**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра інтелектуальних технологій**

**Лабораторна робота №10**

з дисципліни «Алгоритмізація і програмування»

Тема роботи:

«Багатовимірні масиви даних бібліотеки STL»

**Варіант № 1**

Виконав студент

групи КН-11:

**Нестеренко Дмитро Дмитрович**

Перевірила: **Москаленко Наталія Володимирівна**

**Київ – 2022**

**Завдання 1**

Задана матриця розміром КхР (К та Р вводить користувач). Видалити з

матриці: 1) рядок, що містить максимальний елемент матриці; 2) стовбець,

який містить значення, що було найближчим до максимального.

**1.1. Тестові приклади роботи програми:**

1.K=3, P=4. Формуємо рандомну матрицю 3х4. Видаляємо 2 рядок та 1 стовпчик.

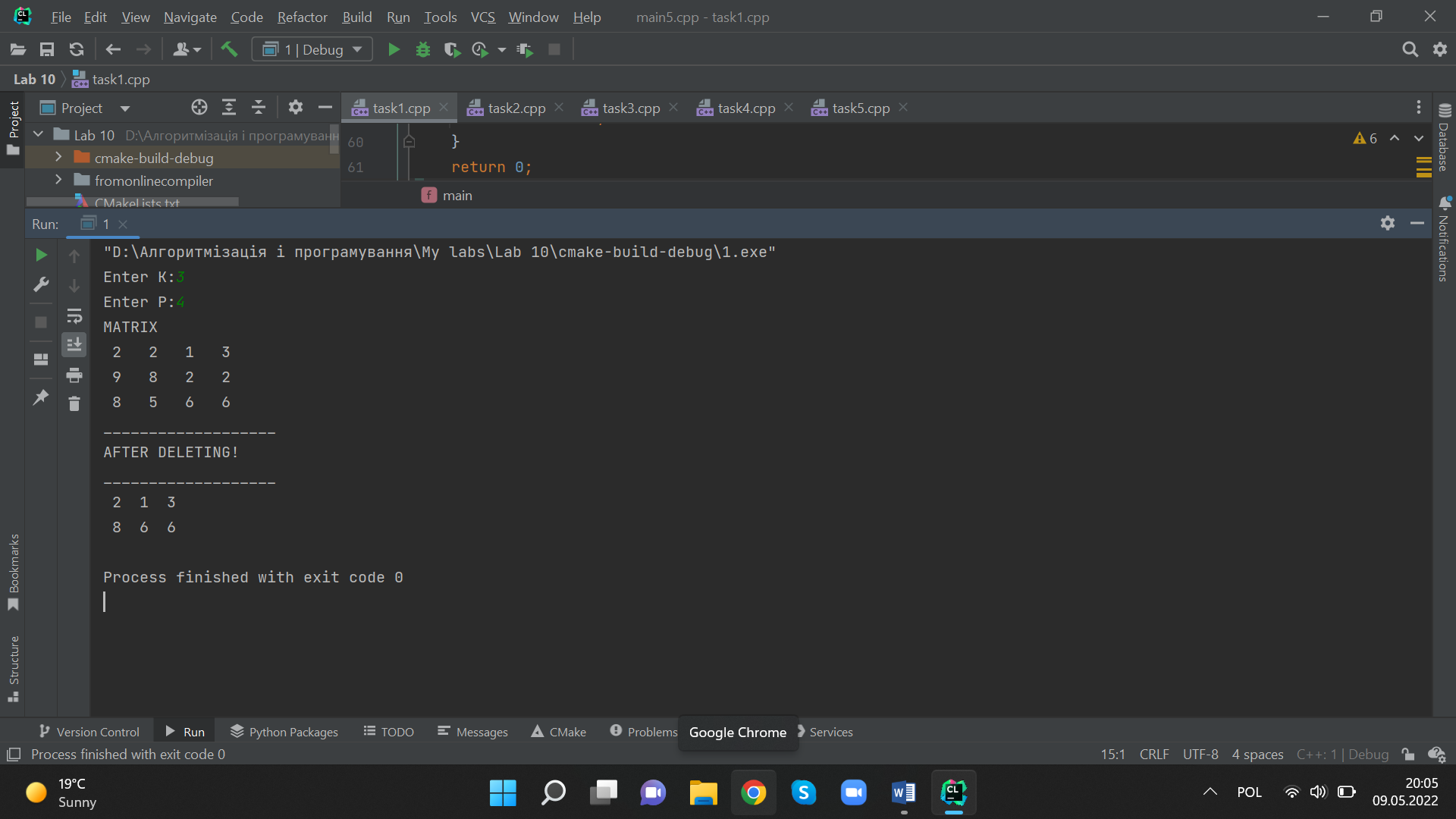


Рисунок 1 – Тестовий приклад до завдання 1

2. K=4, P=4. Формуємо рандомну матрицю 4х4. Видаляємо 1 рядок та 1 стовпчик.

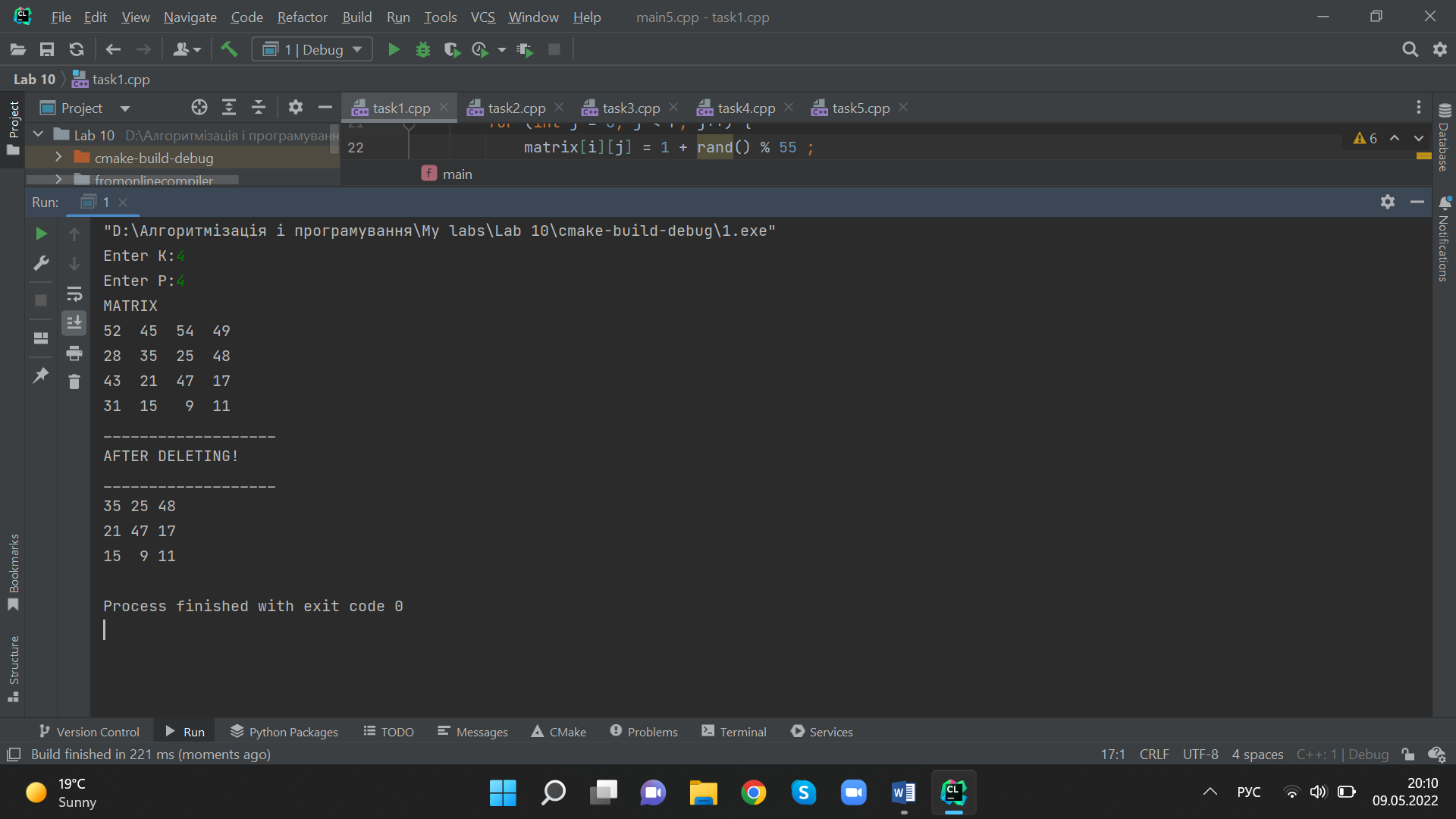


Рисунок 2 - Тестовий приклад до завдання 1

Висновок: результати роботи програми підтверджено тестовими розрахунками.

