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 лабораторной работе №13**

Дисциплина: основы алгоритмизации и программирования

Тема: “Задача о 8 ферзях"

**Вар.21**

Выполнил работу

студент группы ИВТ-20-2Б

Галинов О.Ю.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2020

**Цель задачи**

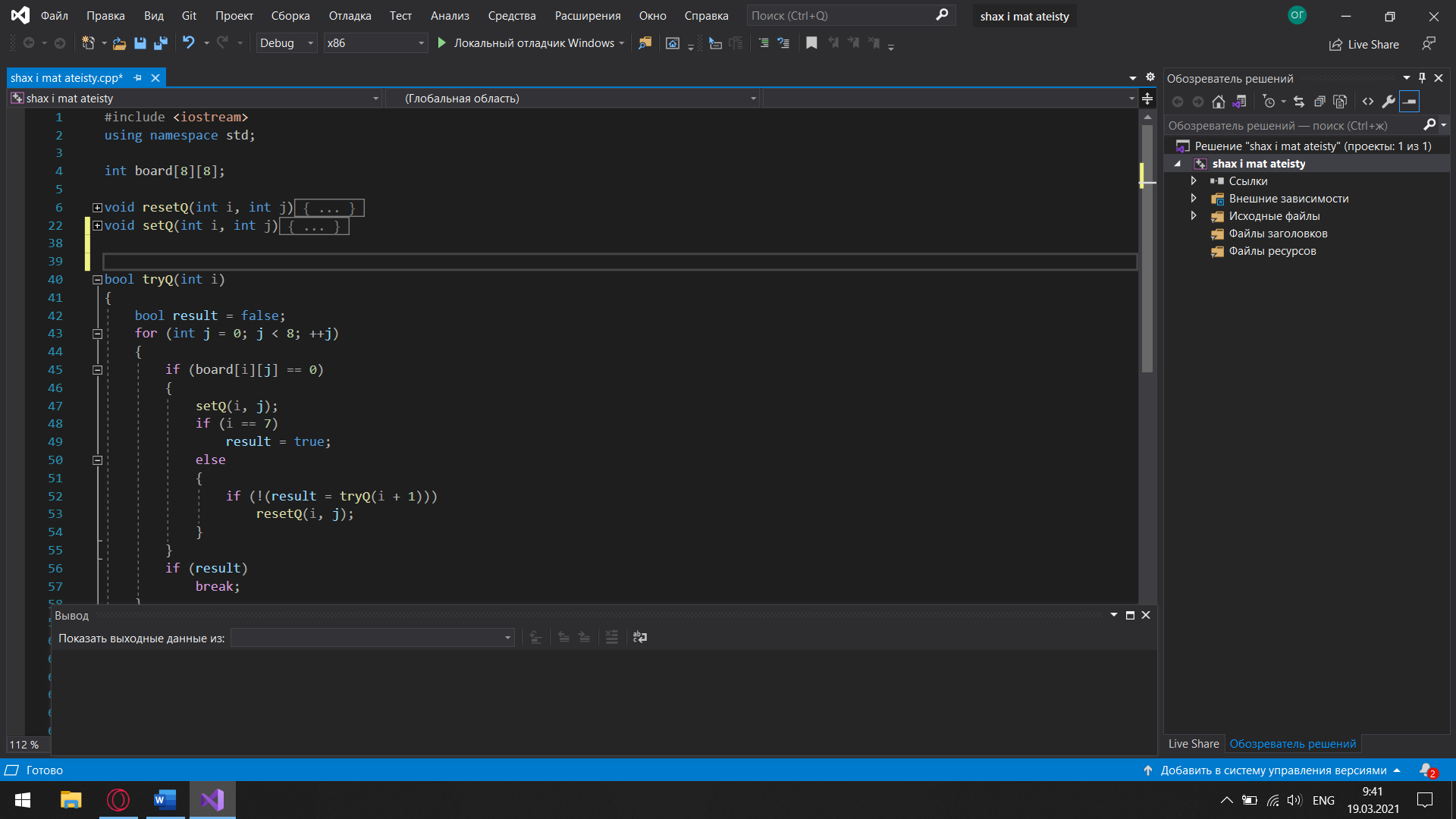
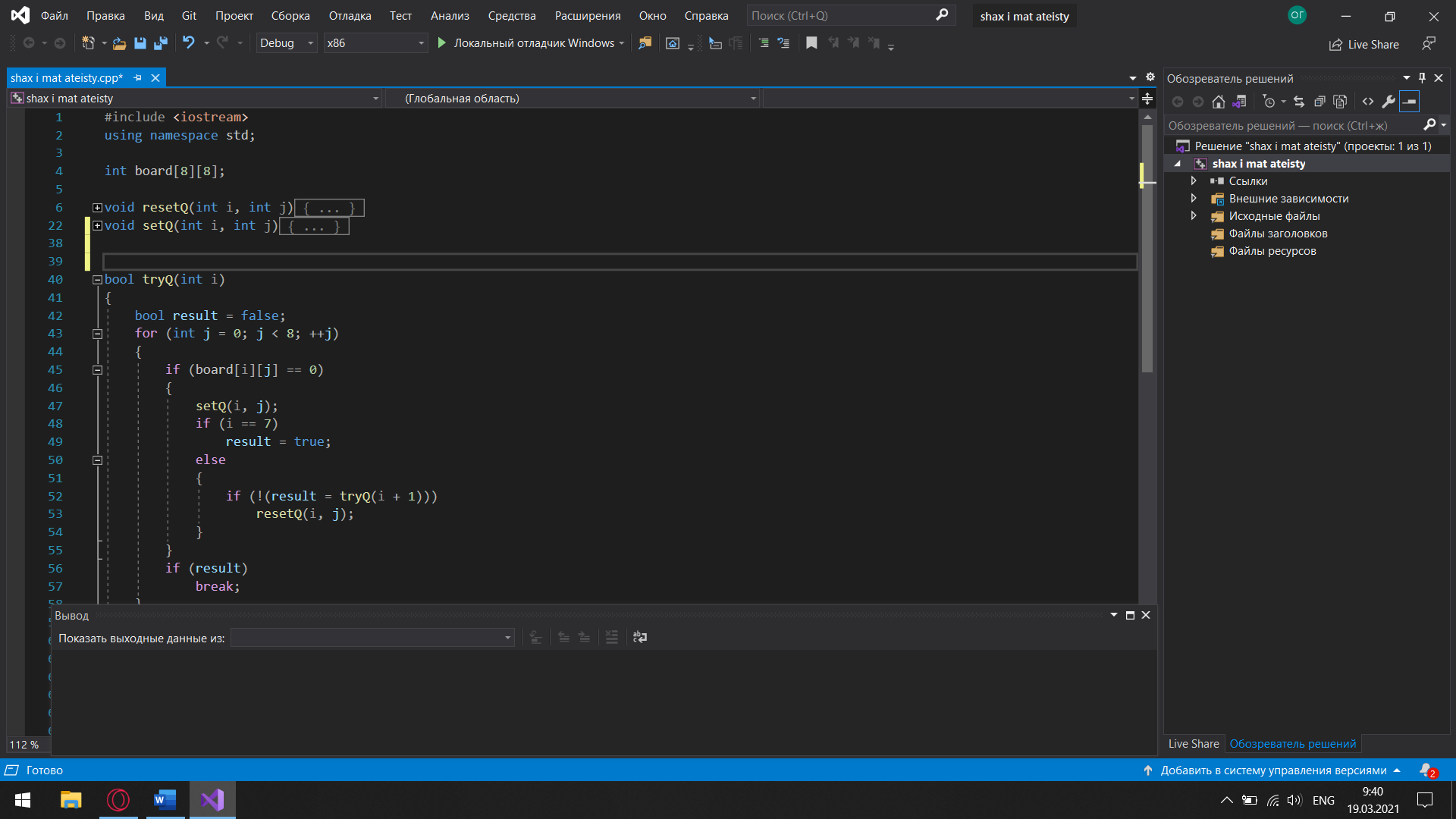
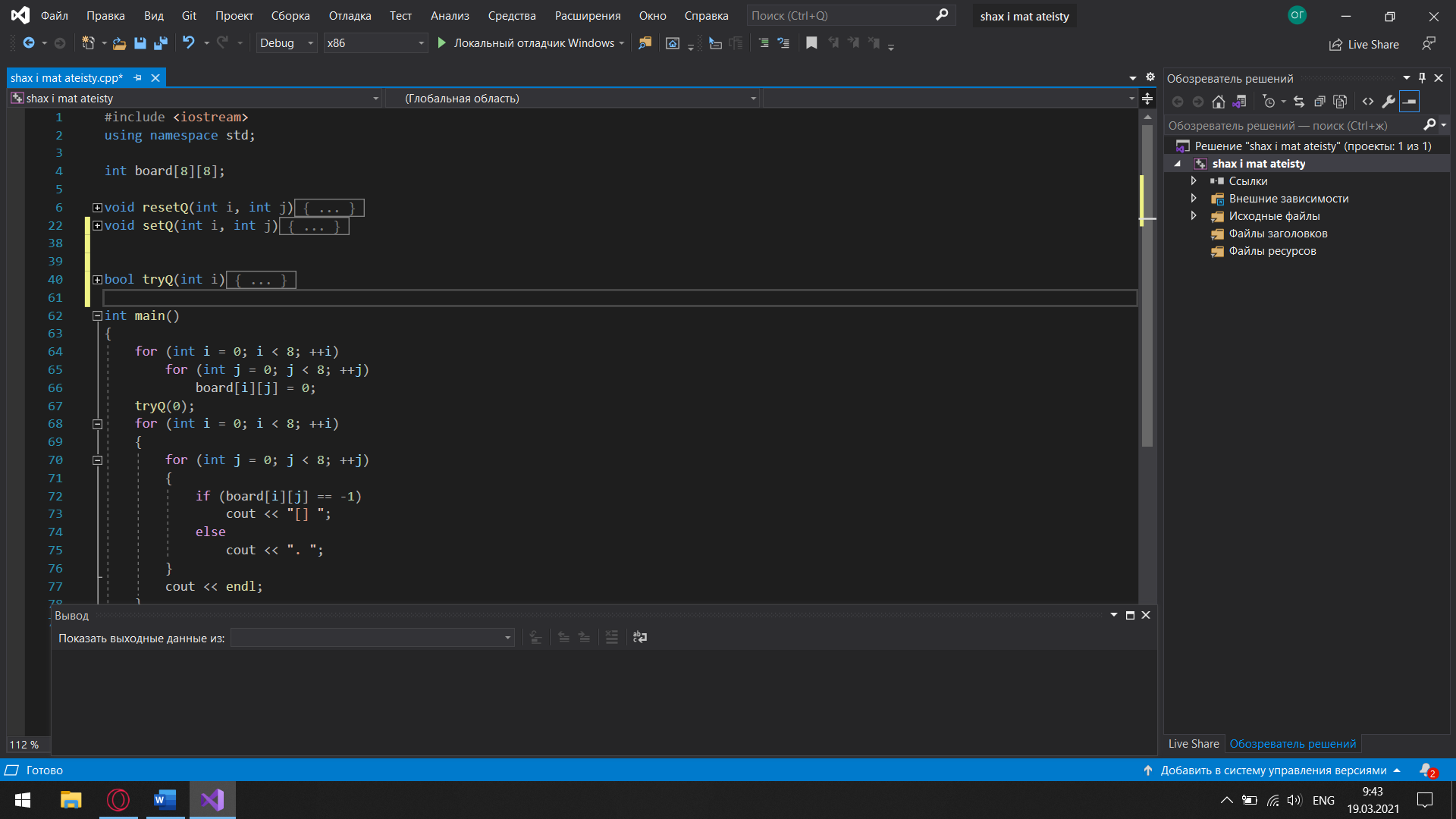
1) Получить практические навыки работы с поиском c возвратом;  
Для достижения поставленной цели необходимо решить следующие задачи:

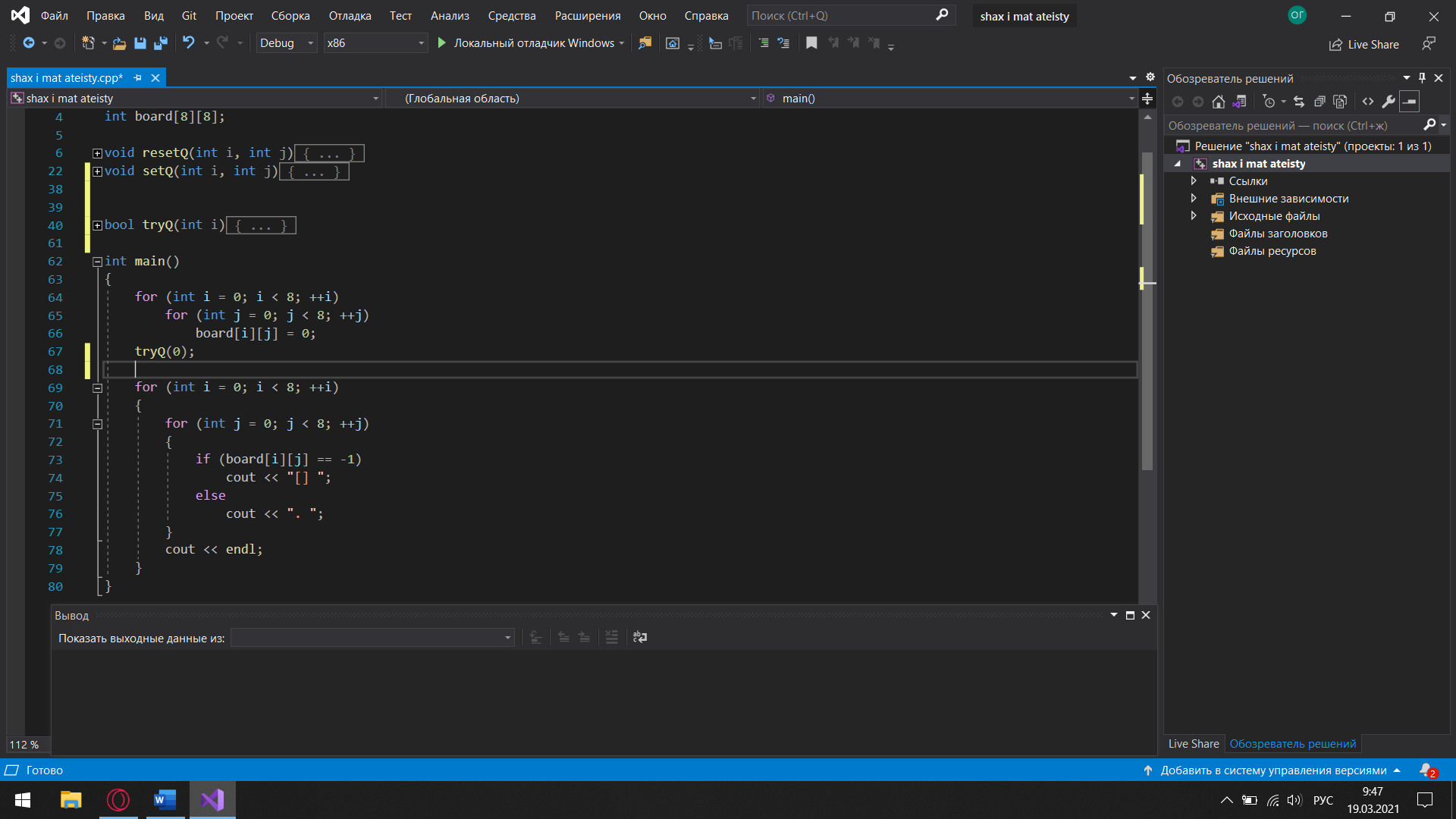
* Разработать функцию установки ферзя.
* Разработать функцию удаления ферзя.
* Разработать функцию нахождения расстановки 8 ферзей.
* Разработать функцию вывода доски.
* Разработать программу.

