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 теории алгоритмов и структуры данных»

Тема: Методы поиска: метод Кнутта-Мориса Пратта, метод Бойера-Мура

Вариант 16

Выполнил:

Студент группы РИС-20-1б

Ознобихин Елисей Андреевич

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

**Цель работы**

1. Получить практические навыки работы с методами поиска данных: методы Бойера-Мура, Кнута, Морриса, Пратта и прямого поиска подстроки в строке.

**Постановка задачи**

(Вариант 16)

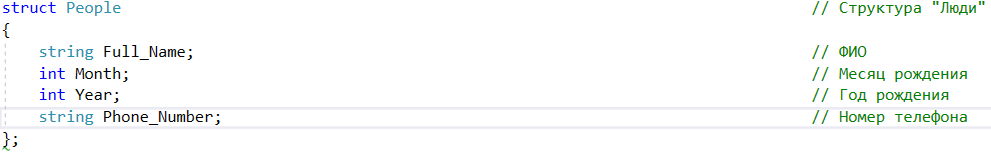
Сформировать множество по варианту. Провести поиск элемента множества по заданному ключу, используя методы Бойера-Мура, Кнута, Морриса, Пратта и прямого поиска подстроки в строке.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\opera_nwVIAxYut4.png

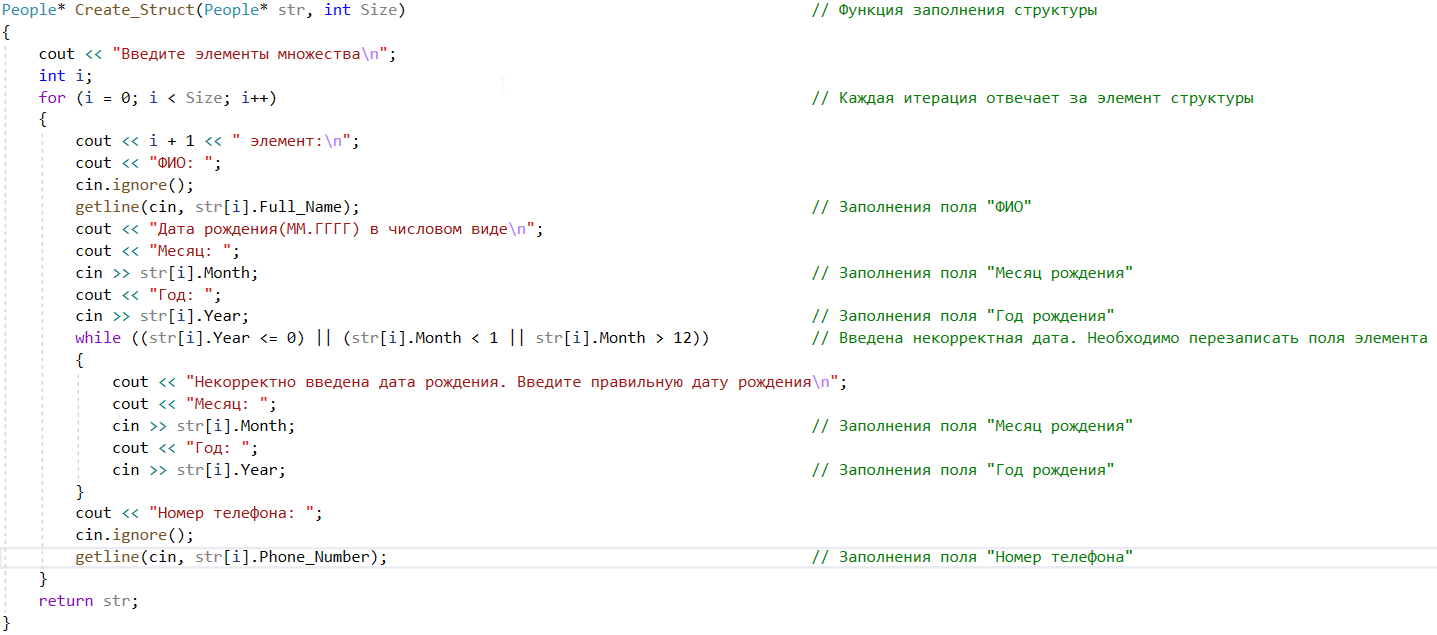
C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\opera_w32hrSK0fC.png

**Анализ задачи**

1. Для решения задачи необходимо:
   1. Сформировать структуру, в которой будут храниться данные об элементах множества.



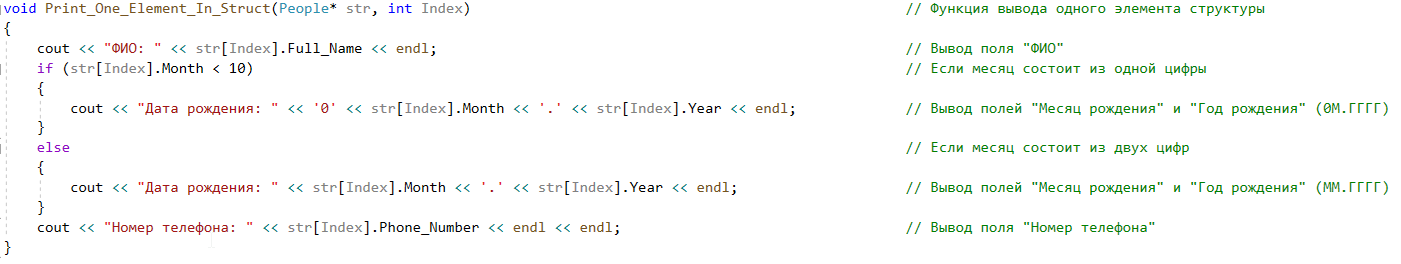
* 1. Разработать функцию записи элементов множества в структуру.



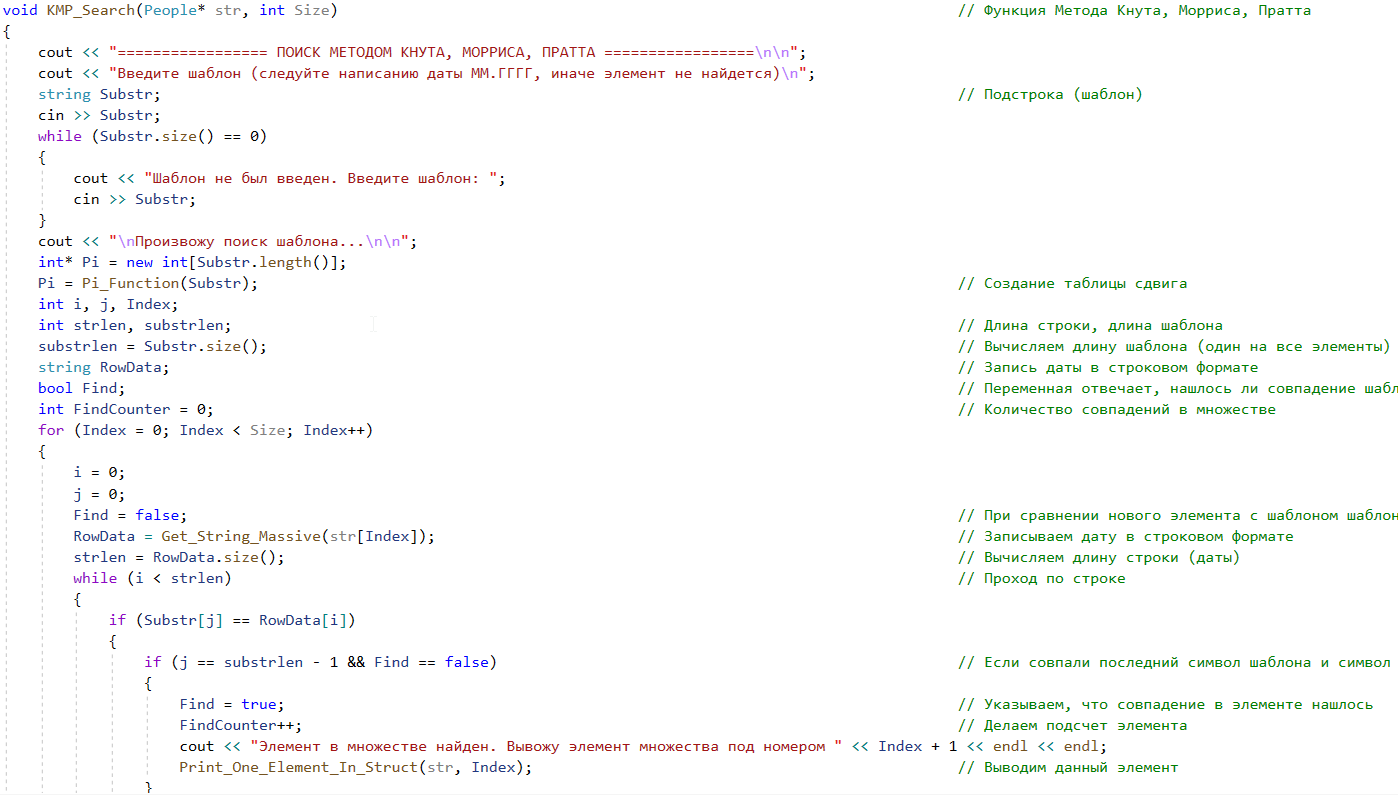
* 1. Разработать функцию для вывода элементов множества.



* 1. Разработать функцию для вывода одного элемента из множества.



* 1. Разработать функцию для реализации Поиска Методом Кнута, Морриса, Пратта.





* 1. Разработать функцию для формирования префикс-функции, или массива сдвига.



* 1. Разработать функцию для преобразования даты рождения в число.

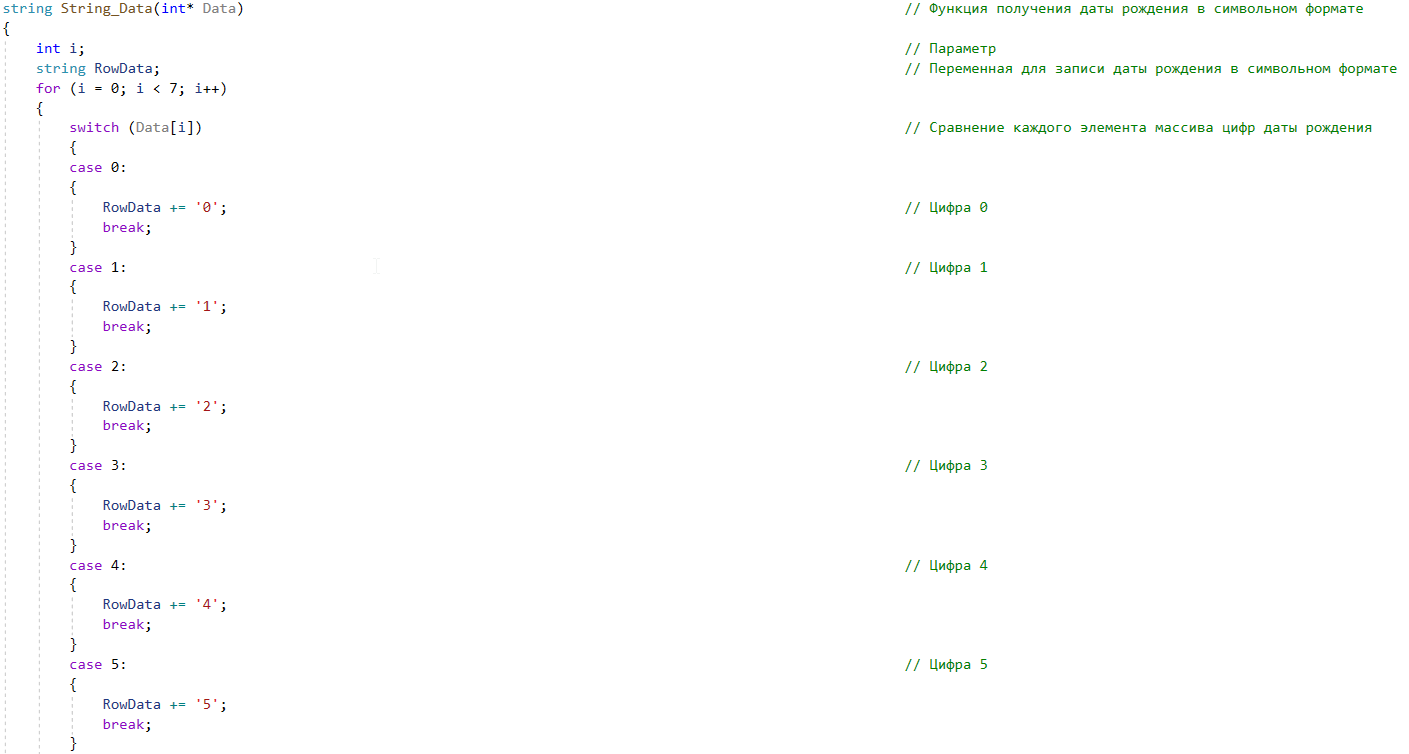


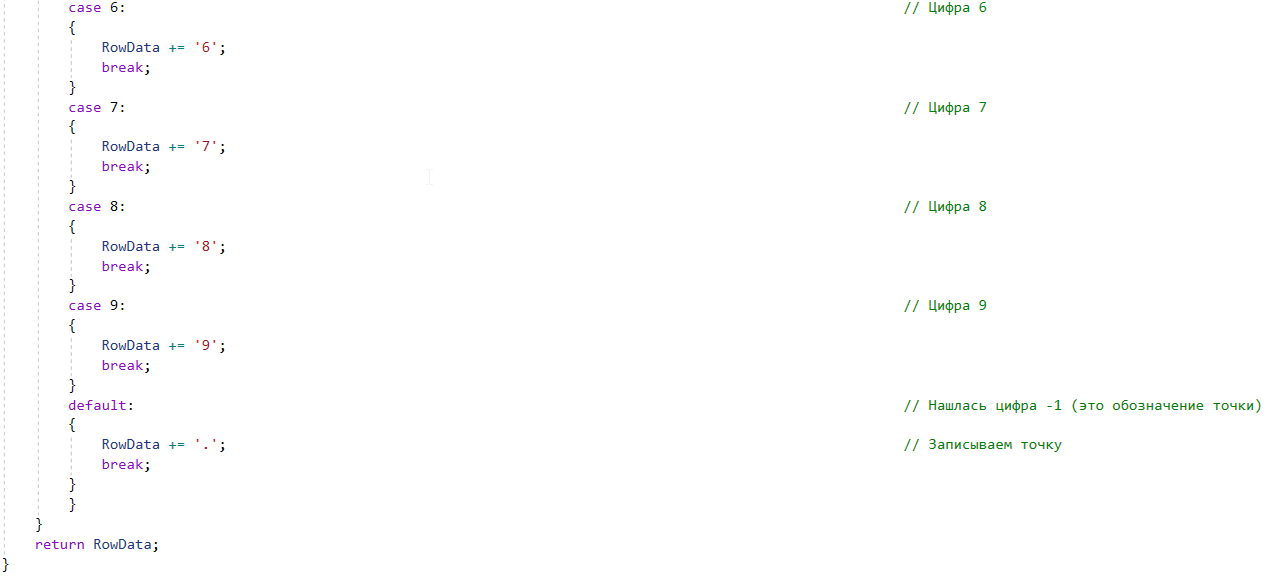
* 1. Разработать функцию для реализации Поиска Методом Бойера-Мура.



