14**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: “Поиск данных в динамических структурах по заданному ключу ”

Вариант 23

Выполнил:

Студент группы ИВТ-20-2Б Брейкин Алексей Дмитриевич

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**

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

2. Выполнить поиск элемента в массиве по ключу в соответствии с вариантом. Для поиска использовать метод Кнута-Морриса-Пратта, Бойера-Мура и подстроки в строке.

**Анализ задачи**

1. Для решения задачи необходимо:
   1. Определить функцию void KMP(DATA\* human, int size, int\* PI, string str, которая отвечает за поиск методом Кнутта-Морриса-Пратта.
   2. Организовать функцию void BM(DATA\* human, string strC, int\* Help, string str, int size, H\* HE), которая отвечает за поиск методом Бойера-Мура.
   3. Организовать фукнцию void InitPI(DATA\* human, int size) для инициализации массива пи исходя из заданной пользователем строки.
   4. Определить функцию int SearchInTable(char sy, string str, int\* Help) для поиска в таблице при поиске методом Бойера-Мура
   5. Определить фукнцию void InitTable(DATA\* human, int size) для инициализации таблицы по которолй будет выполняться поиск при методе Бойера-Мура
   6. Определить функцию void PrintH(DATA\* human, int size) для печать сгенерированного массива
   7. Определить функцию void humanINIT(DATA\* human, int size) для инициализации массива людей
   8. Определить структуру struct DATA для хранения сведений о человеке
   9. Определить структуру struct H как вспомогательную структуру
2. В ходе работы были использованы следующие типы данных
   1. Структура struct DATA для хранения сведений о человеке

struct DATA

{

string Name;

string Surname;

string Middle;

int day;

int month;

int year;

int ind;

int date;

}

* 1. Структура struct H, вспомогательная структура

struct H

{

char symb;

int ind;

};

* 1. В главной функции используется многократно функция switch под «меню», например, под выбор метода поиска

switch (foo)

{

case 1: humanINIT(human, size); break;

case 2: PrintH(human, size); break;

case 3: InitPI(human, size); break;

case 4: InitTable(human, size); break;

case 5: f = true; break;

default:

break;

}

1. Для решения задачи данные были представлены в следующем виде:
   1. Для хранения данных используется массив объектов структуры DATA

struct DATA

{

string Name;

string Surname;

string Middle;

int day;

int month;

int year;

int ind;

int date;

};

1. Для операции ввода и вывода использовались следующие операторы и функции:
   1. Операторы ввода и вывода cin cout, например:

cout << "5: Выйти из программы" << endl;

cin >> foo;

* 1. С помощью фукнции getline()

getline(cin, str);

1. Поставленные задачи будут решены следующими действиями:
   1. Функцией void InitPI(DATA\* human, int size) для инициализации таблицы пи, с помощью которой будет производиться сдвиг указателя на определенное количество символов. Так же эта функция вызывает функцию поиска Кнутта-Морриса-Пратта.

void InitPI(DATA\* human, int size)

{

string str;

cout << "Введите строку по которой будет выполняться поиск: \t";

cin.ignore();

getline(cin, str);

int\* PI = new int[str.length()];

PI[0] = 0;

int j = 0;

int i = 1;

while (i != (str.length()))

{

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

{

PI[i] = j + 1;

i++;

j++;

}

else if (j == 0 && str[i] != str[j])

{

PI[i] = 0;

i++;

}

else if (str[i] != str[j] && j != 0)

{

j = PI[j - 1];

}

}

for (int k = 0; k < str.length(); k++)

{

cout << PI[k] << " ";

}

cout << endl << endl;

KMP(human, size, PI, str);

}

* 1. Сама функция поиска Кнутта-Морриса-Пратта. В которой имя фамилия и отчество складываются в одну строку. Далее мы проходимся по всех этой строке и сверяем ее посимвольно с введенной строкой, при этом инрементируем переменную которая отвечает за номер текущего индекса в строке введенной пользователем. После того как функция выйдет из цикла, она проверяет эту переменную если она не равна 0, то мы получаем число из массива пи на сколько необходимо сдвинуться.

