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 №12**

Дисциплина: «Основы теории алгоритмов и структуры данных»

Тема: Алгоритмы поиска. Работа со списками. Метод интерполяционного поиска

Вариант 19

Выполнила:

Студент группы ИВТ-20-2б

Ананина Арина Юрьевна

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

1. 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. Предусмотреть возможность добавления и удаления элементов из массива (файла).  
4. 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.

2. Анализ задачи

1. В ходе работы были использованы следующие типы данных:
   1. Поля структуры Date – int.
   2. Поле FIO структуры Human типа string, passportNumber - int.
   3. Для хранения объектов структуры Human используется класс vector.
   4. Функции поиска, кроме Интерполяционного, возвращают тип данных int, функция Интерполяционного 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 Human.
2. Для решения задачи данные были представлены в следующем виде:
   1. Для хранения информации о человеке используется структура Human с 3 полями.
3. Структуры, которые использовались при решении задачи:
   1. Структура Human имеет метод заполнения объекта (GenerateData), которая заполняет поля объекта с помощью ДСЧ, для заполнения поля FIO используется 3 алфавита (имен, фамилий, отчеств). Номер паспорта - случайное число от 1 до 6000. Аналогично, заполняются поле DateOfBirth.
4. Для операций ввода и вывода использовались следующие операторы и функции:
   1. cin и cout в методе Print, в функции PrintArray.
   2. Getline, для ввода ФИО пользователем (используется в функции удаления элемента из массива по ключу)
5. Поставленные задачи будут решены следующими действиями:
   1. Функция FindInterPolationSearch преобразует массив объектов структуры в массив номеров паспортов. 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.

int mid, left = 0, right = size - 1;

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

{

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

if (collection[mid].Data < key) {

left = mid + 1;

}

else if (collection[mid].Data > key) {

right = mid - 1;

}

else {

return collection[mid].Index;

}

}

if (collection[left].Data == key) {

return collection[left].Index;

}

if (collection[right].Data == key) {

return collection[right].Index;

}

return-1;

* 1. Функция Line\_search преобразует ключ и массив по тому же принципу что и функция Interpol\_search. С помощью цикла for осуществляется проход по всем элементам массива, в случае удовлетворения элемента массива ключу, функция возвращает индекс i, иначе-1.

int LineSearch(vector<Human> humans, Human key)

{

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

{

if (humans[i].PassportNumber == key.PassportNumber )

return i;

}

return -1;

}

* 1. Функция FindSubstringInString преобразует массив объектов структуры Human в строку номеров паспортов, ключ так же преобразовывается в строку. 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.

string substring = "";

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

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

{

string newLine = "";

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

listOfNumbers += newLine;

}

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

{

int j = 0;

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

j++;

if (j == substring.size())

return i / 10;

}

return-1;

3. Код программы

#include <iostream>

#include <string>

#include <vector>

#include <fstream>

using namespace std;

struct Date

{

int Day, Month, Year;

};

struct Human

{

string FIO;

Date DateOfBirth;

long long PassportNumber;

void Print()

{

cout << endl;

cout << "ФИО: " << FIO << endl;

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

cout << "Номер паспорта: " << PassportNumber << endl;

}

void GenerateData()

{

string names[] = { "Иван", "Юрий", "Алексей", "Артём", "Дмитрий" };

string surnames[] = { "Иванов" , "Ананин", "Тарасов" };

string patronymic[] = {"Андреевич", "Вадимович", "Михаилович", "Кирилович", "Ильич" };

FIO = surnames[rand() % 3] + " " + names[rand() % 5] + " " + patronymic[rand() % 5];

DateOfBirth.Day = 1 + rand() % 28;

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

DateOfBirth.Year = 1900 + rand() % 200;

PassportNumber = 5700000000 + (rand() % 100) \* 1000000 + rand() % 1000000;

}

};

vector<Human> listToDelete;

vector<Human> CreateArray(int size = 100)

{

vector<Human> res;

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

{

Human newEl;

newEl.GenerateData();

res.push\_back(newEl);

}

return res;

}

void SaveToFile(vector<Human> humans)

{

string path = "File.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 << endl;

out << humans[i].DateOfBirth.Month << endl;

out << humans[i].DateOfBirth.Year << endl;

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

}

out.close();

cout << "Список записан в файл File.txt" << endl;

}

void LoadFromFile(vector<Human>& humans)

{

string path = "File.txt";

ifstream in = ifstream();

in.open(path);

if (in.is\_open())