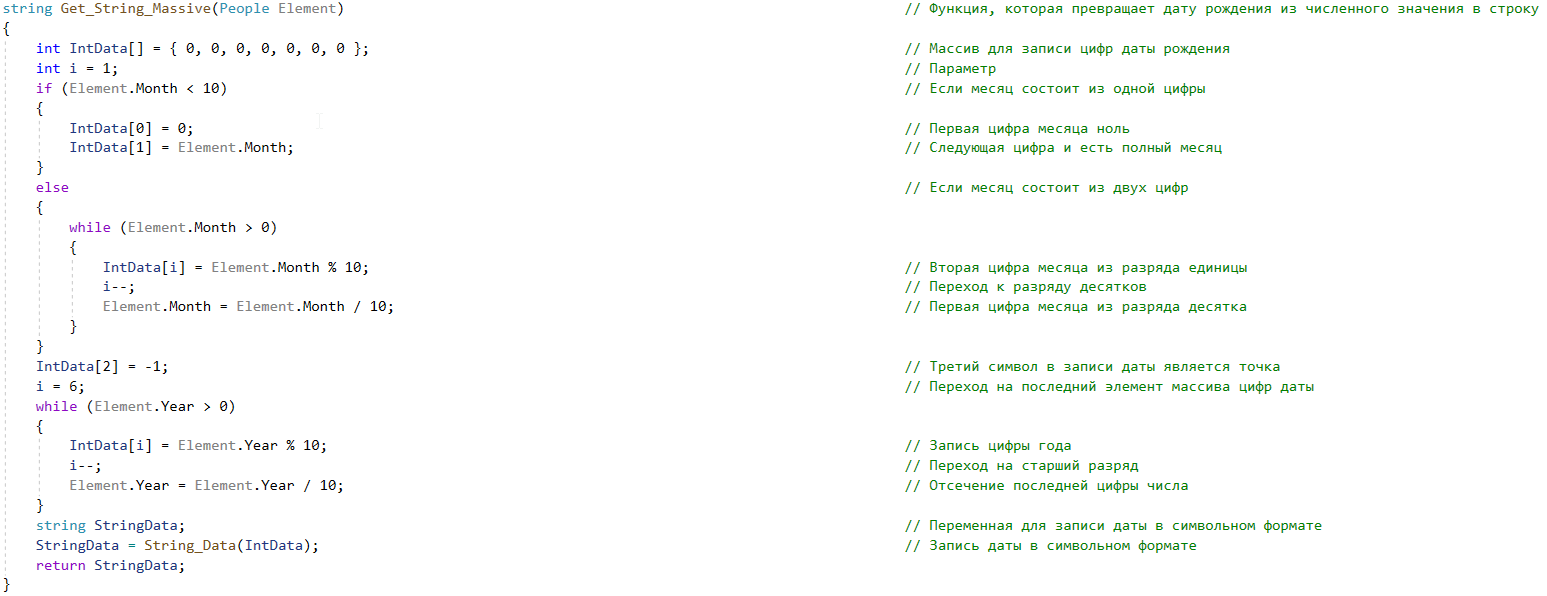


* 1. Разработать функцию для преобразования одной даты рождения в строку символов.





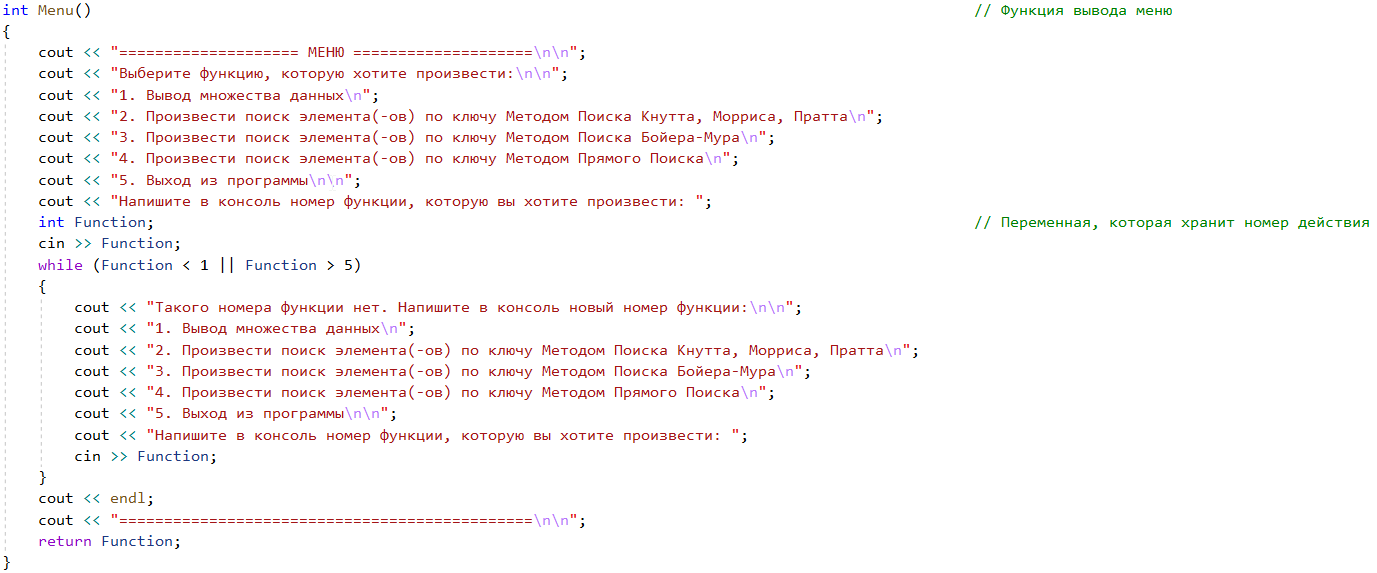
* 1. Разработать функцию для преобразования каждой даты рождения в строку символов.



* 1. Разработать функцию для реализации Поиска Методом Прямого Поиска Подстроки в Строке.



* 1. Разработать функцию для реализации меню.



* 1. Реализовать применение этих функций в главной функции.



1. В ходе работы были использованы типы данных:
   1. Для функции Create\_Struct используются следующие аргументы:
   2. Указатель на People: указатель на первый элемент пустого множества.
   3. Тип int: размерность множества.

Сама функция имеет тип указателя на People и возвращает заполненное множество.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_ZgNFOpIdxE.png

* 1. Для функции Print\_One\_Element\_In\_Struct используются следующие аргументы:
  2. Указатель на People: указатель на первый элемент множества.
  3. Тип int: индекс элемента множества.

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

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_sljV9FZoQ3.png

* 1. Для функции Print\_Struct используются следующие аргументы:
  2. Указатель на People: указатель на первый элемент множества.
  3. Тип int: размерность множества.

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

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_x4xveYtj5d.png

* 1. Для функции KMP\_Search используются следующие аргументы:
  2. Указатель на People: указатель на первый элемент множества.
  3. Тип int: размерность множества.

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

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_Ws0BKU9dpM.png

* 1. Для функции Get\_Data\_In\_Number используются следующие аргументы:

1. Указатель на People: указатель на первый элемент множества.
2. Тип int: размерность множества.

Сама функция имеет тип указателя на long int и возвращает массив дат рождения в числовом формате.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_DtePGdMjrA.png

* 1. Для функции Pi\_Function используются следующие аргументы:

1. Тип string: подстрока, или шаблон.

Сама функция имеет тип указателя на int и возвращает массив сдвига.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_UimeiGWoaT.png

* 1. Для функции BM\_Search используются следующие аргументы:
  2. Указатель на People: указатель на первый элемент множества.
  3. Тип int: размерность множества.

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

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_LqiDSM94n7.png

* 1. Для функции String\_Data используются следующие аргументы:

1. Указатель на int: массив состоящих из даты цифр.

Сама функция имеет тип string и возвращает полученную дату-строку.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_xBd2o59lew.png

* 1. Для функции Get\_String\_Massive используются следующие аргументы:

1. Тип People: элемент множества.

Сама функция имеет тип string и возвращает дату рождения в символьном формате.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_dUmmi9iqCI.png

* 1. Для функции Substring\_Search используются следующие аргументы:

1. Указатель на People: указатель на первый элемент множества.
2. Тип int: размерность множества.

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

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_RDM2xMgaoV.png

* 1. Для функции Menu не используются аргументы.

Сама функция имеет тип int и возвращает номер выполняемого действия.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_VlJTjBFb12.png

1. Для решения задачи данные были представлены в следующем виде:
   1. Для работы данные были записаны в виде структуры с несколькими полями.

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_dEeVYKTCPs.png

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

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_HYVPXHmrnH.png

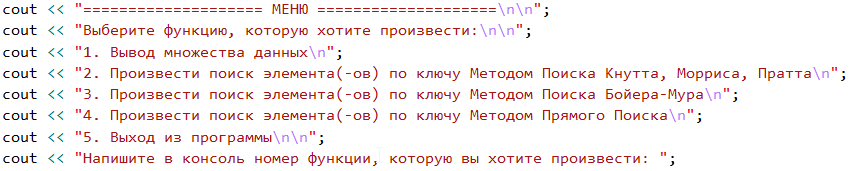
C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_LBe33s5OO4.png

* 1. Ввод строковых данных через консоль реализован с помощью функции getline.

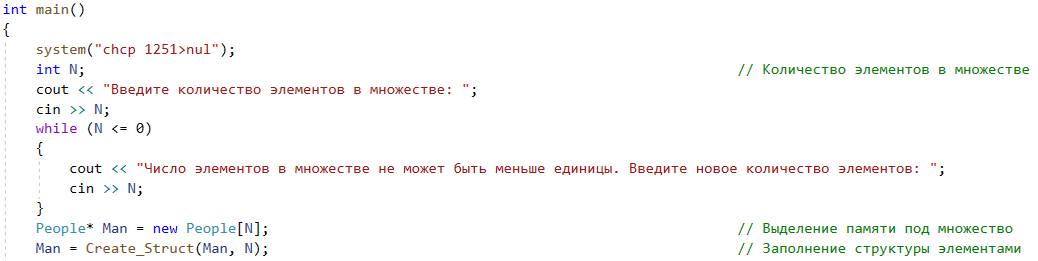
C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_0wVFCrVvLd.png

C:\Users\Асус\Documents\ShareX\Screenshots\2021-02\devenv_LzNEMMjAcK.png

* 1. Вывод данных на консоль реализован с помощью оператора cout.