**Постановка задачи**

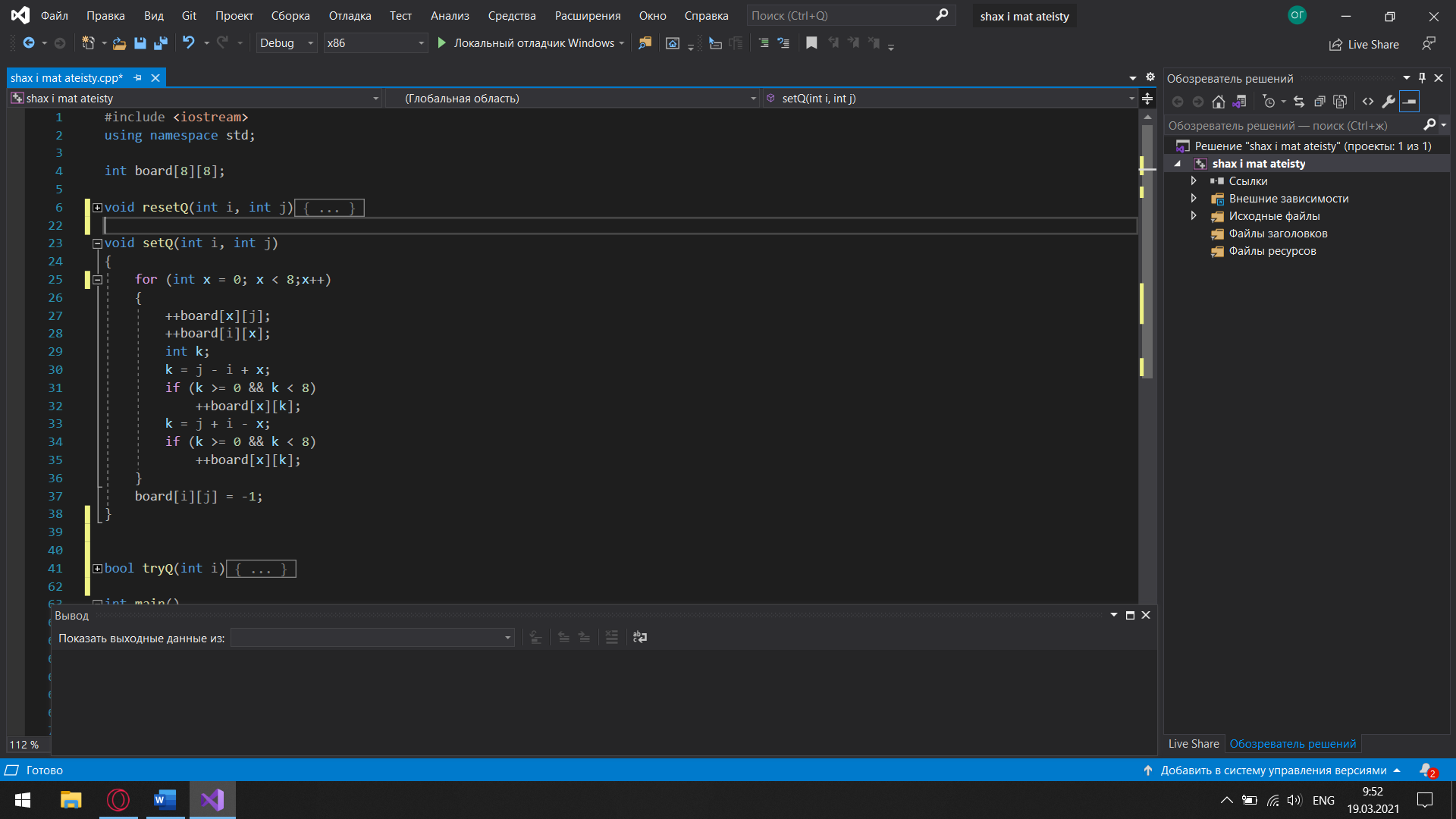
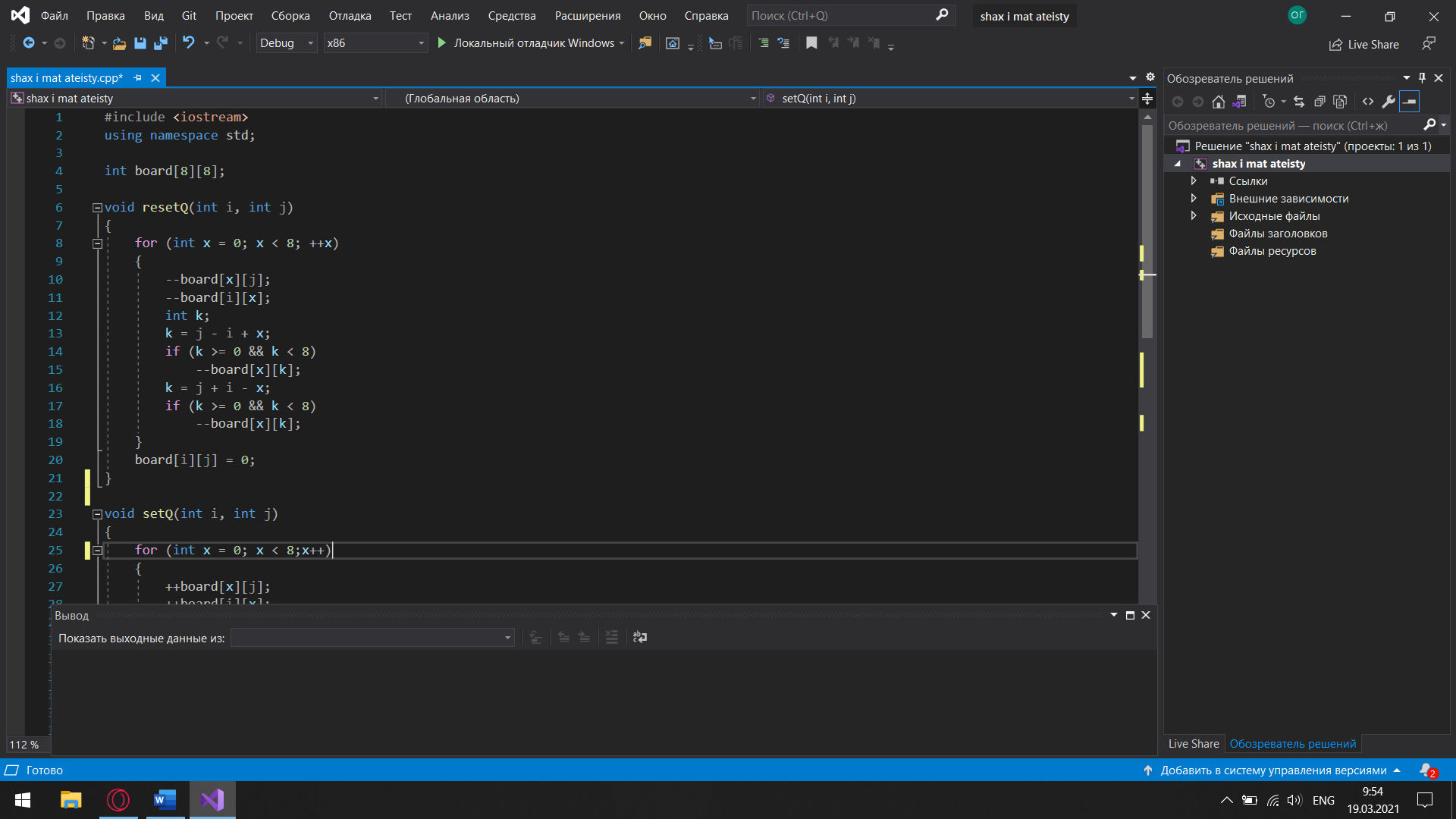