{

humans = vector<Human>();

while (!in.eof())

{

Human newElement = Human();

getline(in, newElement.FIO);

in >> newElement.DateOfBirth.Day;

in >> newElement.DateOfBirth.Month;

in >> newElement.DateOfBirth.Year;

in >> newElement.PassportNumber;

in.ignore();

humans.push\_back(newElement);

}

humans.erase(humans.end() - 1);

cout << "Загрузка из файла завершена" << endl;

}

else

{

cout << "Файл не открылся" << endl;

}

in.close();

}

void PrintArray(vector<Human> humans)

{

if (humans.size() < 1)

{

cout << "Список пуст" << endl;

return;

}

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

{

cout << "Элемент " << i << endl;

humans[i].Print();

}

}

struct SearchItem

{

int Index;

long long Data;

};

int FindInterPolationSearch(vector<Human> humans, Human info)

{

long long key = info.PassportNumber;

int size = humans.size();

SearchItem\* collection = new SearchItem[size];

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

{

SearchItem item;

item.Index = i;

item.Data = humans[i].PassportNumber;

collection[i] = item;

}

for (int i = 0; i < size - 1; i++)

{

for (int j = 0; j < size - i - 1; j++)

{

if (collection[j].Data > collection[j + 1].Data)

{

SearchItem tmp = collection[j];

collection[j] = collection[j + 1];

collection[j + 1] = tmp;

}

}

}

int mid, left = 0, right = size - 1;

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

{

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

if (collection[mid].Data < key) {

left = mid + 1;

}

else if (collection[mid].Data > key) {

right = mid - 1;

}

else {

return collection[mid].Index;

}

}

if (collection[left].Data == key) {

return collection[left].Index;

}

if (collection[right].Data == key) {

return collection[right].Index;

}

return -1;

}

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

{

string listOfNumbers;

string substring = "";

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

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

{

string newLine = "";

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

listOfNumbers += newLine;

}

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

{

int j = 0;

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

j++;

if (j == substring.size())

return i / 10;

}

return -1;

}

int LineSearch(vector<Human> humans, Human key)

{

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

{

if (humans[i].PassportNumber == key.PassportNumber )

return i;

}

return -1;

}

void Search(vector<Human> humans)

{

if (humans.size() < 1)

{

cout << "" << endl;

return;

}

Human key;

cout << endl << "Введите номер паспорта, по которому необходимо искать элемент: ";

cin >> key.PassportNumber;

int ind = FindInterPolationSearch(humans, key);

cout << endl << "Интерполяционный поиск" << endl;

if (ind != -1)

cout << "Человек с заданным номером паспорта есть: " << humans[ind].FIO << endl;

else

cout << "Человек с заданным номером паспорта отсутствует" << endl;

ind = LineSearch(humans, key);

cout << endl << "Линейный поиск" << endl;

if (ind != -1)

cout << "Человек с заданным номером паспорта есть: " << humans[ind].FIO << endl;

else

cout << "Человек с заданным номером паспорта отсутствует" << endl;

ind = FindSubstringInString(humans, key);

cout << endl << "Поиск подстроки в строке" << endl;

if (ind != -1)

cout << "Человек с заданным номером паспорта есть: " << humans[ind].FIO << endl;

else

cout << "Человек с заданным номером паспорта отсутствует" << endl;

}

void AddElement(vector<Human>& humans)

{

char vvod;

cout << "1 - добавить по номеру" << endl;

cout << "2 - добавить по ключу" << endl;

cin >> vvod;

if (vvod == '1')

{

int ind;

do

{

cout << "Введите номер элемента,на место которого необходимо вставить элемент: ";

cin >> ind;

} while (ind < 0 || ind > humans.size());

Human t;

t.GenerateData();

humans.insert(humans.begin() + ind, t);

}

else if (vvod == '2')

{

Human key;

cout << endl << "Введите номер паспорта элемента, на место которого надо вставить элемент: ";

cin >> key.PassportNumber ;

int ind = FindSubstringInString(humans, key);

if (ind != -1)

{

Human t;

t.GenerateData();

humans.insert(humans.begin() + ind, t);

}

else

{

cout << "Элемент с заданным номером паспорта не найден" << endl;

}

}

else

{

cout << "Введено некорректное значение." << endl;

}

}

void DeleteElement(vector<Human>& humans)

{

if (humans.size() == 0)

{

cout << "Список пуст" << endl;

return;