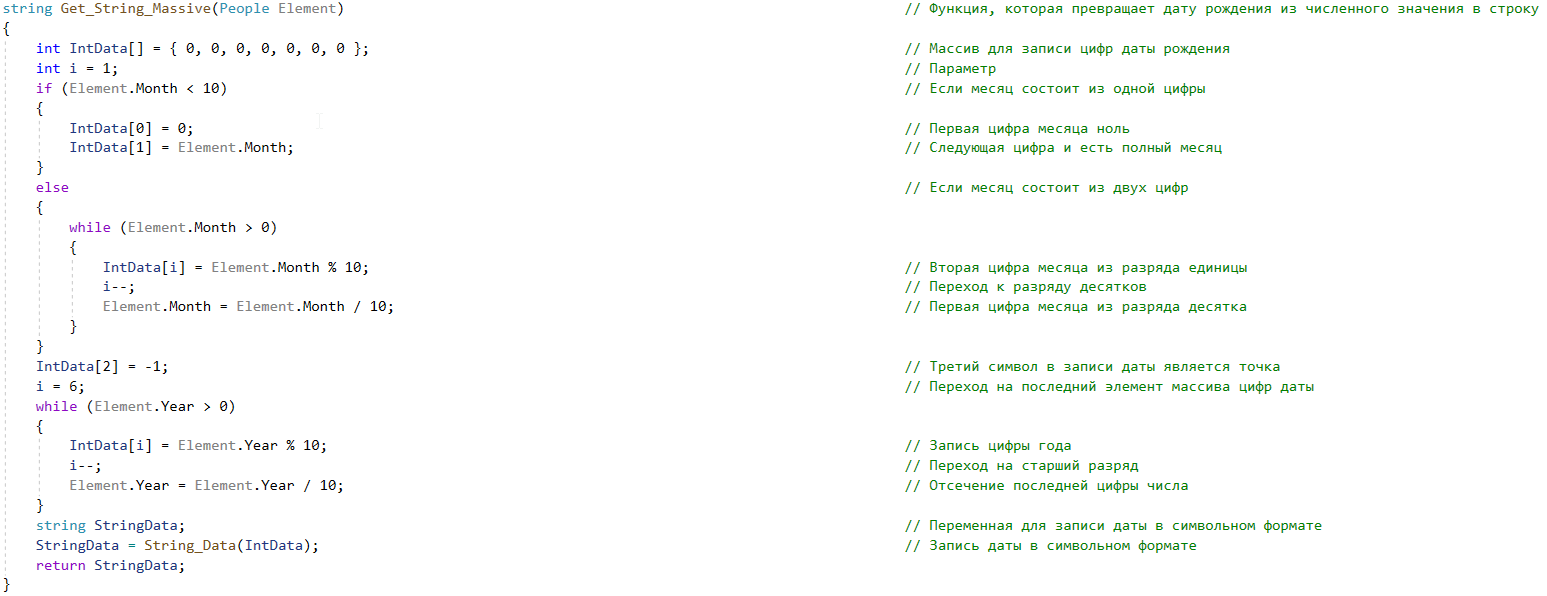
1. Поставленные задачи решены следующими действиями:
   1. Для начала работы программы множество обязательно заполняется, чтобы исключить проверки на пустое множество.

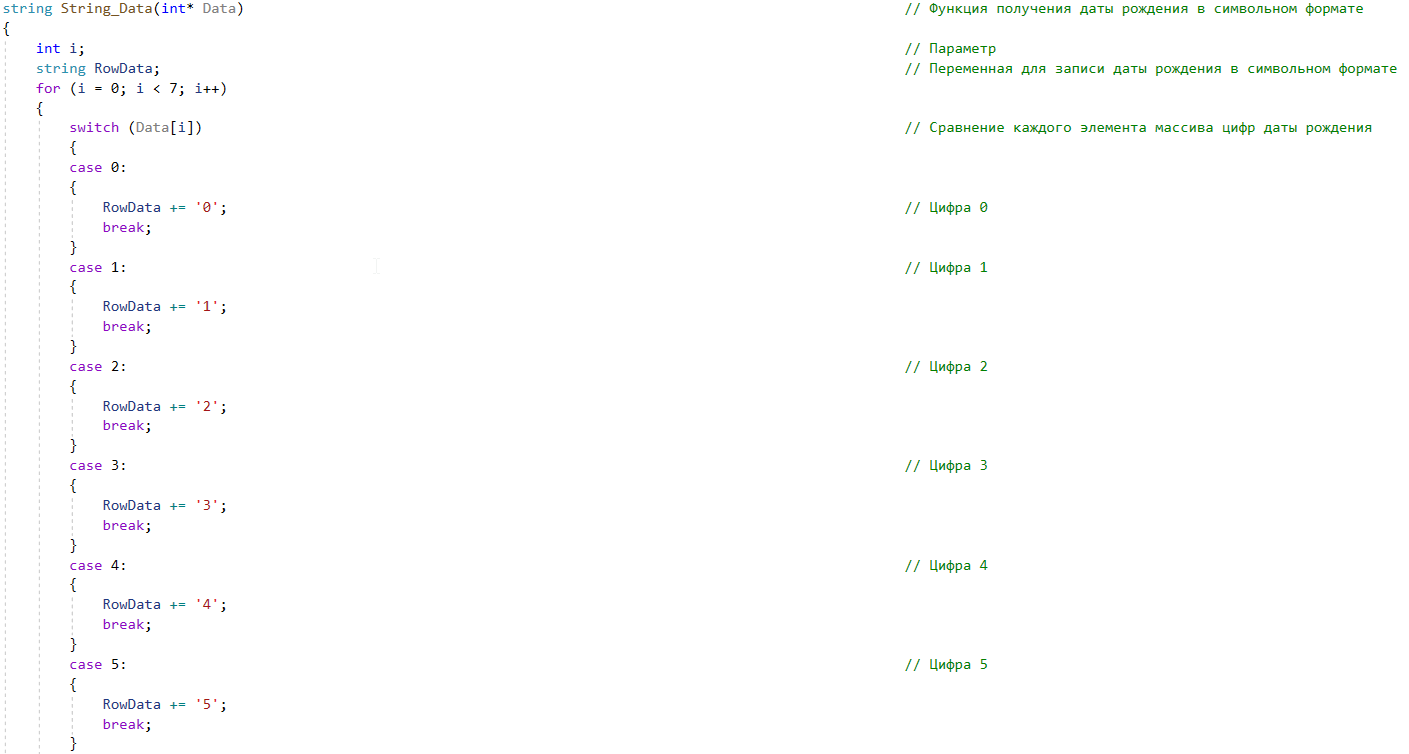


* 1. Для реализации Метода поиска Бойера-Мура использовались формирование массив сдвига для каждого символа из кодировки ASCII и функция преобразования дат рождения в строку символов, поскольку данный метод используется для строкового типа данных.

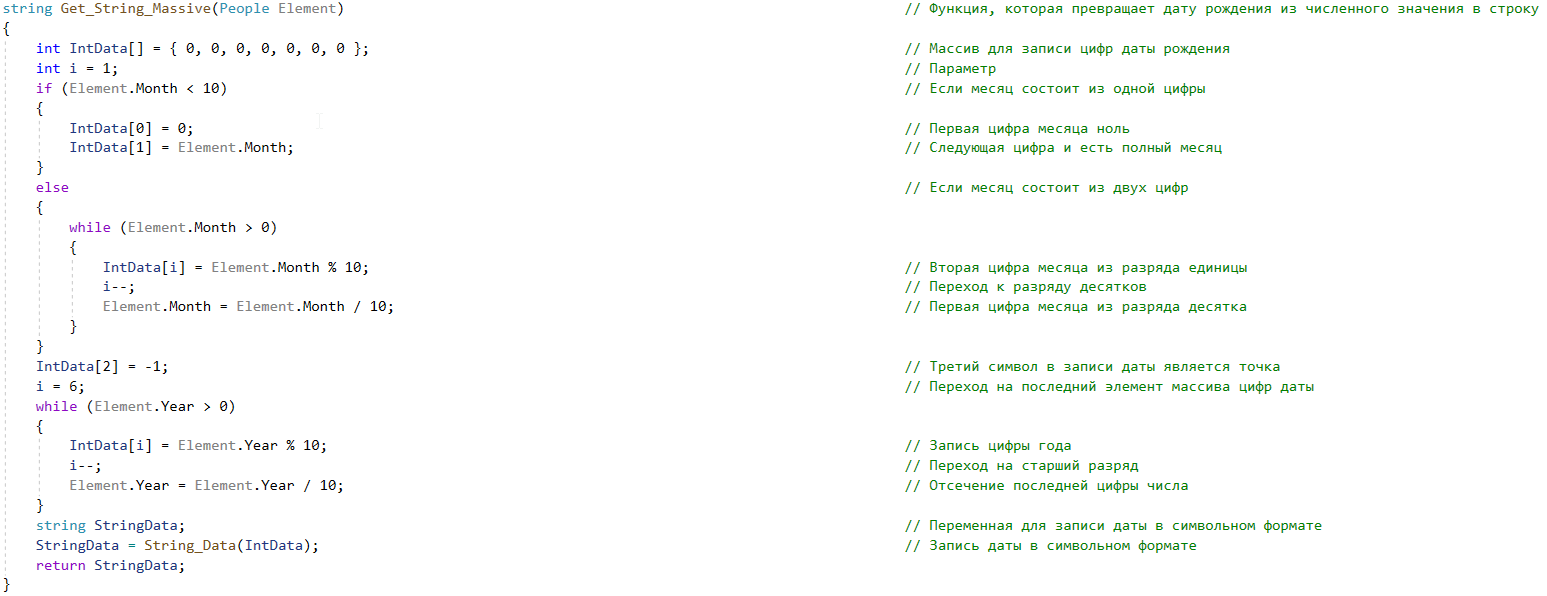


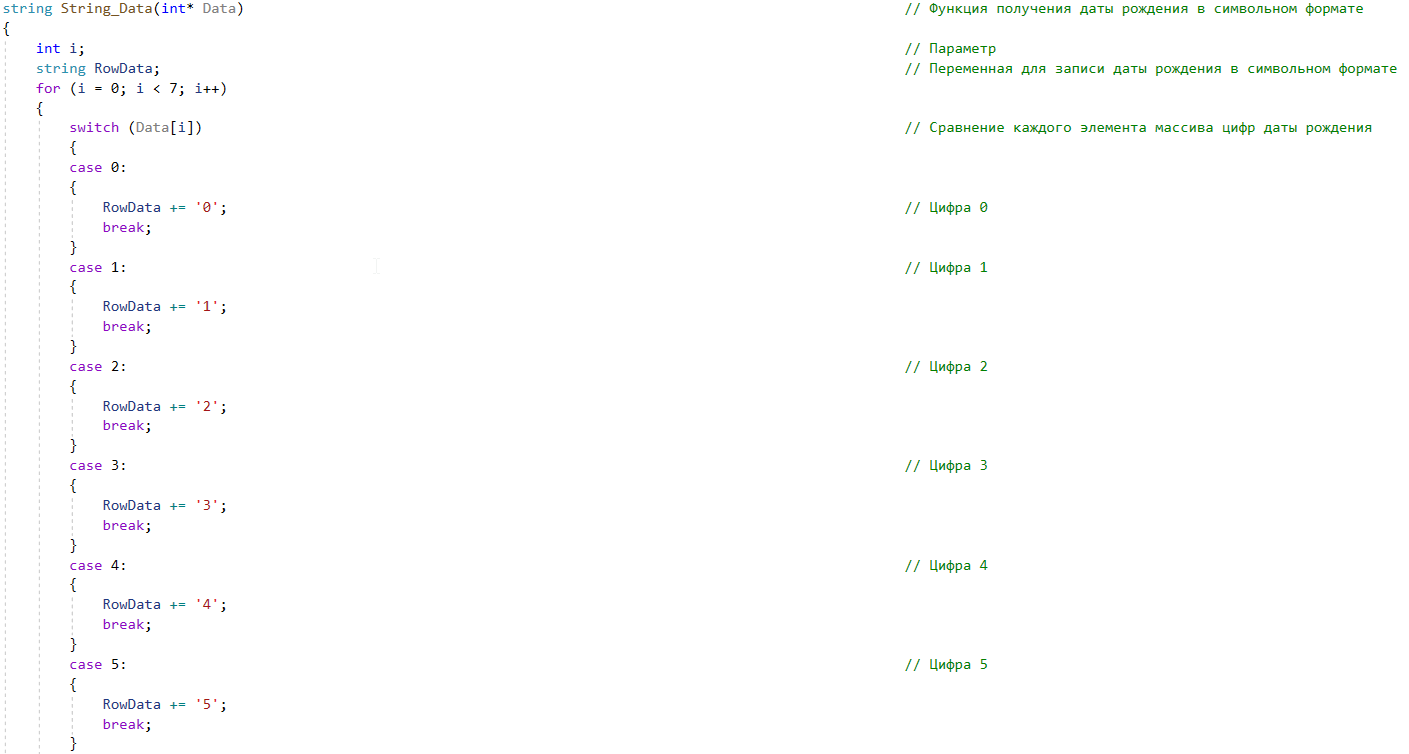






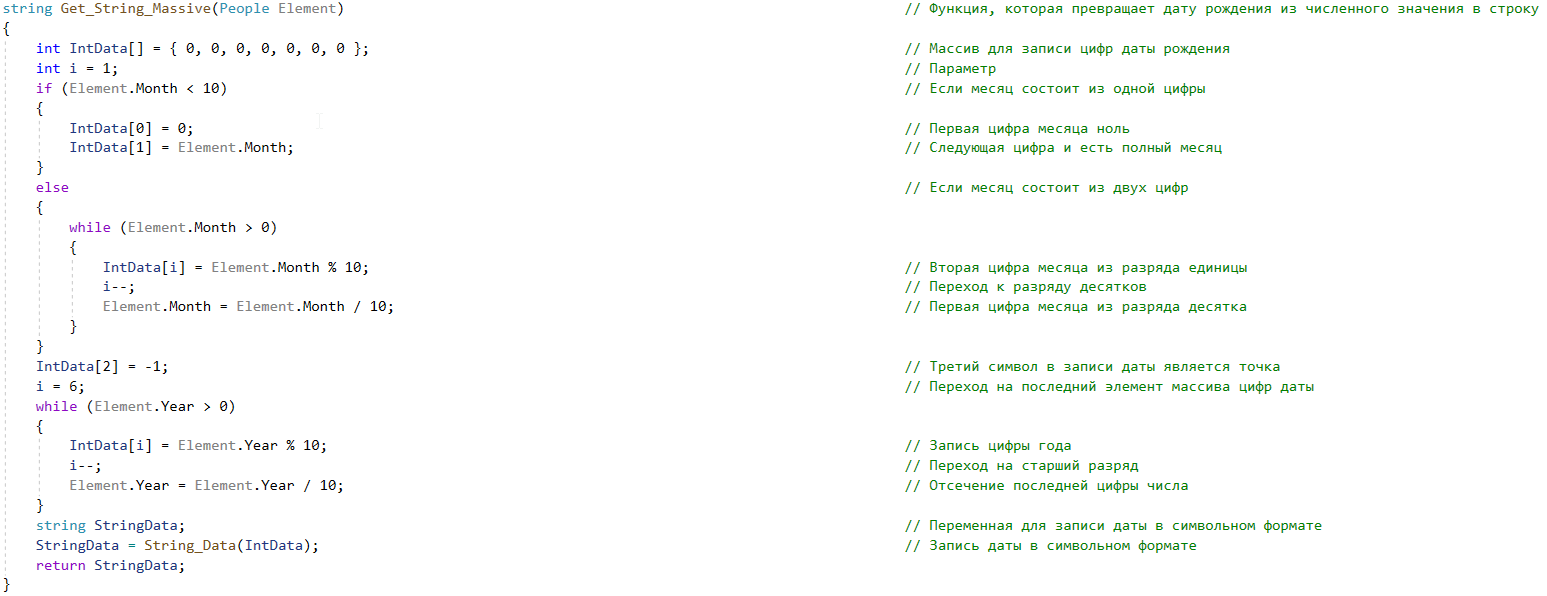
* 1. Для упрощения работы с методами использовались функции получения всех дат рождения в символьном формате и преобразования даты рождения из числа в строку символов.