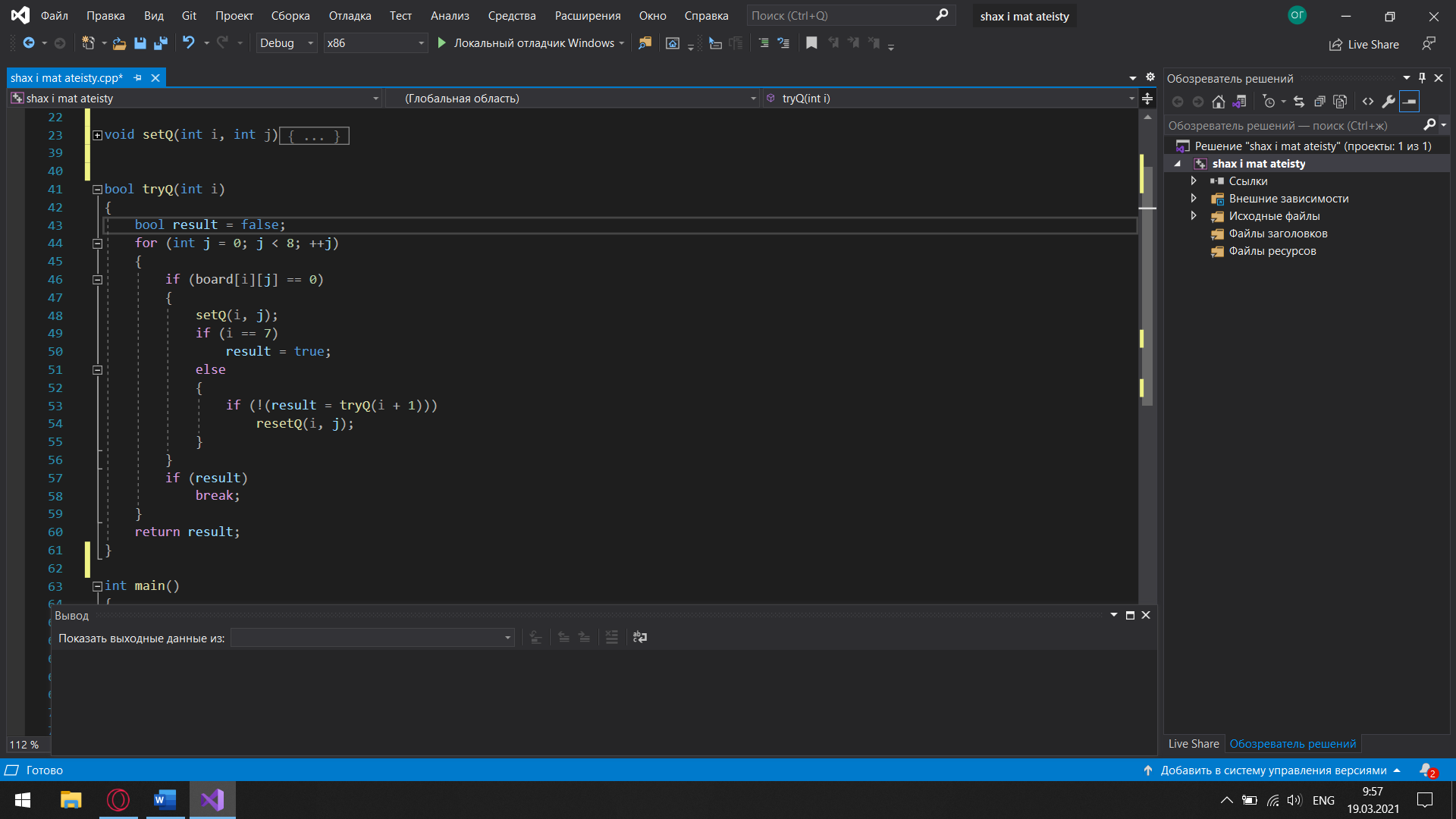
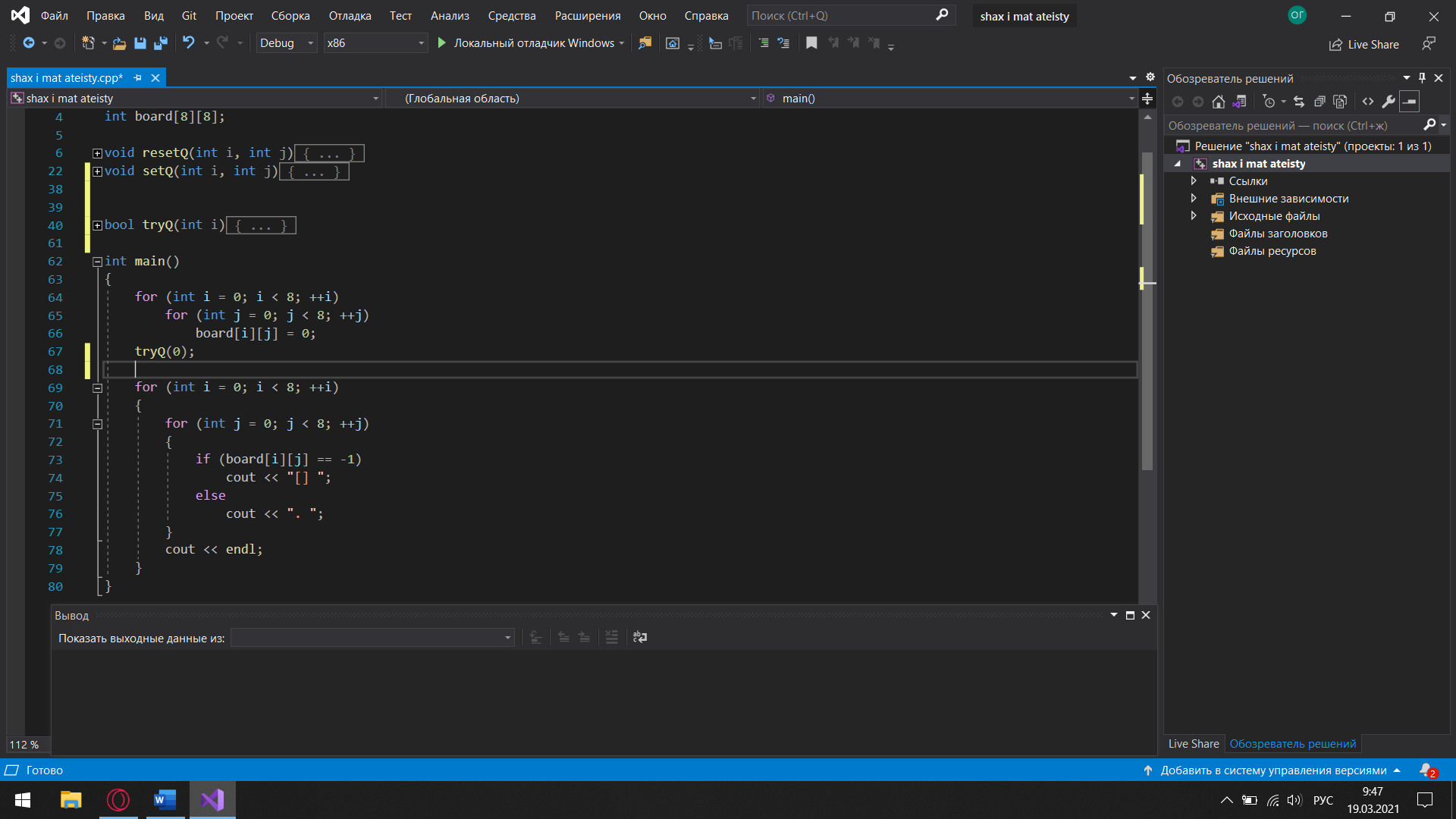
Дана шахматная доска 8x8. Расставить ферзей так, чтобы любые два ферзя не били друг друга.