}

char vvod;

cout << "1 - удаление по номеру элемента" << endl;

cout << "2 - удаление по ключу" << endl;

cin >> vvod;

if (vvod == '1')

{

int ind;

do

{

cout << "Введите номер элемента, который необходимо удалить ";

cin >> ind;

} while (ind < 0 || ind >= humans.size());

listToDelete.push\_back(humans[ind]);

humans.erase(humans.begin() + ind);

}

else if (vvod == '2')

{

Human key;

cout << endl << "Введите номер паспорта, по которому надо удалить элемент: ";

cin >> key.PassportNumber;

int ind = FindSubstringInString(humans, key);

if (ind != -1)

{

Human t;

listToDelete.push\_back(humans[ind]);

humans.erase(humans.begin() + ind);

}

else

{

cout << "Такого элемента в списке нет." << endl;

}

}

else

{

cout << "Введено некорректное значение." << endl;

}

}

int main()

{

setlocale(LC\_ALL, "rus");

vector<Human> humans;

char vvod = ' ';

while (vvod != '7')

{

cout << "0 - поиск элемента" << endl;

cout << "1 - создание списка" << endl;

cout << "2 - считать из файла" << endl;

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

cout << "4 - распечатать список" << endl;

cout << "5 - добавить элемент" << endl;

cout << "6 - удалить элемент" << endl;

cout << "7 - выход" << endl;

cin >> vvod;

switch (vvod)

{

case '0': Search(humans); break;

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

case '2': LoadFromFile(humans); break;

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

case '4': PrintArray(humans); break;

case '5': AddElement(humans); break;

case '6': DeleteElement(humans); break;