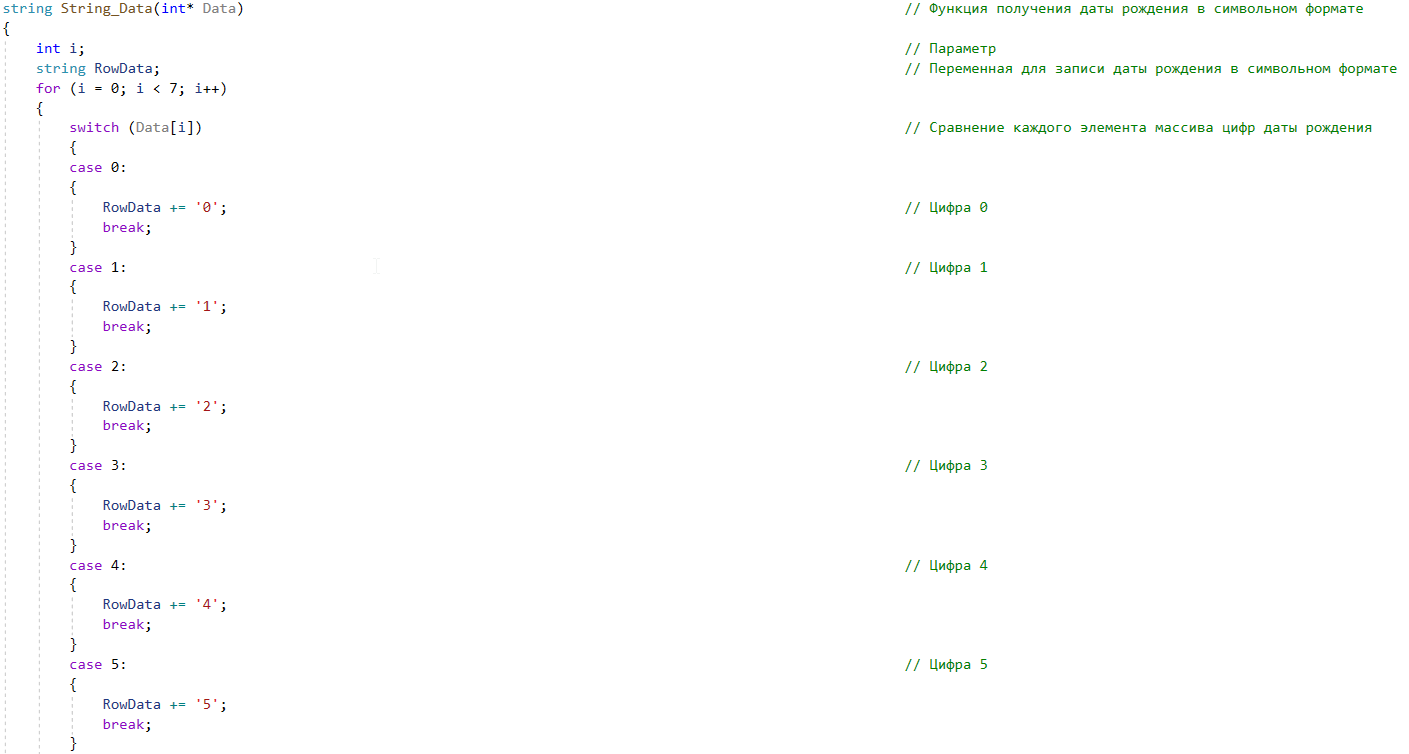




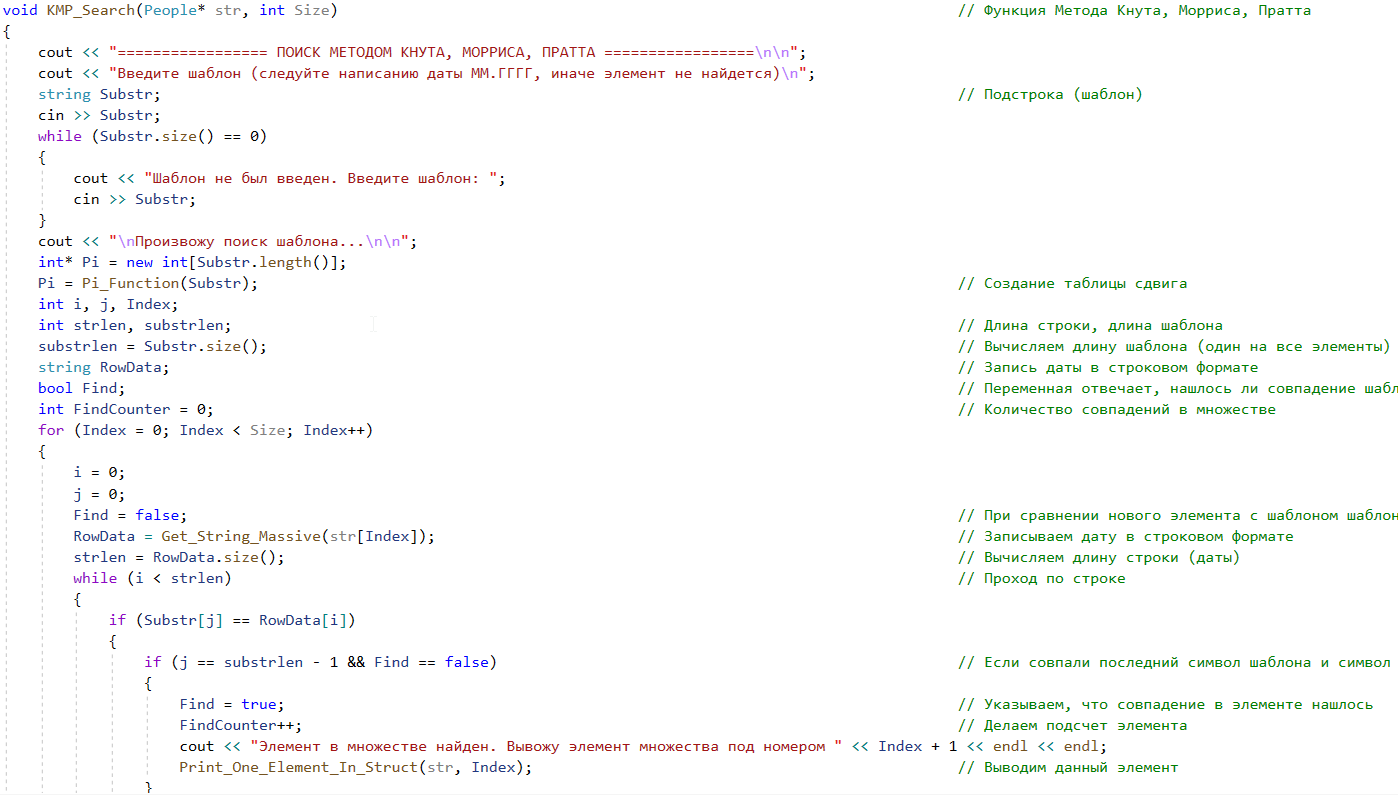
* 1. Для реализации Метода Прямого Поиска Подстроки в Строке использовалась функции преобразования дат рождения в строку символов, поскольку данный метод используется для строкового типа данных.



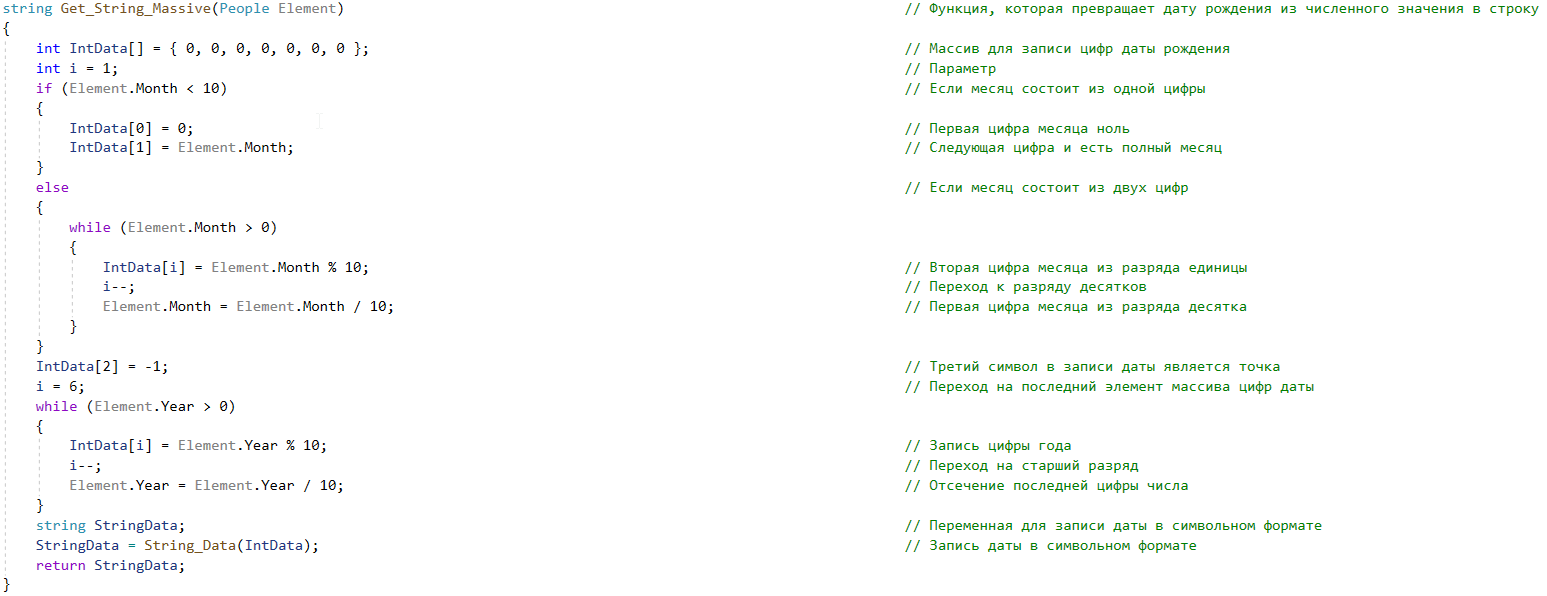


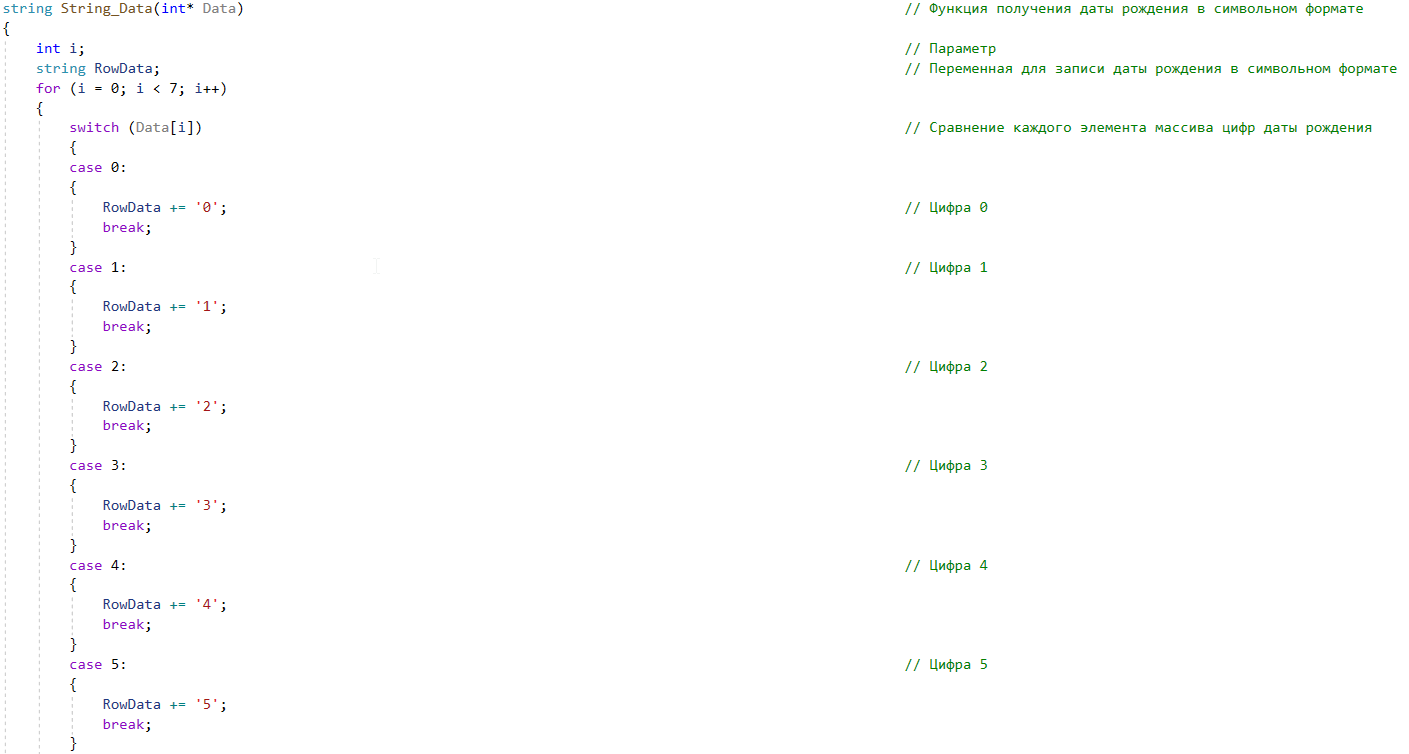


* 1. Для реализации Метода Кнута, Морриса, Пратта использовались функции формирования префикс-функции (или массива сдвига) и преобразования дат рождения в строку символов, поскольку данный метод используется для строкового типа данных.