**1.2. Текст програми:**

#include<iostream>

#include <vector>

#include <ctime>

#include <cstdlib>

#include <iomanip>

#include <algorithm>

using namespace std;

int main() {

srand(time(0));

int K, P;

cout << "Enter K:";

cin >> K;

cout << "Enter P:";

cin >> P;

vector<vector<int>> matrix(K, vector<int>(P));

cout << "MATRIX" << endl;

for (int i = 0; i < K; i++) {

for (int j = 0; j < P; j++) {

matrix[i][j] = 1 + rand() % 9 ;

cout << setw(2) << matrix[i][j] << " ";

}

cout << endl;

}

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

int max = matrix[0][0];

int row = 0, col = 0;

for (int i = 0; i < K; i++) {

for (int j = 0; j < P; j++) {

if (matrix[i][j] > max) {

max = matrix[i][j];

row = i;

}

}

}

int preMax=0;

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

for (int j = 0; j < matrix[i].size(); j++) {

if (preMax < matrix[i][j] && matrix[i][j]<max) {

preMax = matrix[i][j];

col = j;

}

}

}

//delete row with max

matrix.erase(matrix.begin() + row);

//delete column

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

matrix[i].erase(matrix[i].begin() + col);

}

cout<<"AFTER DELETING!\n";

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

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

for (int j = 0; j < matrix[i].size(); j++) {

cout <<setw(2)<< matrix[i][j] << " ";

}

cout << endl;

}

return 0;

}

**Завдання 2**

Задана матриця розміром КхР (К та Р вводить користувач). 1) Після рядка,

що містить всі додатні значення вставити рядок з нулями. 2) Перед стовпцем,

елементи якого утворюють спадну послідовність вставити стовпець з

одиницями.

**2.1. Тестові приклади роботи програми:**

1.Матриця 2х3. Додаємо відповідні рядки та стовпці.