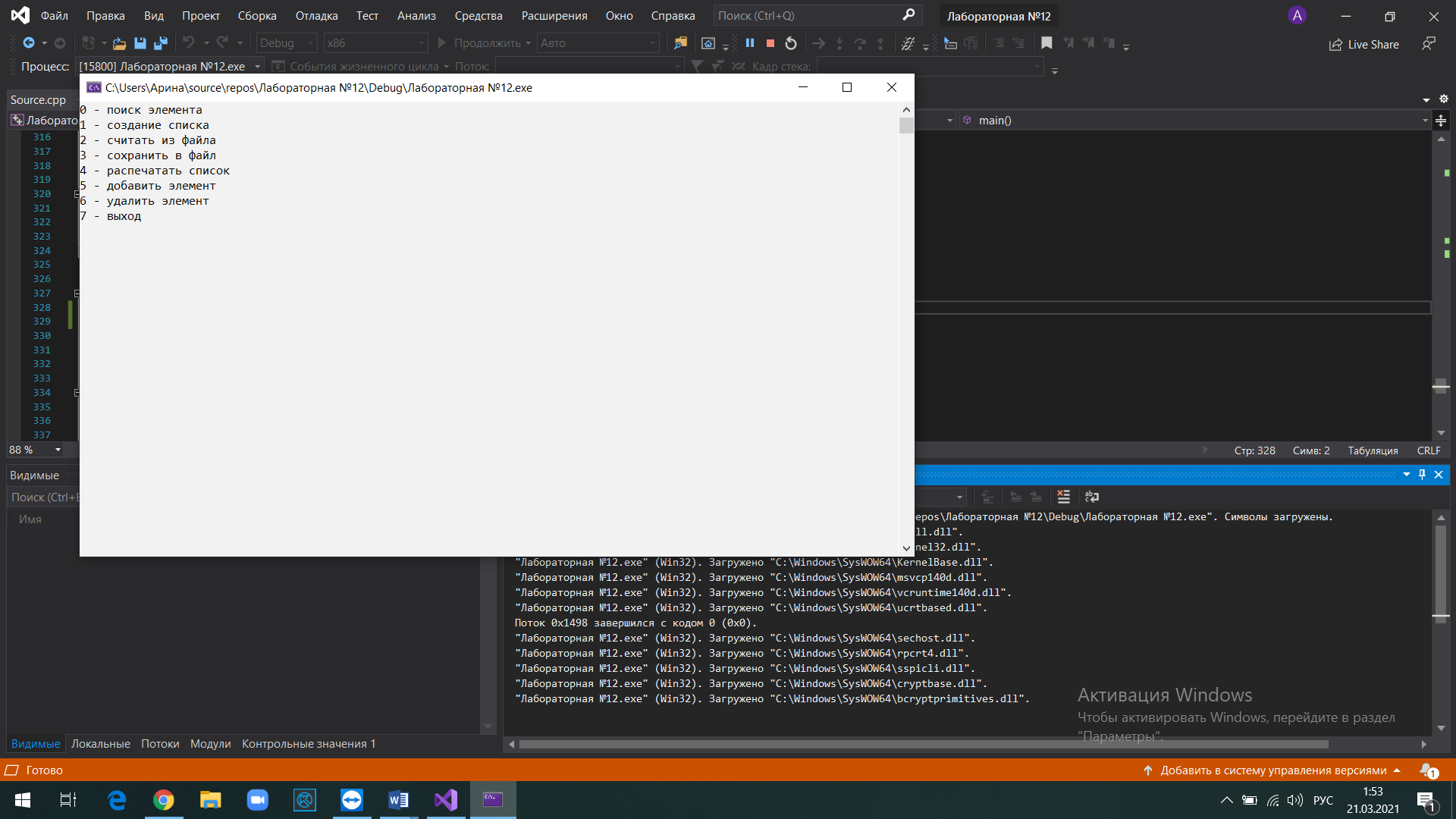
}

}

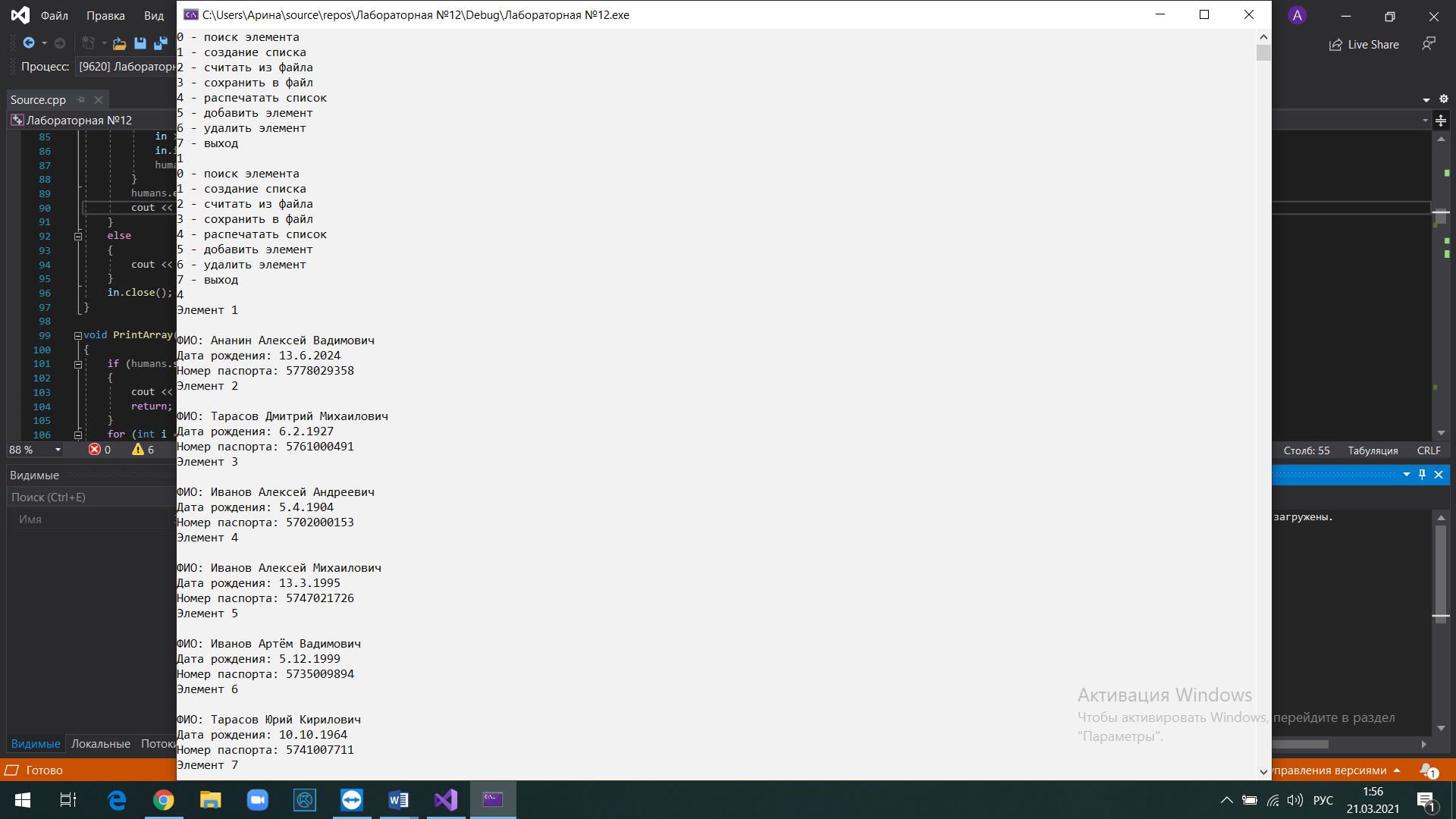
}

4. Выполнение программы.