void KMP(DATA\* human, int size, int\* PI, string str)

{

string STR = "";

bool f = false;

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

{

STR = human[i].Surname + " " + human[i].Name + " " + human[i].Middle;

int l = 0;

int k = 0;

while (k < STR.length())

{

while (STR[k] == str[l])

{

k++;

l++;

if (l == str.length())

{

cout << "Образ найден!" << endl << endl;

cout << "ind:\t" << "[" << human[i].ind << "]" << endl;

cout << human[i].Surname << "\n" << human[i].Name << "\n" << human[i].Middle << endl;

cout << human[i].day << "." << human[i].month << "." << human[i].year << endl;

f = true;

}

}

if (l != 0)

{

l = PI[l - 1];

}

else {

k++;

}

}

}

if (f != true)

{

cout << "Образа в строке нет!";

}

}

* 1. Функция для инициализации таблицы, которая понадобиться для поиска Бойера-Мура. В которой производится ввод пользователем искомой строки и добавляем в конец \*, далее мы разворачиваем эту строку находим значения а которые мы будем сдвигать, по определенному алгоритму. Полеченную таблицу мы передаем в функцию Бойера-Мура и вызываем саму эту функцию

void InitTable(DATA\* human, int size)

{

string str;

cout << "Введите строку по которой будет выполняться поиск: \t";

cin.ignore();

getline(cin, str);

str = str + "\*";

string strC = str;

H\* HE = new H[str.length()];

int k = str.length() - 1;

for (int i = 0; i < str.length(); i++)

{

HE[i].symb = str[i];

}

for (int i = 0; i < str.length(); i++)

{

strC[i] = str[k];

k--;

}

int\* Help = new int[(str.length())]; // Массив индексов

Help[0] = strC.length() - 1;

for (int i = 2; i < strC.length(); i++)

{

Help[i] = i - 1;

cout << Help[i] << " ";

}

for (int i = 2; i < strC.length() - 1; i++)

{

int tmp = strC[i]; // переменная, хранящая символ в цикле

for (int j = i + 1; j < strC.length(); j++)

{

if (tmp == strC[j])

{

Help[j] = Help[i];

}

}

}

int y = 1;

bool F = false;

Help[y] = strC.size() - 1; // Присваиваем изначально первому элементу длинну строки

y++;

while (y < strC.length() && F == false)

{

if (strC[1] == strC[y])

{

Help[1] = Help[y];

F = true;

}

y++;

}

cout << endl;

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

{

cout << Help[i] << " ";

}

cout << endl;

BM(human, strC, Help, str, size, HE);

cout << endl;

delete[] HE;

}

* 1. Непосредственно сама функция Бойера-Мура. Мы проходимся по всем элементам массива, который хранит данные о людях. Так же создаем переменную, которая будет хранить ФИО в одной строке. Далее мы проходимся по полученной строке, с конца, и будем сравнивать ее с нашей таблицей. Датее происходит сдвиг в соответствии с таблицей.

void BM(DATA\* human, string strC, int\* Help, string str, int size, H\* HE)

{

bool F = false;

string STR = "";

bool f1 = false;

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

{

string STR = human[t].Surname + " " + human[t].Name + " " + human[t].Middle;

int k = str.length() - 2;

char sy;

int shift = k;

bool f3 = false;

while (k < STR.length() && f3 == false)

{

int i = k;// указатель индекса искомой строки

sy = STR[k];

int j = str.length() - 2;

while (str[j] == STR[i] && j > 0)

{

i--;

j--;

if (str[0] == STR[i])

{

f3 = true;

}

}

shift = SearchInTable(sy, strC, Help); // сдвиг указателя искомой строки

k = k + shift;

if (f3 == true)

{

cout << endl << endl << "ind:\t" << human[t].ind;

cout << endl << human[t].Surname << "\n" << human[t].Name << "\n" << human[t].Middle << endl;

F = true;

}

}

}

if (F == false)