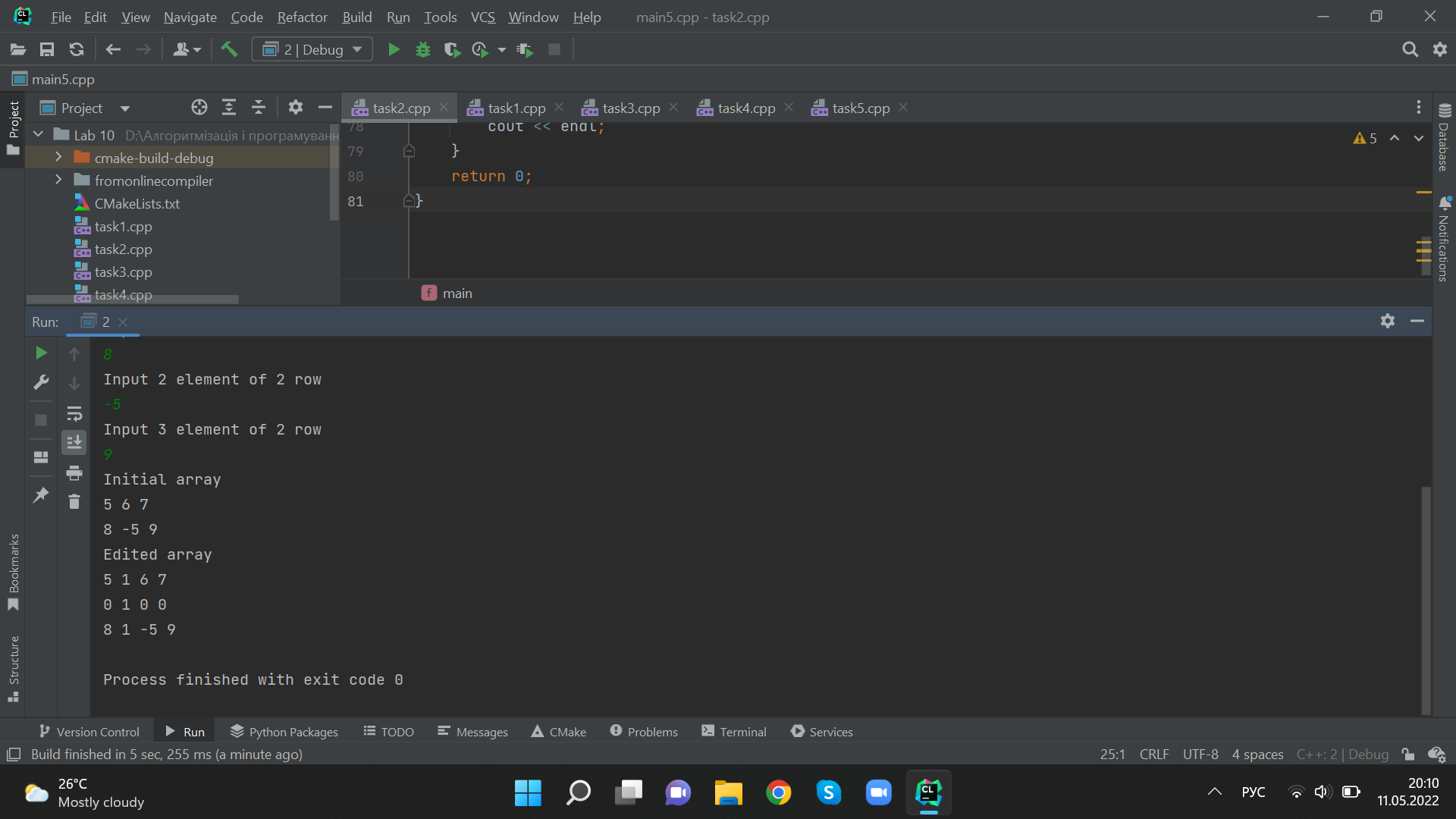
****

Рисунок 3 – Тестовий приклад до завдання 2

2. Матриця 2х2. Додаємо відповідні рядки та стовпці.