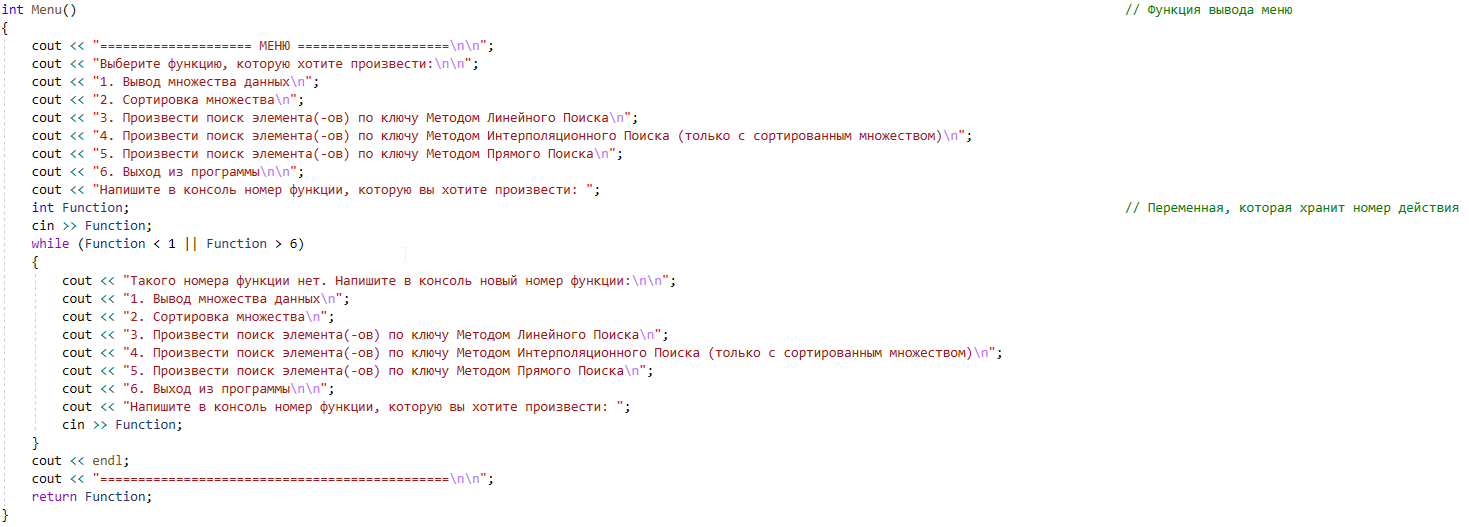




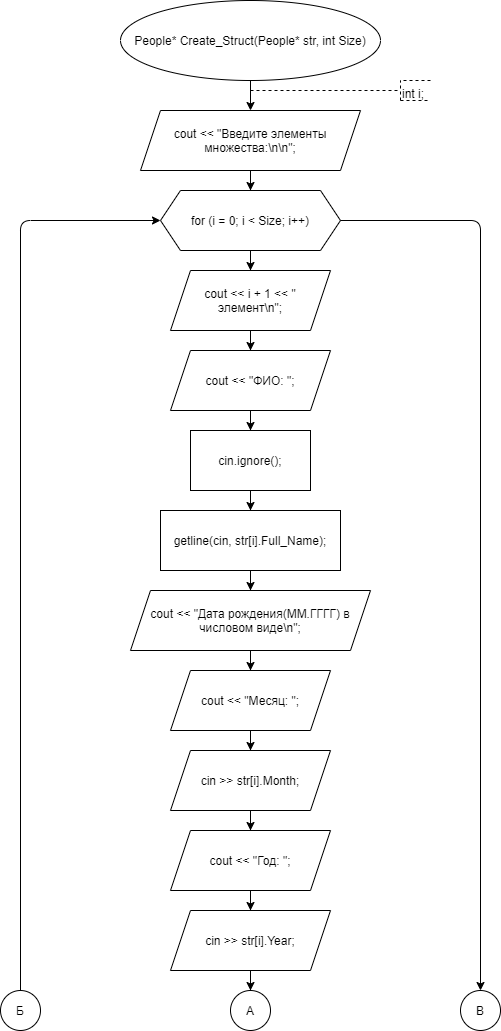


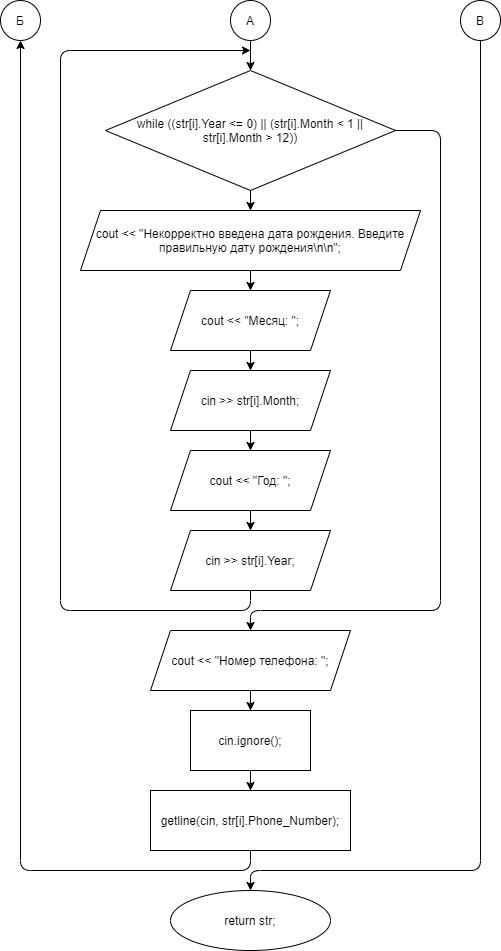


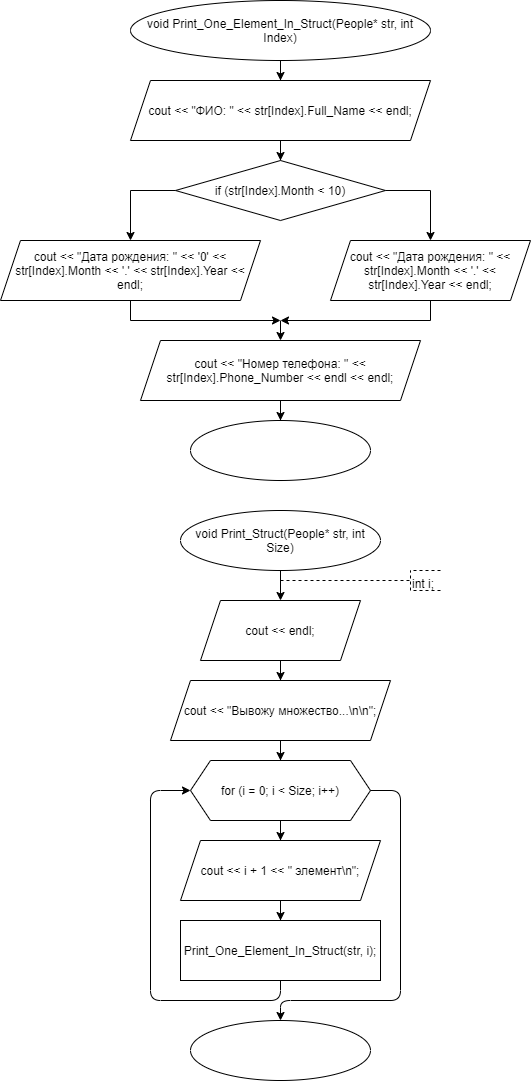
* 1. Для работы с меню от пользователя требуется ввести номер соответствующей функции, которую он желает произвести. Номера функций соответсвуют порядковому номеру, выводимому на консоль.