{

cout << endl << endl << "Элементы не найдены!" << endl << endl;

}

}

* 1. Функция для поиска по таблице для выполнения поиска Бойера-Мура.

int SearchInTable(char sy, string str, int\* Help)

{

for (int i = 1; i < str.length(); i++)

{

if (sy == str[i])

{

return Help[i];

}

}

return Help[0];

}

* 1. Функция печати всего массива людей

void PrintH(DATA\* human, int size)

{

cout << endl;

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

{

cout << i << endl;

cout << "ind: " << "[" << human[i].ind << "]" << endl;

cout << human[i].Surname << endl;

cout << human[i].Name << endl;

cout << human[i].Middle << endl;

cout << human[i].day << "." << human[i].month << "." << human[i].year << endl << endl;

}

}

* 1. Функция для инициализации массива объектов структур DATA

void humanINIT(DATA\* human, int size)

{

string name[] = { "александр", "алексей", "анатолий", "андрей", "антон", "аркадий", "арсений", "артём", "артур", "борис", "вадим", "максим", "валерий", "василий", "виктор", "виталий", "владимир", "владислав", "вячеслав", "георгий", "глеб", "григорий", "даниил", "денис", "дмитрий", "двгений", "дгор", "иван", "игорь", "илья", "кирилл", "константин", "лев", "леонид", "максим", "марк", "матвей", "михаил", "никита", "николай", "олег", "павел", "пётр", "роман", "руслан", "сергей", "степан", "тимур", "фёдор", "юрий", "ярослав" };

string surname[] = { "пантелеев", "чувашев", "галинов", "ананина", "масылюк", "вострокнутова", "сабуров", "исламов", "шамай", "филатов", "сафронов", "шарпов", "карелов", "брейкин", "копытов", "солдатов", "кузнецов", "пажгин", "фотин", "бадретдинов", "механошин", "булдаков", "тулинов", "тедеев", "колпаков", "черных", "нефедов", "рябцев", "пепеляев", "тарасов", "аркадьев" };

string middle[] = { "романович", "иосифович", "аникитевич", "прохорович", "архипович", "александрович", "мартьянович", "богданович", "никифорович", "георгиевич" , "прокофиевич" , "валериевич" , "владиславович" , "данилевич" , "ульянович" , "кондратиевич" , "валериевич" , "сократович" , "кондратович" , "двграфович" , "алексеевич" , "кондратович" , "тихонович" , "игнатиевич" , "алексеевич" , "венедиктович" , "мартьянович" , "родионович" , "тимурович" , "самуилович" , "климентович" , "натанович" , "леонович" , "игоревич" , "захарович" , "андронович" , "платонович" , "несторович" , "модестович" , "дрофеевич" , "адамович" };

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

{

human[i].ind = i;

human[i].year = 1903 + rand() % 117;

human[i].month = 1 + rand() % 12;

int a = human[i].month;

switch (a)

{

case 1: human[i].day = 1 + (rand() % 31); break;

case 2: human[i].day = 1 + (rand() % 28); break;

case 3: human[i].day = 1 + (rand() % 31); break;

case 4: human[i].day = 1 + (rand() % 30); break;

case 5: human[i].day = 1 + (rand() % 31); break;

case 6: human[i].day = 1 + (rand() % 30); break;

case 7: human[i].day = 1 + (rand() % 31); break;

case 8: human[i].day = 1 + (rand() % 31); break;

case 9: human[i].day = 1 + (rand() % 30); break;

case 10: human[i].day = 1 + (rand() % 31); break;

case 11: human[i].day = 1 + (rand() % 30); break;

case 12: human[i].day = 1 + (rand() % 31); break;

default:

break;

}

int g = rand() % (sizeof(name) / sizeof(name[0]));

human[i].Name = name[g];

g = rand() % (sizeof(surname) / sizeof(surname[0]));

human[i].Surname = surname[g];

g = rand() % (sizeof(middle) / sizeof(middle[0]));

human[i].Middle = middle[g];

human[i].date = human[i].day + human[i].month \* 100 + human[i].year \* 10000;