**Анализ задачи**

1. Для решения задачи используются типы данных **int**.
   1. В качестве шахматного поля выступает глобальный двумерный статический массив (**board**) типа **int**. Он хранит в себе информацию о свободных клетках (0), клетках, занятых ферзями (-1), клетках, находящихся под боем (число большее, чем нуль).  
      
   2. Функция добавления и удаления ферзя принимают координаты самого ферзя (i, j) и не возвращают значений.  
      
   3. Функция проверки следующей строки на возможность поставить ферзя возвращает значение типа **bool**. В качетсве аргумента функции выступает текущая строка (i).  
      
2. Для решения задачи необходимо:
   1. Разработать функцию установки ферзя (setQ).
   2. Разработать функцию удаления ферзя с позиции (resetQ).
   3. Разработать функцию, проверяющую следующую строку на возможность поставить ферзя (tryQ).
3. Для решения задачи данные были представлены в следующем виде:
   1. Поле шахматной доски представлено в виде матрицы 8x8 (см.п.1.1).
4. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Вывод происходит в функции main() с помощью двух циклов for().



* 1. Функция не требует вводить какие-либо данные.

1. Поставленные задачи будут решены следующими действиями:
   1. Функция setQ заполняет клетки главной и побочной диагонали атаки ферзя, его расположение, клетки столбца и строки. Это реализовано с помощью цикла for. Заполнение ряда происходит с помощью индексации board[x][j], где j-ряд, в котором стоит ферзь, x - управляющая переменная цикла. Значение в этих ячейках инкрементируется. Аналогично заполнению ряда происходит заполнение строки, за одним лишь исключением - индексация board[i][x]. Для заполнения диагоналей используются операторы ветвления if, с условием k >= 0 && k < 8, где k для главной диагонали рассчитывается по формуле k = j - i + x, а для побочной k = j + i – x. После выхода из цикла ячейке с индексом i, j присваивается значение -1.  
      
   2. Аналогично п.5.1 функция resetQ удаляет ферзя, за исключением того, что значение клеток, находящихся под боем, декрементируется, значению ячейки с индексами i, j присваивается значение **0**.  
      
   3. Функция tryQ использует переменную типа bool, значение которой изначально равно false. Функция работает по принципу рекурсии. Перебор клеток осуществляется с помощью цикла for, если найдена клетка со значением 0, то вызывается функция setQ, которая ставит ферзя в позицию i строки j столбца (j - управляющая переменная цикла). Если значение i равно 7, то функция прекращает свою работу (все ферзи расставлены). Иначе, вызывается функция tryQ с параметром i+1. Если функция возвращает значение false, то ферзь удаляется, и перебор строки продолжается.  
      
   4. Вывод доски происходит с помощью двух циклов for().  
      