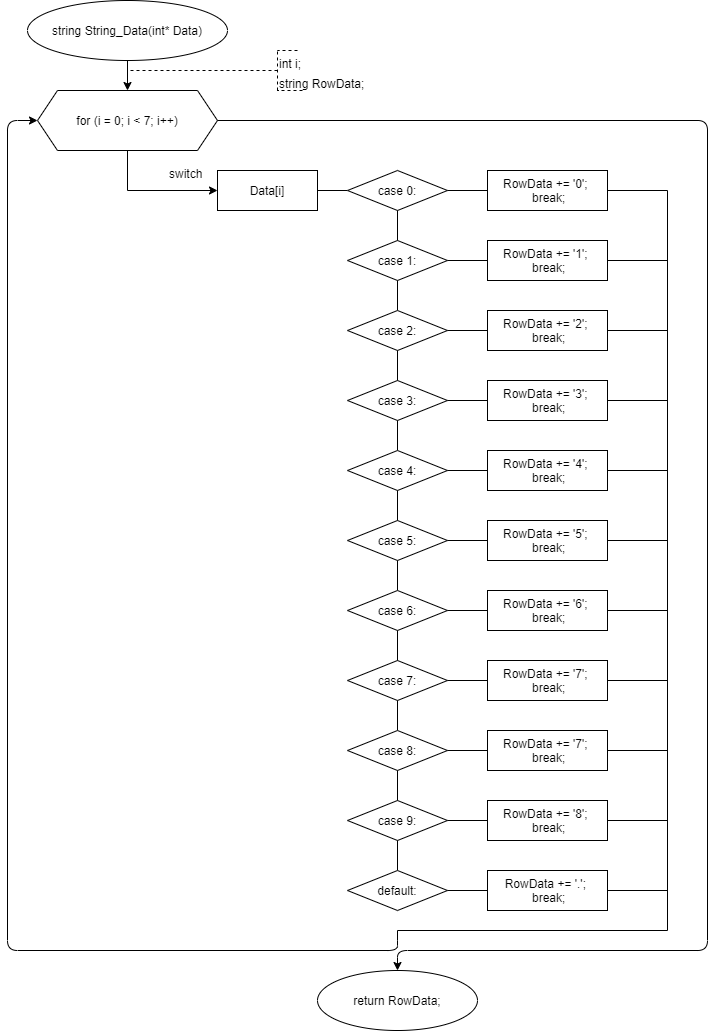


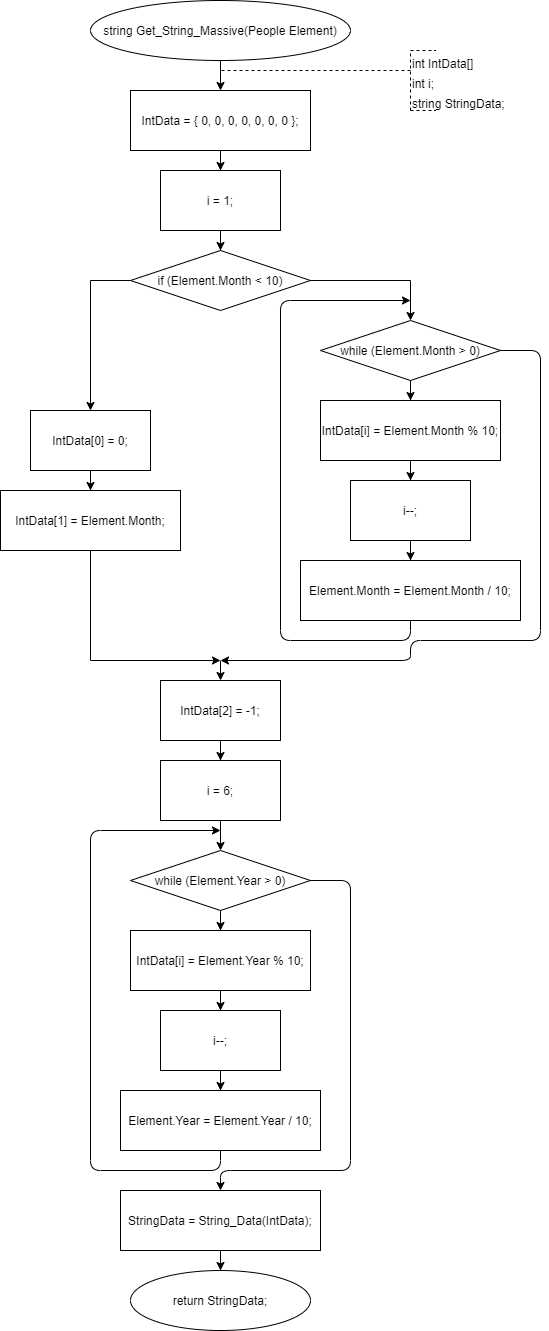
**Блок-схема программы**

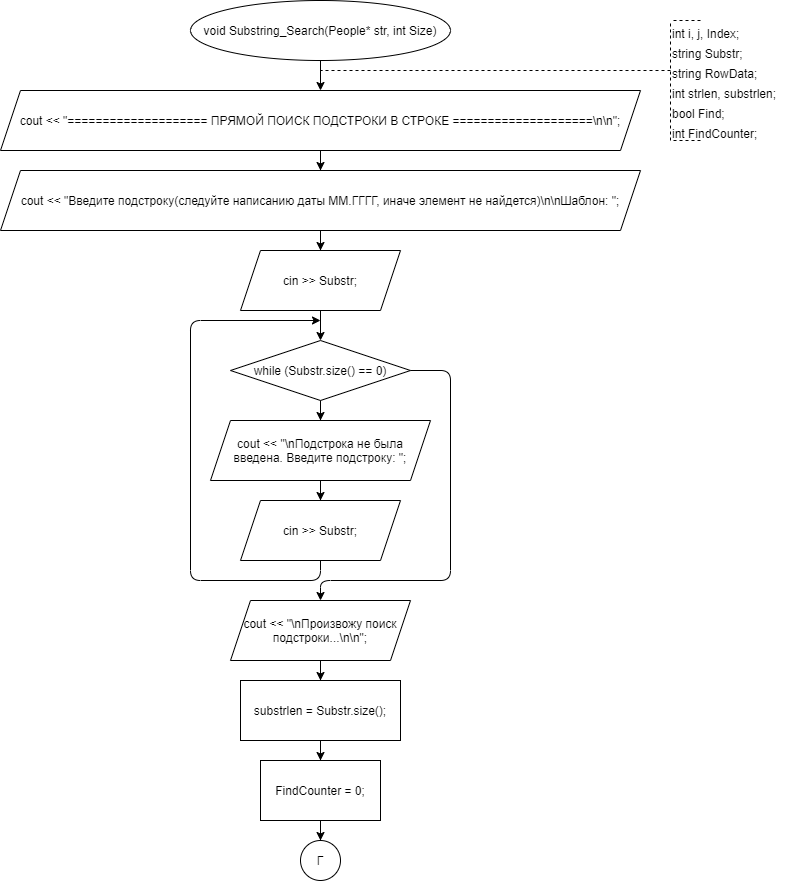


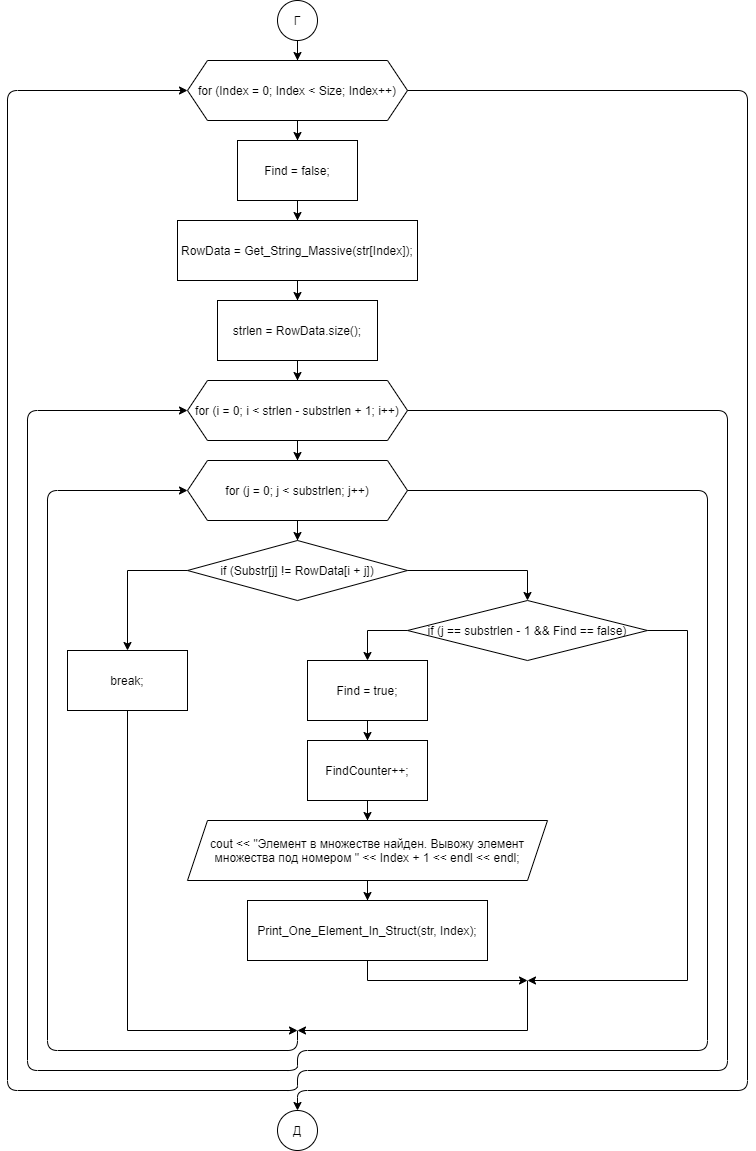


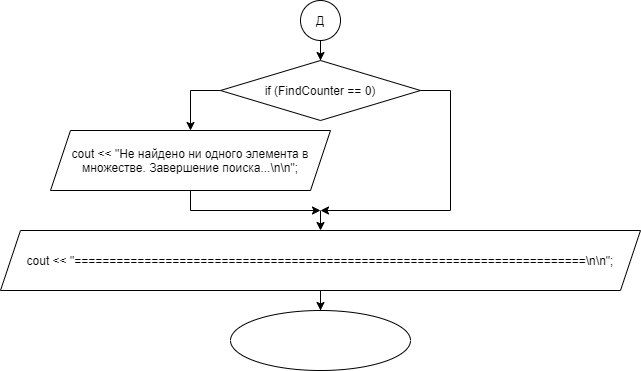


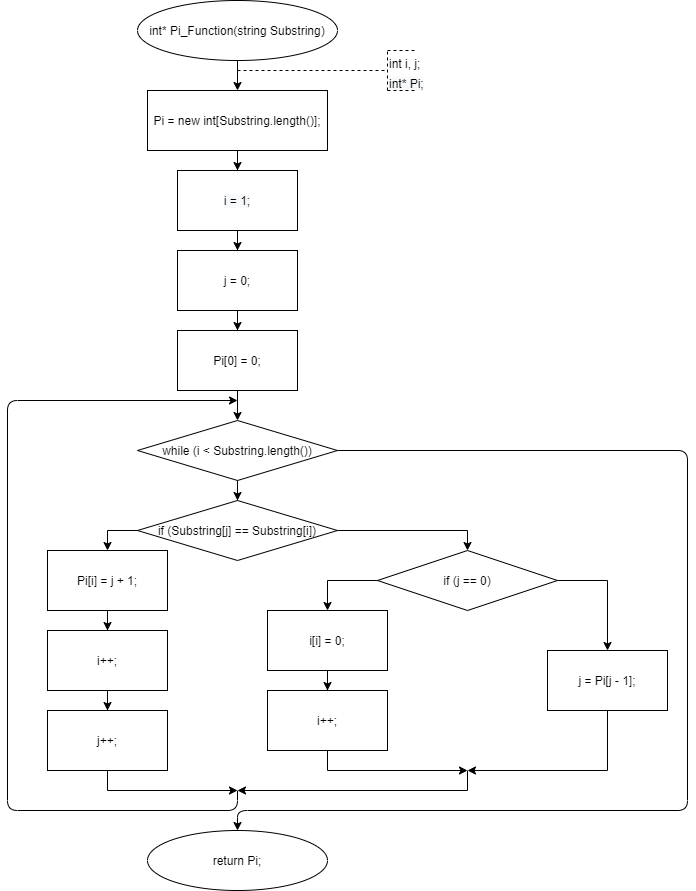


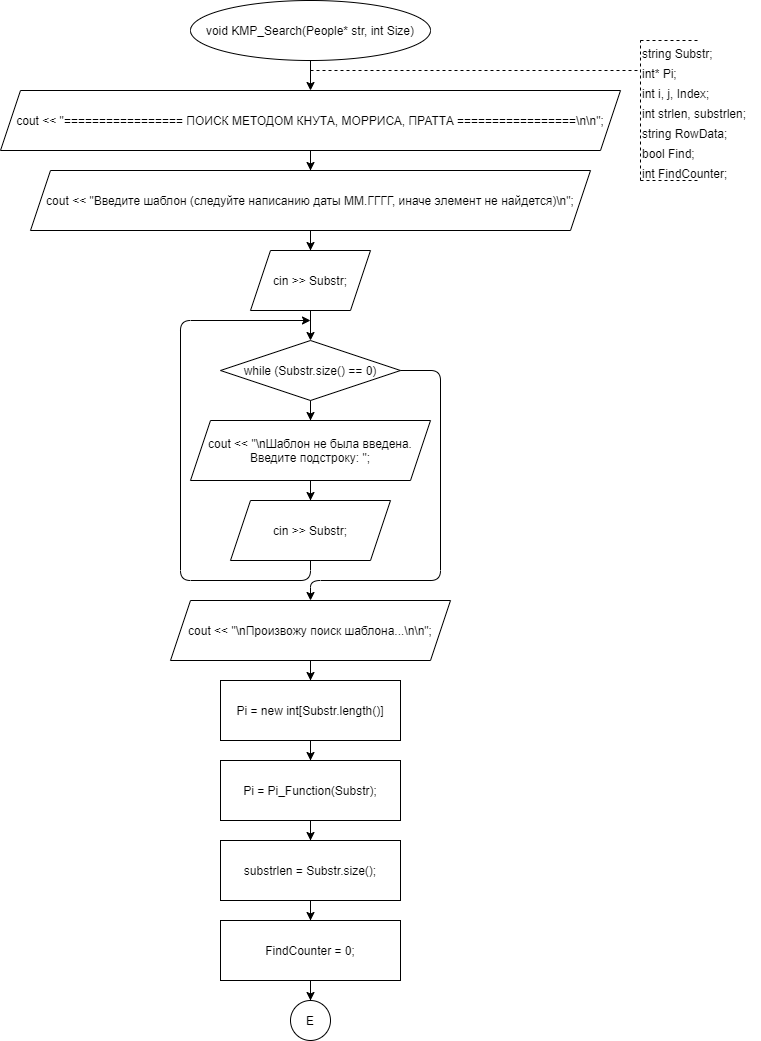


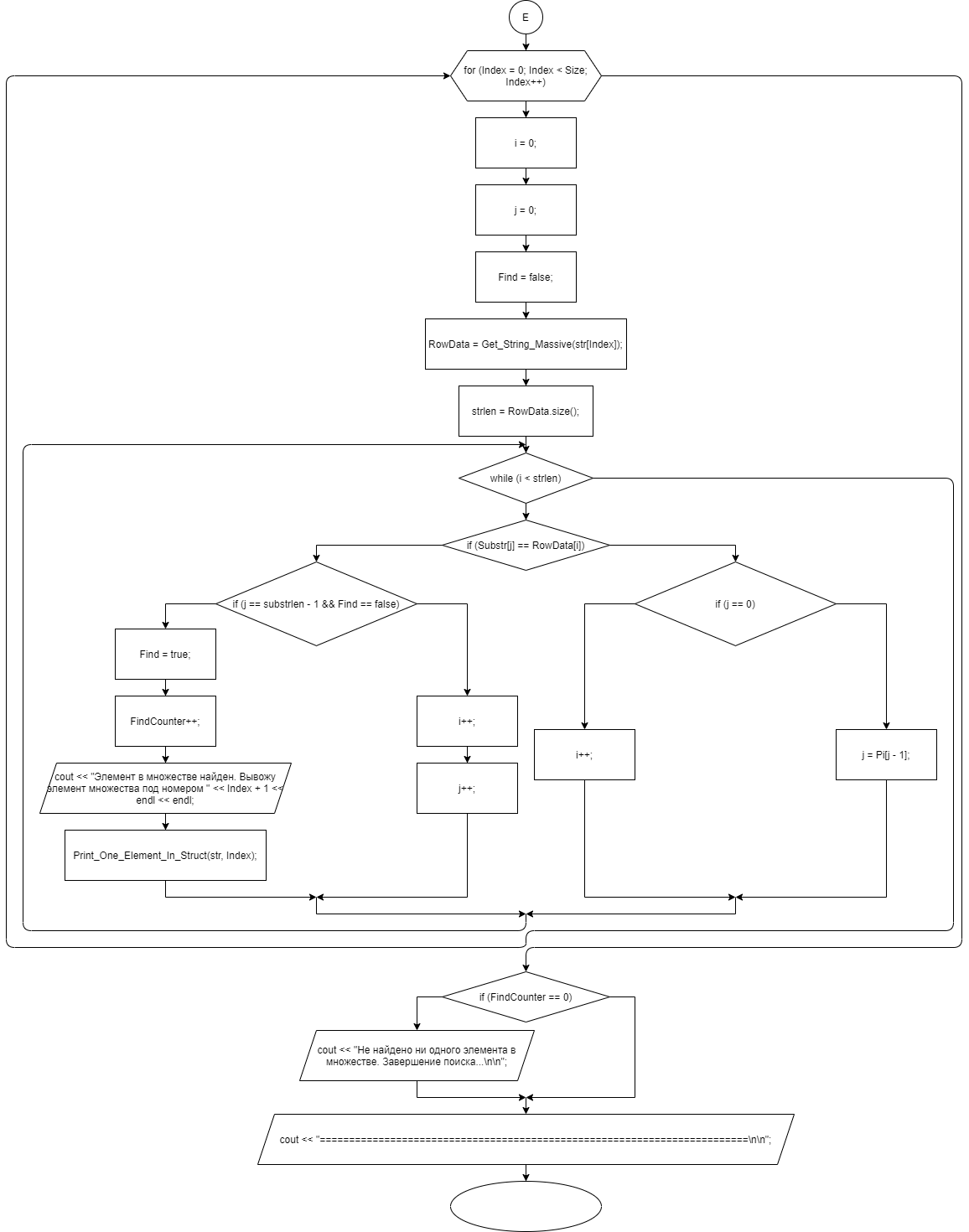


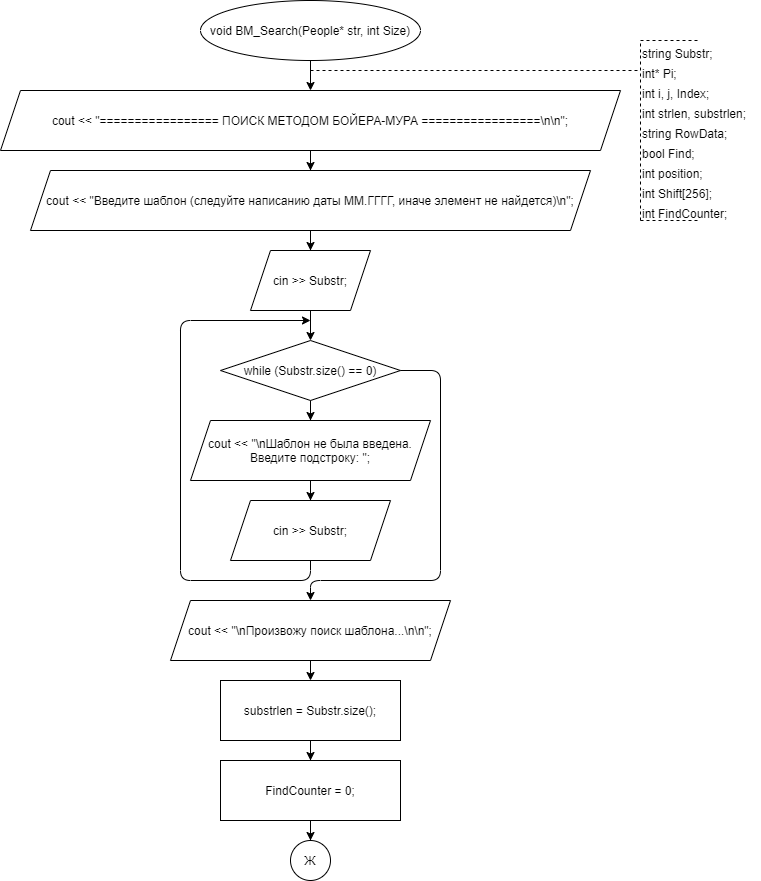


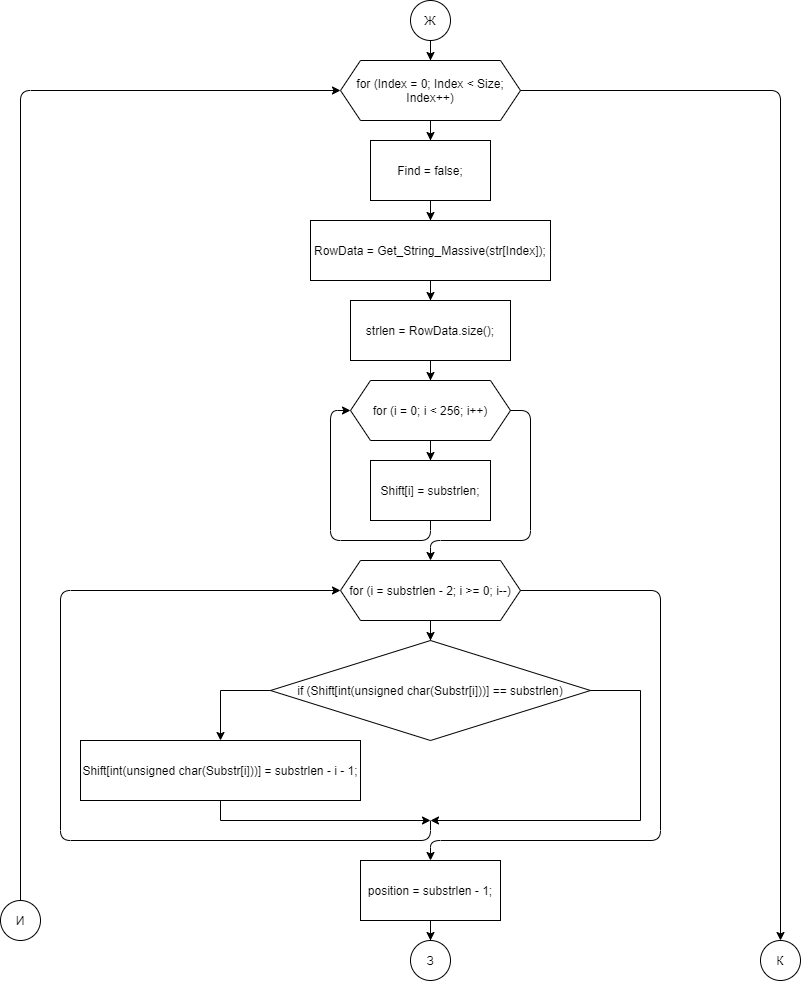


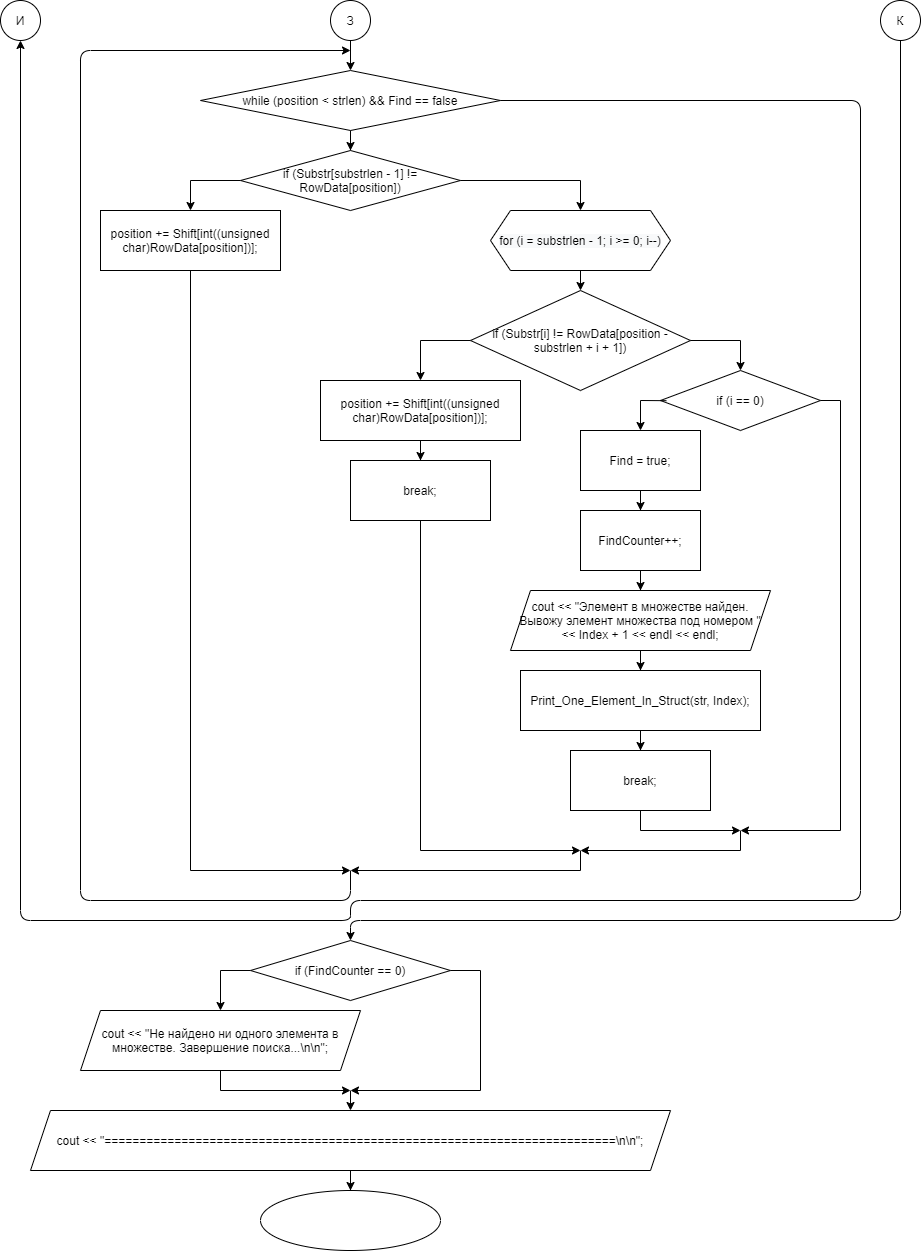


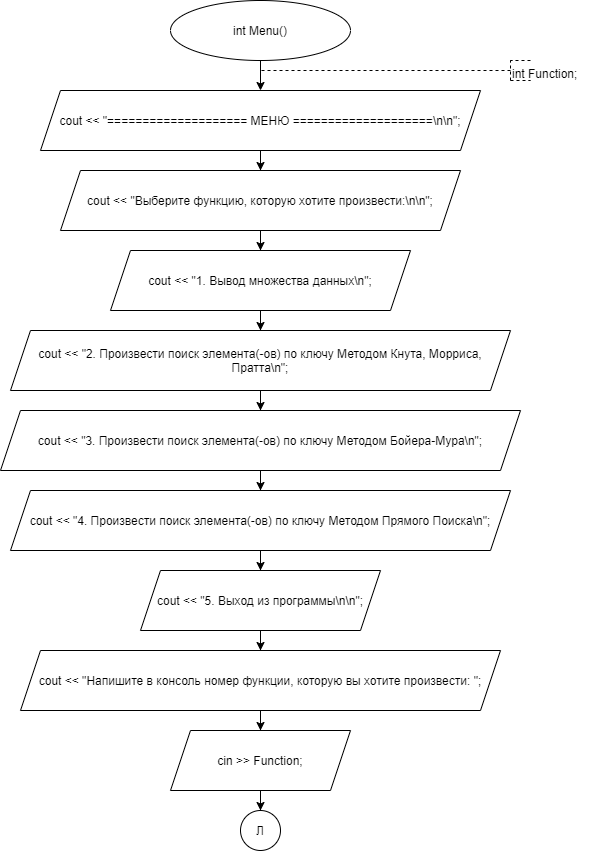


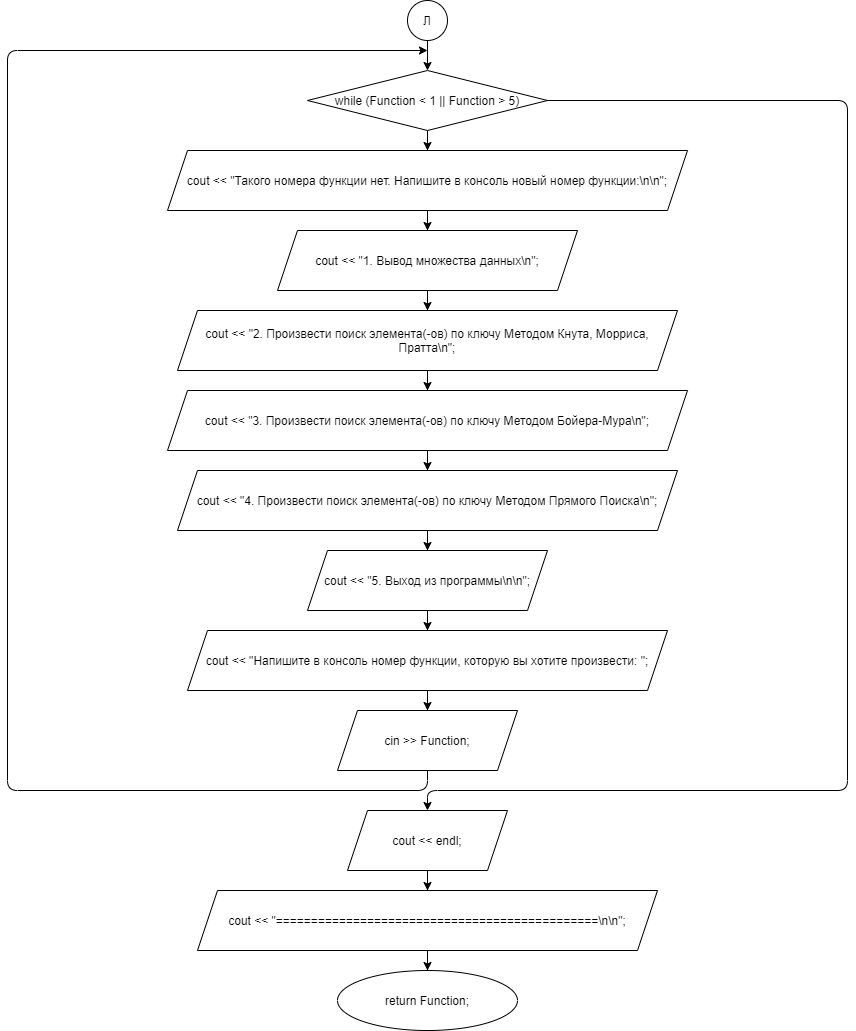


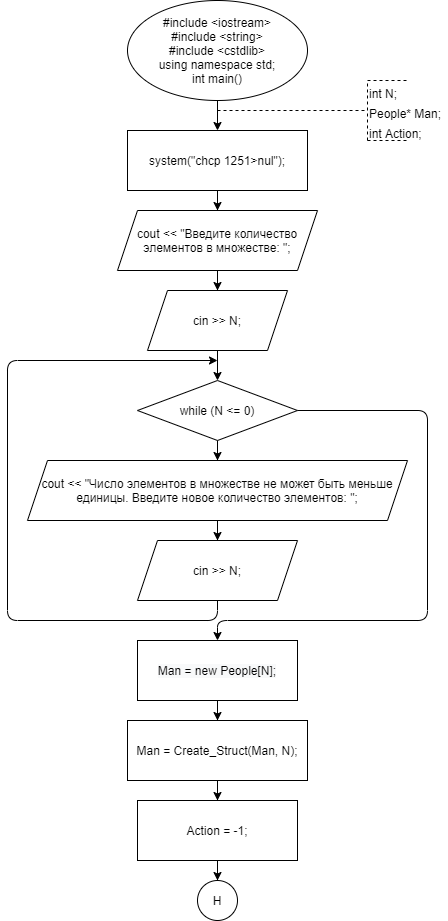


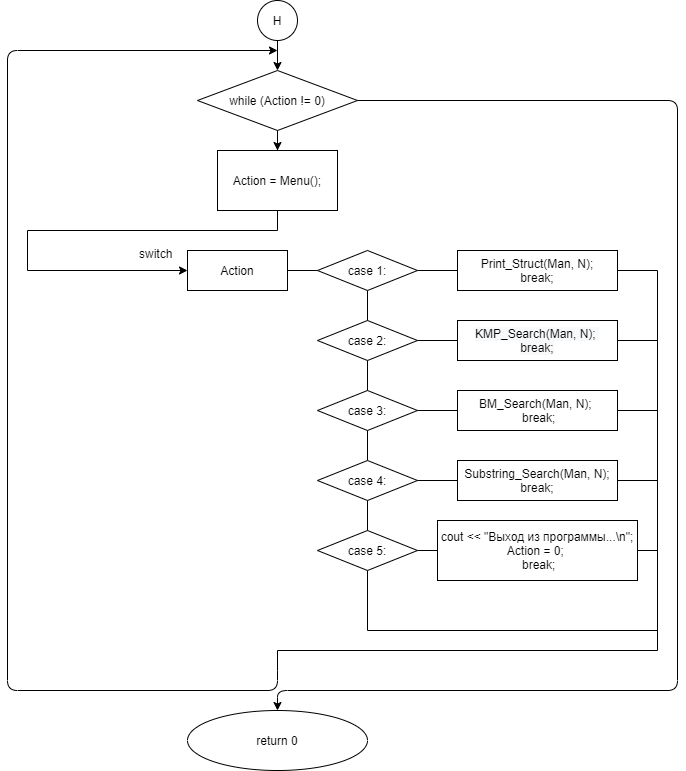












**Реализация задачи на языке С++**

#include <iostream>

#include <string>

#include <cstdlib>

using namespace std;

struct People

{

string Full\_Name;

int Month;

int Year;

string Phone\_Number;

};

People\* Create\_Struct(People\* str, int Size)

{

cout << "Введите элементы множества\n";

int i;

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

{

cout << i + 1 << " элемент:\n";

cout << "ФИО: ";

cin.ignore();

getline(cin, str[i].Full\_Name);

cout << "Дата рождения(ММ.ГГГГ) в числовом виде\n";

cout << "Месяц: ";

cin >> str[i].Month;

cout << "Год: ";

cin >> str[i].Year;

while ((str[i].Year <= 0) || (str[i].Month < 1 || str[i].Month > 12))

{

cout << "Некорректно введена дата рождения. Введите правильную дату рождения\n";

cout << "Месяц: ";

cin >> str[i].Month;

cout << "Год: ";

cin >> str[i].Year;

}

cout << "Номер телефона: ";

cin.ignore();

getline(cin, str[i].Phone\_Number);

}

return str;

}

void Print\_One\_Element\_In\_Struct(People\* str, int Index)

{

cout << "ФИО: " << str[Index].Full\_Name << endl;

if (str[Index].Month < 10)

{

cout << "Дата рождения: " << '0' << str[Index].Month << '.' << str[Index].Year << endl;

}

else

{

cout << "Дата рождения: " << str[Index].Month << '.' << str[Index].Year << endl;

}

cout << "Номер телефона: " << str[Index].Phone\_Number << endl << endl;

}

void Print\_Struct(People\* str, int Size)

{

cout << endl;

cout << "Вывожу множество...\n\n";

int i;

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

{

cout << i + 1 << " элемент:\n";

Print\_One\_Element\_In\_Struct(str, i);

}

}

string String\_Data(int\* Data)

{

int i;

string RowData;

for (i = 0; i < 7; i++)

{

switch (Data[i])

{

case 0:

{

RowData += '0';

break;

}

case 1:

{

RowData += '1';

break;

}

case 2:

{

RowData += '2';

break;

}

case 3:

{

RowData += '3';

break;

}

case 4:

{

RowData += '4';

break;

}

case 5:

{

RowData += '5';

break;

}

case 6:

{

RowData += '6';

break;

}

case 7:

{

RowData += '7';

break;

}

case 8:

{

RowData += '8';

break;

}

case 9:

{

RowData += '9';

break;

}

default:

{

RowData += '.';

break;

}

}

}

return RowData;

}

string Get\_String\_Massive(People Element)

{

int IntData[] = {0, 0, 0, 0, 0, 0, 0};

int i = 1;

if (Element.Month < 10)

{

IntData[0] = 0;

IntData[1] = Element.Month % 10;

}

else

{

while (Element.Month > 0)

{

IntData[i] = Element.Month % 10;

i--;

Element.Month = Element.Month / 10;

}

}

IntData[2] = -1;

i = 6;

while (Element.Year > 0)