}

}

* 1. Функция выбора bool Choice(DATA\* human, int size, int foo, bool f)

bool Choice(DATA\* human, int size, int foo, bool f)

{

switch (foo)

{

case 1: humanINIT(human, size); break;

case 2: PrintH(human, size); break;

case 3: InitPI(human, size); break;

case 4: InitTable(human, size); break;

case 5: f = true; break;

default:

break;

}

return f;

}

* 1. Главная функция main

int main()

{

system("chcp 1251>nul");

srand(time(NULL));

int size;

size = 100;

DATA\* human = new DATA[size];

bool f = false;

int foo = -1;

while (f == false)

{

cout << endl;

cout << "Выберете действие:" << endl;

cout << "1: Инициализировать массив" << endl;

cout << "2: Напечатать массив в консоль" << endl;

cout << "3: Алгоритм поиска КМП" << endl;

cout << "4: Алгоритм поиска Бойера-Мура" << endl;

cout << "5: Выйти из программы" << endl;

cin >> foo;

f = Choice(human, size, foo, f);

}

}

**Блок-схема**























**Код**

#include <iostream>

#include <string>

#include <list>

#include <ctime>

#include <fstream>

using namespace std;

struct DATA

{

string Name;

string Surname;

string Middle;

int day;

int month;

int year;

int ind;

int date;

};

struct H

{

char symb;

int ind;

};

DATA\* ptr;

// инициализация объекта структуры

void humanINIT(DATA\* human, int size)

{

string name[] = { "александр", "алексей", "анатолий", "андрей", "антон", "аркадий", "арсений", "артём", "артур", "борис", "вадим", "максим", "валерий", "василий", "виктор", "виталий", "владимир", "владислав", "вячеслав", "георгий", "глеб", "григорий", "даниил", "денис", "дмитрий", "двгений", "дгор", "иван", "игорь", "илья", "кирилл", "константин", "лев", "леонид", "максим", "марк", "матвей", "михаил", "никита", "николай", "олег", "павел", "пётр", "роман", "руслан", "сергей", "степан", "тимур", "фёдор", "юрий", "ярослав" };

string surname[] = { "пантелеев", "чувашев", "галинов", "ананина", "масылюк", "вострокнутова", "сабуров", "исламов", "шамай", "филатов", "сафронов", "шарпов", "карелов", "брейкин", "копытов", "солдатов", "кузнецов", "пажгин", "фотин", "бадретдинов", "механошин", "булдаков", "тулинов", "тедеев", "колпаков", "черных", "нефедов", "рябцев", "пепеляев", "тарасов", "аркадьев" };

string middle[] = { "романович", "иосифович", "аникитевич", "прохорович", "архипович", "александрович", "мартьянович", "богданович", "никифорович", "георгиевич" , "прокофиевич" , "валериевич" , "владиславович" , "данилевич" , "ульянович" , "кондратиевич" , "валериевич" , "сократович" , "кондратович" , "двграфович" , "алексеевич" , "кондратович" , "тихонович" , "игнатиевич" , "алексеевич" , "венедиктович" , "мартьянович" , "родионович" , "тимурович" , "самуилович" , "климентович" , "натанович" , "леонович" , "игоревич" , "захарович" , "андронович" , "платонович" , "несторович" , "модестович" , "дрофеевич" , "адамович" };

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

{

human[i].ind = i;

human[i].year = 1903 + rand() % 117;

human[i].month = 1 + rand() % 12;

int a = human[i].month;

switch (a)

{

case 1: human[i].day = 1 + (rand() % 31); break;

case 2: human[i].day = 1 + (rand() % 28); break;

case 3: human[i].day = 1 + (rand() % 31); break;

case 4: human[i].day = 1 + (rand() % 30); break;

case 5: human[i].day = 1 + (rand() % 31); break;

case 6: human[i].day = 1 + (rand() % 30); break;

case 7: human[i].day = 1 + (rand() % 31); break;

case 8: human[i].day = 1 + (rand() % 31); break;

case 9: human[i].day = 1 + (rand() % 30); break;

case 10: human[i].day = 1 + (rand() % 31); break;

case 11: human[i].day = 1 + (rand() % 30); break;

case 12: human[i].day = 1 + (rand() % 31); break;

default:

break;