**Код программы**

#include <iostream>

using namespace std;

int board[8][8];

void resetQ(int i, int j)

{

for (int x = 0; x < 8; ++x)

{

--board[x][j];

--board[i][x];

int k;

k = j - i + x;

if (k >= 0 && k < 8)

--board[x][k];

k = j + i - x;

if (k >= 0 && k < 8)

--board[x][k];

}

board[i][j] = 0;

}

void setQ(int i, int j)

{

for (int x = 0; x < 8;x++)

{

++board[x][j];

++board[i][x];

int k;

k = j - i + x;

if (k >= 0 && k < 8)

++board[x][k];

k = j + i - x;

if (k >= 0 && k < 8)

++board[x][k];

}

board[i][j] = -1;

}

bool tryQ(int i)

{

bool result = false;

for (int j = 0; j < 8; ++j)

{

if (board[i][j] == 0)

{

setQ(i, j);

if (i == 7)

result = true;

else

{

if (!(result = tryQ(i + 1)))

resetQ(i, j);

}

}

if (result)

break;

}

return result;

}

int main()

{

for (int i = 0; i < 8; ++i)

for (int j = 0; j < 8; ++j)

board[i][j] = 0;

tryQ(0);

for (int i = 0; i < 8; ++i)

{

for (int j = 0; j < 8; ++j)

{

if (board[i][j] == -1)

cout << "[] ";

else

cout << ". ";

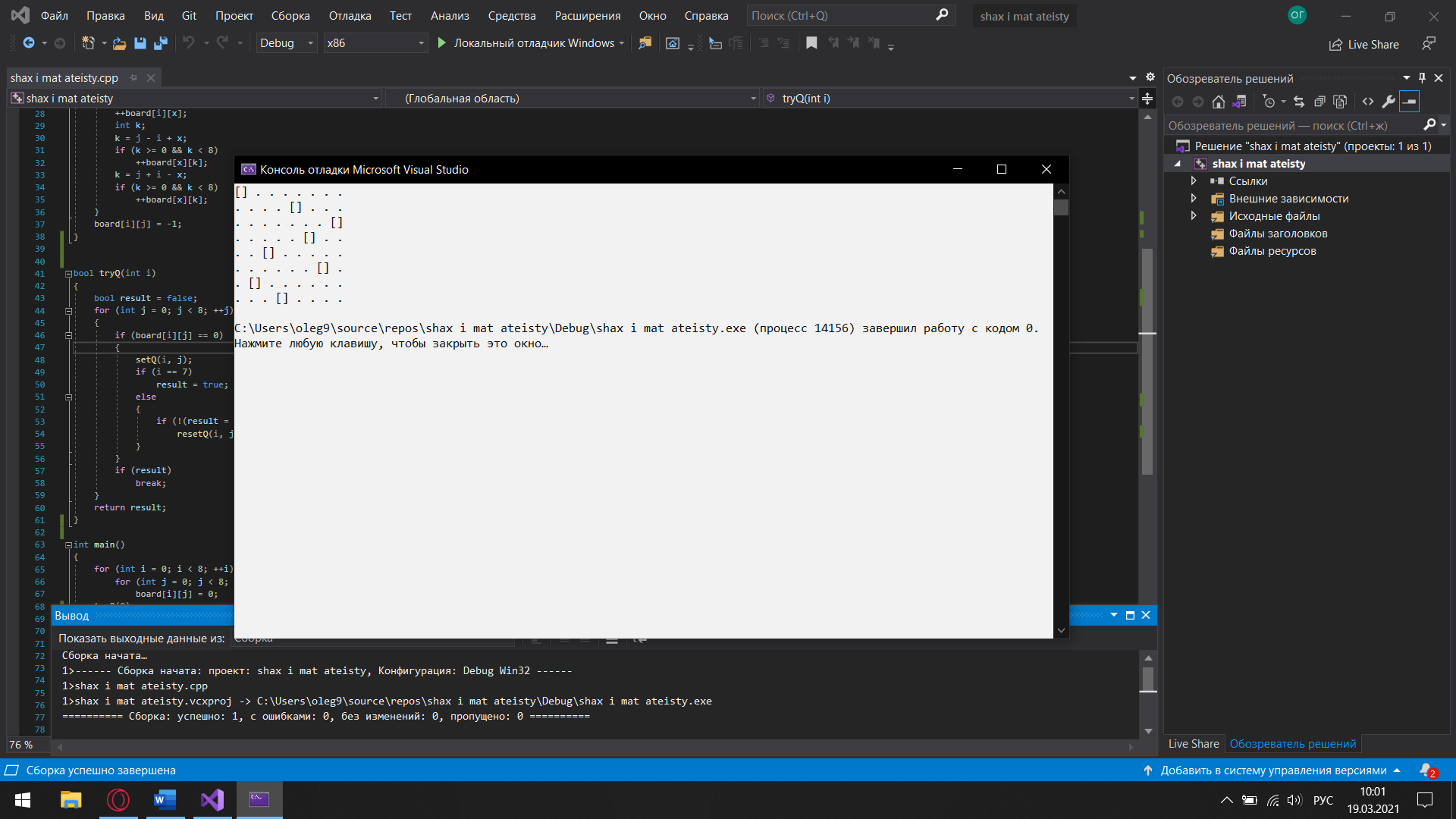
}

cout << endl;

}

}

**Работа программы**



**Блок-схема**