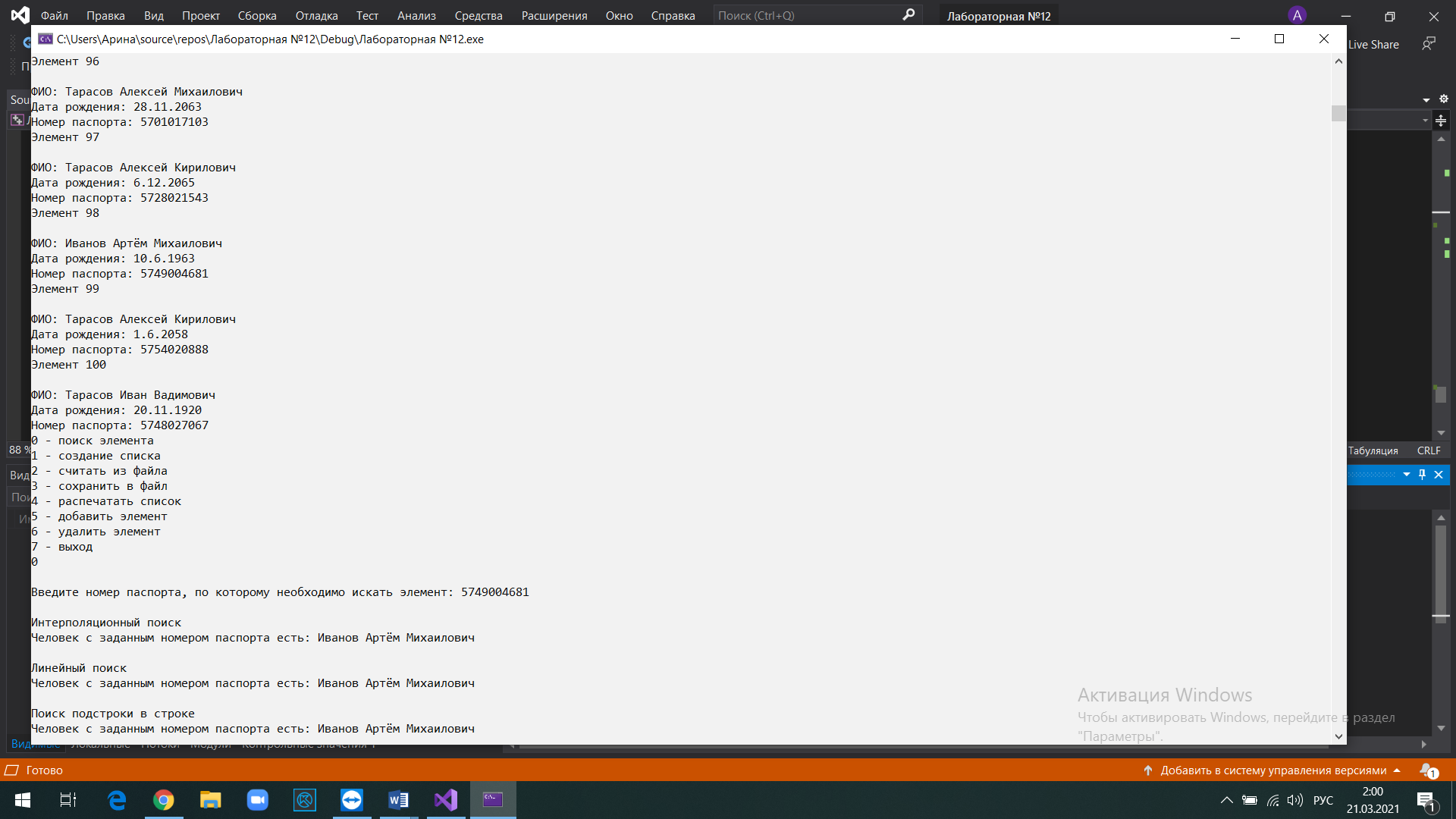
Организованно меню, при вводе пользователем числа, выполняется соответствующая команда.