}

int g = rand() % (sizeof(name) / sizeof(name[0]));

human[i].Name = name[g];

g = rand() % (sizeof(surname) / sizeof(surname[0]));

human[i].Surname = surname[g];

g = rand() % (sizeof(middle) / sizeof(middle[0]));

human[i].Middle = middle[g];

human[i].date = human[i].day + human[i].month \* 100 + human[i].year \* 10000;

}

}

// печать массива

void PrintH(DATA\* human, int size)

{

cout << endl;

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

{

cout << i << endl;

cout << "ind: " << "[" << human[i].ind << "]" << endl;

cout << human[i].Surname << endl;

cout << human[i].Name << endl;

cout << human[i].Middle << endl;

cout << human[i].day << "." << human[i].month << "." << human[i].year << endl << endl;

}

}

// поиск кнутта мориса прата

void KMP(DATA\* human, int size, int\* PI, string str)

{

string STR = "";

bool f = false;

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

{

STR = human[i].Surname + " " + human[i].Name + " " + human[i].Middle;

int l = 0;

int k = 0;

while (k < STR.length())

{

while (STR[k] == str[l])

{

k++;

l++;

if (l == str.length())

{

cout << "Образ найден!" << endl << endl;

cout << "ind:\t" << "[" << human[i].ind << "]" << endl;

cout << human[i].Surname << "\n" << human[i].Name << "\n" << human[i].Middle << endl;

cout << human[i].day << "." << human[i].month << "." << human[i].year << endl;

f = true;

}

}

if (l != 0)

{

l = PI[l - 1];

}

else {

k++;

}

}

}

if (f != true)

{

cout << "Образа в строке нет!";

}

}

// инициализация пи массива

void InitPI(DATA\* human, int size)

{

string str;

cout << "Введите строку по которой будет выполняться поиск: \t";

cin.ignore();

getline(cin, str);

int\* PI = new int[str.length()];

PI[0] = 0;

int j = 0;

int i = 1;

while (i != (str.length()))

{

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

{

PI[i] = j + 1;

i++;

j++;

}

else if (j == 0 && str[i] != str[j])

{

PI[i] = 0;

i++;

}

else if (str[i] != str[j] && j != 0)

{

j = PI[j - 1];

}

}

for (int k = 0; k < str.length(); k++)

{

cout << PI[k] << " ";

}

cout << endl << endl;

KMP(human, size, PI, str);

}

// поиск по таблице

int SearchInTable(char sy, string str, int\* Help)

{

for (int i = 1; i < str.length(); i++)

{

if (sy == str[i])

{

return Help[i];

}

}

return Help[0];

}

// поиск боера мура

void BM(DATA\* human, string strC, int\* Help, string str, int size, H\* HE)

{

bool F = false;

string STR = "";

bool f1 = false;

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

{

string STR = human[t].Surname + " " + human[t].Name + " " + human[t].Middle;

int k = str.length() - 2;

char sy;

int shift = k;

bool f3 = false;

while (k < STR.length() && f3 == false)

{

int i = k;// указатель индекса искомой строки

sy = STR[k];

int j = str.length() - 2;

while (str[j] == STR[i] && j > 0)

{

i--;

j--;

if (str[0] == STR[i])

{

f3 = true;

}

}

shift = SearchInTable(sy, strC, Help); // сдвиг указателя искомой строки

k = k + shift;

if (f3 == true)

{

cout << endl << endl << "ind:\t" << human[t].ind;

cout << endl << human[t].Surname << "\n" << human[t].Name << "\n" << human[t].Middle << endl;

F = true;

}

}

}

if (F == false)

{

cout << endl << endl << "Элементы не найдены!" << endl << endl;