{

IntData[i] = Element.Year % 10;

i--;

Element.Year = Element.Year / 10;

}

string StringData;

StringData = String\_Data(IntData);

return StringData;

}

void Substring\_Search(People\* str, int Size)

{

cout << "==================== ПРЯМОЙ ПОИСК ПОДСТРОКИ В СТРОКЕ ====================\n\n";

cout << "Введите подстроку (следуйте написанию даты MM.ГГГГ, иначе элемент не найдется)\n\nШаблон: ";

string Substr;

cin >> Substr;

while (Substr.size() == 0)

{

cout << "\nПодстрока не была введена. Введите подстроку: ";

cin >> Substr;

}

cout << "\nПроизвожу поиск подстроки...\n\n";

int i, j, Index;

int strlen, substrlen;

substrlen = Substr.size();

string RowData;

bool Find;

int FindCounter = 0;

for (Index = 0; Index < Size; Index++)

{

Find = false;

RowData = Get\_String\_Massive(str[Index]);

strlen = RowData.size();

for (i = 0; i < strlen - substrlen + 1; i++)

{

for (j = 0; j < substrlen; j++)

{

if (Substr[j] != RowData[i + j])

{

break;

}

else

{

if (j == substrlen - 1 && Find == false)

{

Find = true;

FindCounter++;

cout << "Элемент в множестве найден. Вывожу элемент множества под номером " << Index + 1 << endl << endl;

Print\_One\_Element\_In\_Struct(str, Index);

}

}

}

}

}

if (FindCounter == 0)

{

cout << "Не найдено ни одного элемента в множестве. Завершение поиска...\n\n";

}

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

}

int\* Pi\_Function(string Substring)

{

int\* Pi = new int[Substring.length()];

int i = 1, j = 0;

Pi[0] = 0;

while (i < Substring.length())

{

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

{

Pi[i] = j + 1;

i++;

j++;

}

else

{

if (j == 0)

{

Pi[i] = 0;

i++;

}

else

{

j = Pi[j - 1];

}

}

}

return Pi;

}

void KMP\_Search(People\* str, int Size)

{

cout << "================= ПОИСК МЕТОДОМ КНУТА, МОРРИСА, ПРАТТА =================\n\n";

cout << "Введите шаблон (следуйте написанию даты MM.ГГГГ, иначе элемент не найдется)\n";

string Substr; // Подстрока (шаблон)

cin >> Substr;

while (Substr.size() == 0)

{

cout << "Шаблон не был введен. Введите шаблон: ";

cin >> Substr;

}

cout << "\nПроизвожу поиск шаблона...\n\n";

int\* Pi = new int[Substr.length()];

Pi = Pi\_Function(Substr);

int i, j, Index;

int strlen, substrlen;

substrlen = Substr.size();

string RowData;

bool Find;

int FindCounter = 0;

for (Index = 0; Index < Size; Index++)

{

i = 0;

j = 0;

Find = false;

RowData = Get\_String\_Massive(str[Index]);

strlen = RowData.size();

while (i < strlen)

{

if (Substr[j] == RowData[i])

{

if (j == substrlen - 1 && Find == false)

{

Find = true;

FindCounter++;

cout << "Элемент в множестве найден. Вывожу элемент множества под номером " << Index + 1 << endl << endl;

Print\_One\_Element\_In\_Struct(str, Index);

}

else

{

j++;

i++;

}

}

else

{

if (j == 0)

{

i++;

}

else

{

j = Pi[j - 1];

}

}

}

}

if (FindCounter == 0)

{

cout << "Не найдено ни одного элемента в множестве. Завершение поиска...\n\n";

}

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

}

void BM\_Search(People\* str, int Size)

{

cout << "================= ПОИСК МЕТОДОМ БОЙЕРА-МУРА =================\n\n";