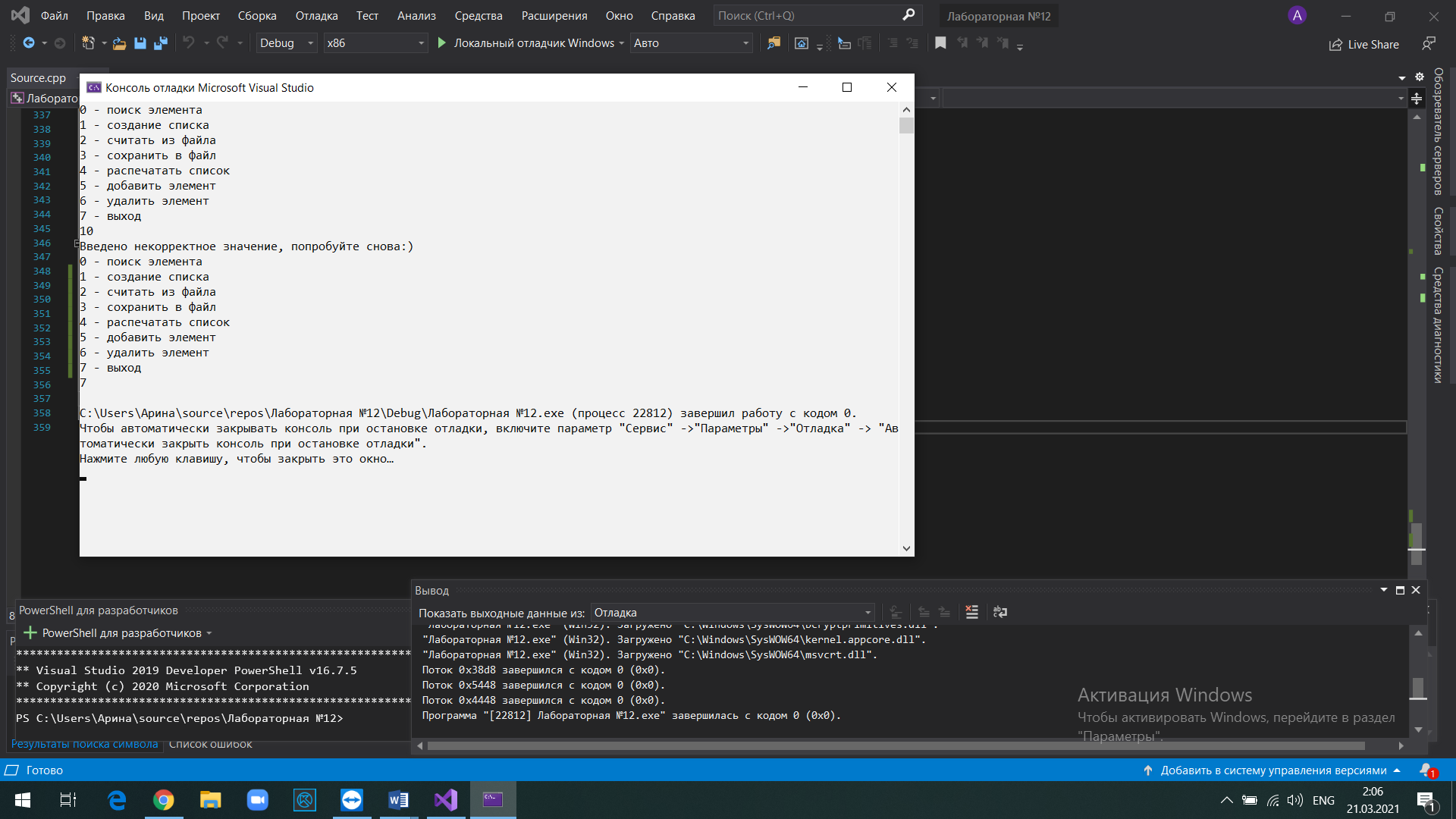
Создание списка происходит на 100 элементов, при вводе команда №4 все элементы печатаются по порядку.



При поиске элемента по ключу, осуществляется каждый из методов и выводится имя найденного по ключу элемента.



При вводе пользователем несуществующей команды, программа выведет на экран сообщение об этом и запросит новое значение.



5. Блок схема