}

}

// создание таблицы

void InitTable(DATA\* human, int size)

{

string str;

cout << "Введите строку по которой будет выполняться поиск: \t";

cin.ignore();

getline(cin, str);

str = str + "\*";

string strC = str;

H\* HE = new H[str.length()];

int k = str.length() - 1;

for (int i = 0; i < str.length(); i++)

{

HE[i].symb = str[i];

}

for (int i = 0; i < str.length(); i++)

{

strC[i] = str[k];

k--;

}

int\* Help = new int[(str.length())]; // Массив индексов

Help[0] = strC.length() - 1;

for (int i = 2; i < strC.length(); i++)

{

Help[i] = i - 1;

cout << Help[i] << " ";

}

for (int i = 2; i < strC.length() - 1; i++)

{

int tmp = strC[i]; // переменная, хранящая символ в цикле

for (int j = i + 1; j < strC.length(); j++)

{

if (tmp == strC[j])

{

Help[j] = Help[i];

}

}

}

int y = 1;

bool F = false;

Help[y] = strC.size() - 1; // Присваиваем изначально первому элементу длинну строки

y++;

while (y < strC.length() && F == false)

{

if (strC[1] == strC[y])

{

Help[1] = Help[y];

F = true;

}

y++;

}

cout << endl;

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

{

cout << Help[i] << " ";

}

cout << endl;

BM(human, strC, Help, str, size, HE);

cout << endl;

delete[] HE;

}

// меню

bool Choice(DATA\* human, int size, int foo, bool f)

{

switch (foo)

{

case 1: humanINIT(human, size); break;

case 2: PrintH(human, size); break;

case 3: InitPI(human, size); break;

case 4: InitTable(human, size); break;

case 5: f = true; break;

default:

break;

}

return f;

}

int main()

{

system("chcp 1251>nul");

srand(time(NULL));

int size;

size = 100;

DATA\* human = new DATA[size];

bool f = false;

int foo = -1;

while (f == false)

{

cout << endl;

cout << "Выберете действие:" << endl;

cout << "1: Инициализировать массив" << endl;

cout << "2: Напечатать массив в консоль" << endl;

cout << "3: Алгоритм поиска КМП" << endl;

cout << "4: Алгоритм поиска Бойера-Мура" << endl;

cout << "5: Выйти из программы" << endl;