cout << "Введите шаблон (следуйте написанию даты MM.ГГГГ, иначе элемент не найдется)\n";

string Substr;

cin >> Substr;

while (Substr.size() == 0)

{

cout << "Шаблон не был введен. Введите шаблон: ";

cin >> Substr;

}

cout << "\nПроизвожу поиск шаблона...\n\n";

int i, Index;

int strlen, substrlen;

substrlen = Substr.size();

string RowData;

bool Find;

int FindCounter = 0;

for (Index = 0; Index < Size; Index++)

{

Find = false;

RowData = Get\_String\_Massive(str[Index]);

strlen = RowData.size();

int position;

int Shift[256];

for (i = 0; i < 256; i++)

{

Shift[i] = substrlen;

}

for (i = substrlen - 2; i >= 0; i--)

{

if (Shift[int(unsigned char(Substr[i]))] == substrlen)

{

Shift[int(unsigned char(Substr[i]))] = substrlen - i - 1;

}

}

position = substrlen - 1;

while (position < strlen)

{

if (Substr[substrlen - 1] != RowData[position])

{

position += Shift[int((unsigned char)RowData[position])];

}

else

{

for (i = substrlen - 1; i >= 0; i--)

{

if (Substr[i] != RowData[position - substrlen + i + 1])

{

position += Shift[int((unsigned char)RowData[position])];

break;

}

else if (i == 0)

{

Find = true;

FindCounter++;

cout << "Элемент в множестве найден. Вывожу элемент множества под номером " << Index + 1 << endl << endl;

Print\_One\_Element\_In\_Struct(str, Index);

break;

}

}

}

}

}

if (FindCounter == 0)

{

cout << "Не найдено ни одного элемента в множестве. Завершение поиска...\n\n";

}

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

}

int Menu()

{

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

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

cout << "1. Вывод множества данных\n";

cout << "2. Произвести поиск элемента(-ов) по ключу Методом Поиска Кнутта, Морриса, Пратта\n";

cout << "3. Произвести поиск элемента(-ов) по ключу Методом Поиска Бойера-Мура\n";

cout << "4. Произвести поиск элемента(-ов) по ключу Методом Прямого Поиска\n";

cout << "5. Выход из программы\n\n";

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

int Function; // Переменная, которая хранит номер действия

cin >> Function;

while (Function < 1 || Function > 5)

{

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

cout << "1. Вывод множества данных\n";

cout << "2. Произвести поиск элемента(-ов) по ключу Методом Поиска Кнутта, Морриса, Пратта\n";

cout << "3. Произвести поиск элемента(-ов) по ключу Методом Поиска Бойера-Мура\n";

cout << "4. Произвести поиск элемента(-ов) по ключу Методом Прямого Поиска\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");

int N;

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

cin >> N;

while (N <= 0)

{

cout << "Число элементов в множестве не может быть меньше единицы. Введите новое количество элементов: ";

cin >> N;

}

People\* Man = new People[N];

Man = Create\_Struct(Man, N);

int Action = -1;

while (Action != 0)

{

Action = Menu();

switch (Action)

{

case 1:

{

Print\_Struct(Man, N);

break;

}

case 2:

{

BM\_Search(Man, N);

break;

}

case 3:

{

KMP\_Search(Man, N);

break;

}

case 4:

{

Substring\_Search(Man, N);

break;

}

case 5:

{

cout << "Выход из программы...\n";

Action = 0;

break;

}

}

}

}

**Скриншоты результатов работы программы**