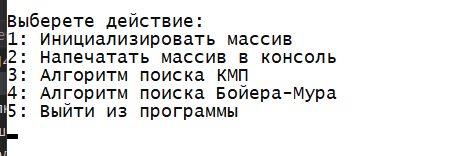
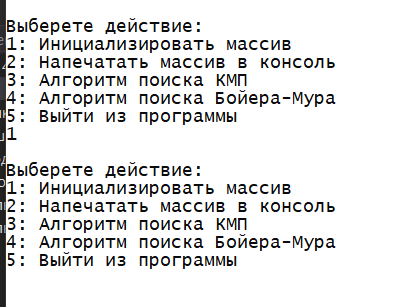
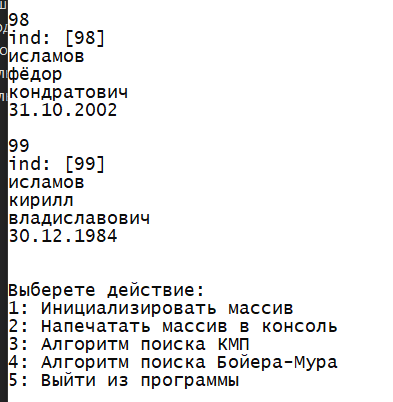
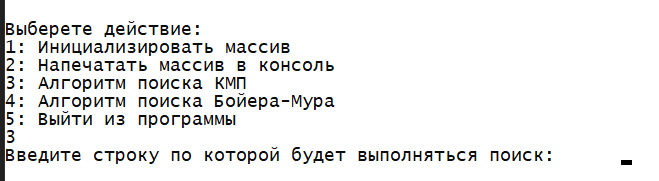
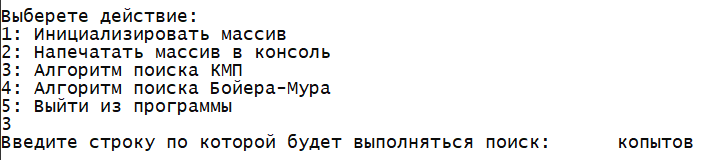
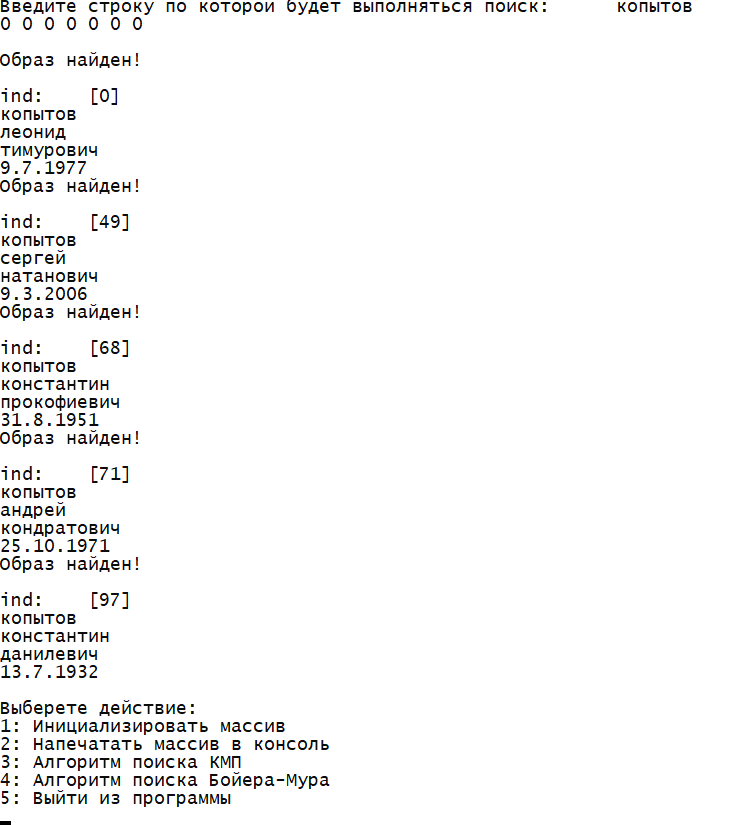
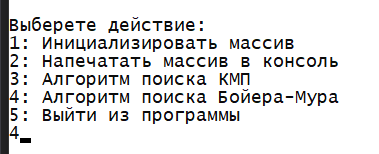
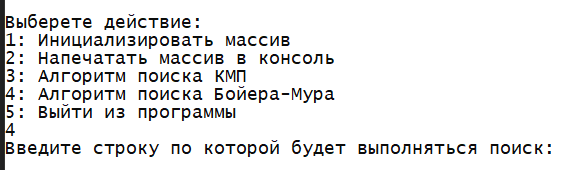
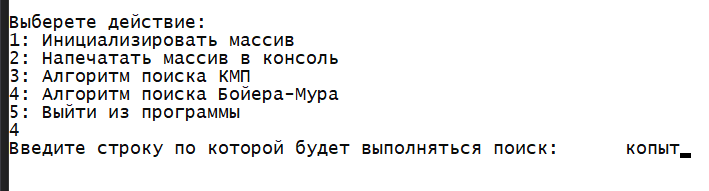
cin >> foo;

f = Choice(human, size, foo, f);

}

}

**Скриншоты**

1. **Кнутта-Морриса-Пратта**
   1. 
   2. **Инициализируем массив**
      1. 
   3. **Напечатаем массив в консоль**
      1. 
   4. **Поиск Кнутта-Морриса-Пратта**
      1. 
   5. **Вводим строку по которой будет выполнятся поиск**
      1. 
      2. 
2. **Поиск Бойера-Мура**
   1. 
   2. 
   3. **Вводим искомую строку**
      1. 
      2. 