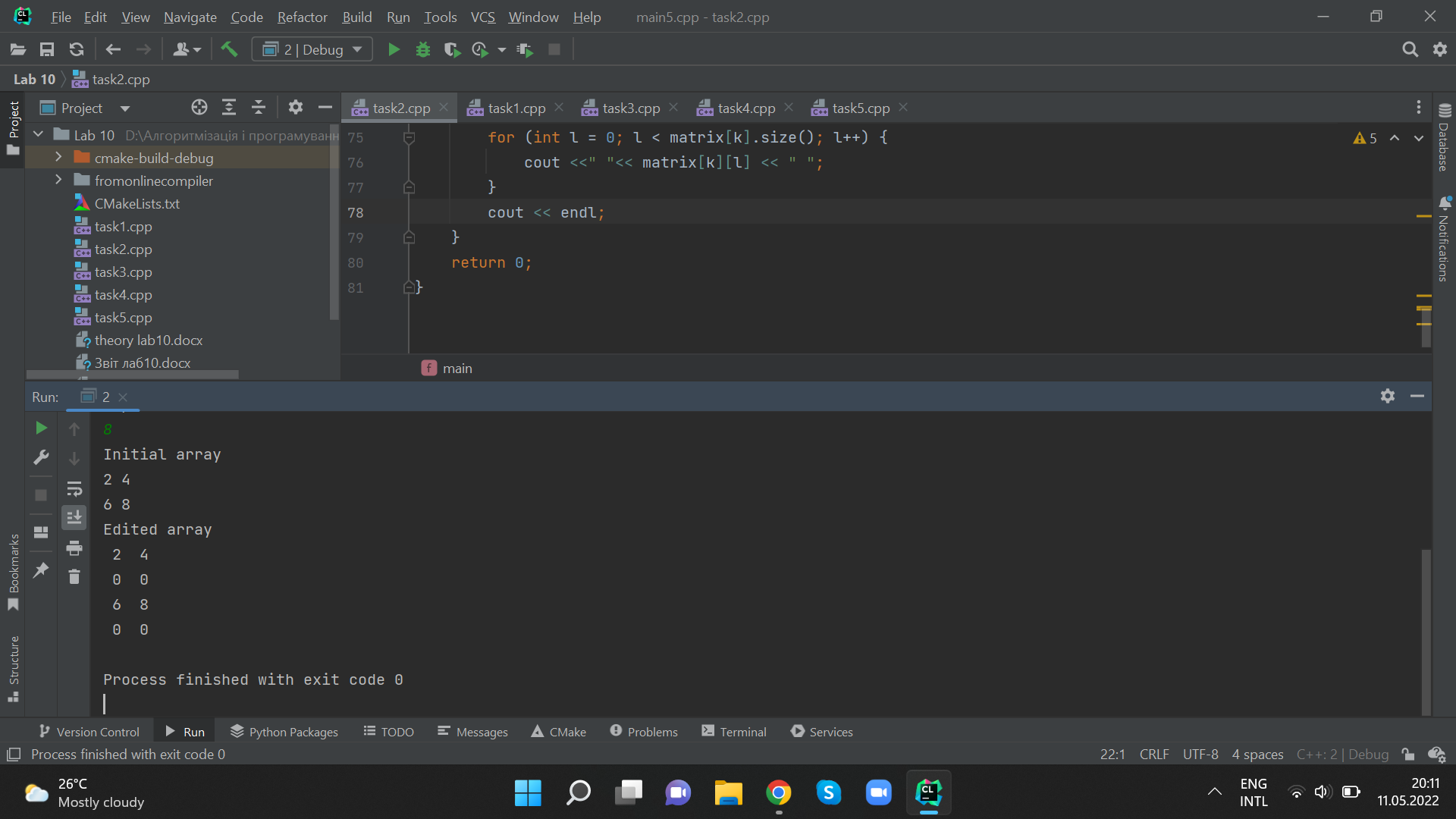


Рисунок 4- Тестовий приклад до завдання 2

Висновок: результати роботи програми підтверджено тестовими розрахунками.

**2.2. Текст програми:**

#include <iostream>

#include <vector>

using namespace std;

int main()

{

int row;

int col;

bool isPositive = true;

bool isNegative = true;

int counter = 0;

int additional = 1;

cout << "Input rows ";

cin >> row;

cout << "Input cols ";

cin >> col;

vector<vector<int>>matrix(row, vector<int>(col));

vector<int>rowPositive;

vector<int>colNegative;

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

for (int j = 0; j < matrix[i].size(); j++) {

cout << "Input " << j + 1 << " element of " << i + 1 << " row" << endl;

cin >> matrix[i][j];

}}

cout << "Initial array" << endl;

for (int k = 0; k < matrix.size(); k++) {

for (int l = 0; l < matrix[k].size(); l++) {

cout << matrix[k][l] << " ";

}

cout << endl;}

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

for (int j = 0; j < matrix[i].size() && isPositive; j++) {

counter += 1;

if (matrix[i][j] < 0) {

isPositive = false;

}

}

if (counter == col && isPositive) {

rowPositive.push\_back(i);

}

if (!isPositive) {

isPositive = true;

}

counter = 0;}

for (int k = 0; k < col; k++) {

counter = 1;

for (int l = 1; l < row; l++) {

counter += 1;

if (matrix[l][k] >= matrix[l-1][k]) {

isNegative = false;

}

}

if (counter == row && isNegative) {

colNegative.push\_back(k);

}

if (!isNegative) {

isNegative = true;

}

}

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

matrix.insert(matrix.begin() + rowPositive[i] +additional, vector<int>(col, 0));

additional+=1;

}

additional = 0;

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

for (int j = 0; j < matrix.size(); j++) {

matrix[j].insert(matrix[j].begin() + colNegative[i] + additional, 1);

}

additional += 1;

}

cout << "Edited array" << endl;

for (int k = 0; k < matrix.size(); k++) {

for (int l = 0; l < matrix[k].size(); l++) {

cout << matrix[k][l] << " ";

}

cout << endl;

}

return 0;

}

**Завдання 3**

Задана матриця розміром КхР (К та Р вводить користувач). 1) Поміняти

місцями рядок, що містить всі парні елементи з рядком, що містить всі непарні

елементи (вважати, що такі рядки існують по одному). 2) Поміняти місцями

стовпці, що містять мінімальний та максимальний елементи.

**3.1. Тестові приклади роботи програми:**

1. Заповнюємо елементами матрицю 2х3. Міняємо рядки та стовпчики, виводимо змінену матрицю.

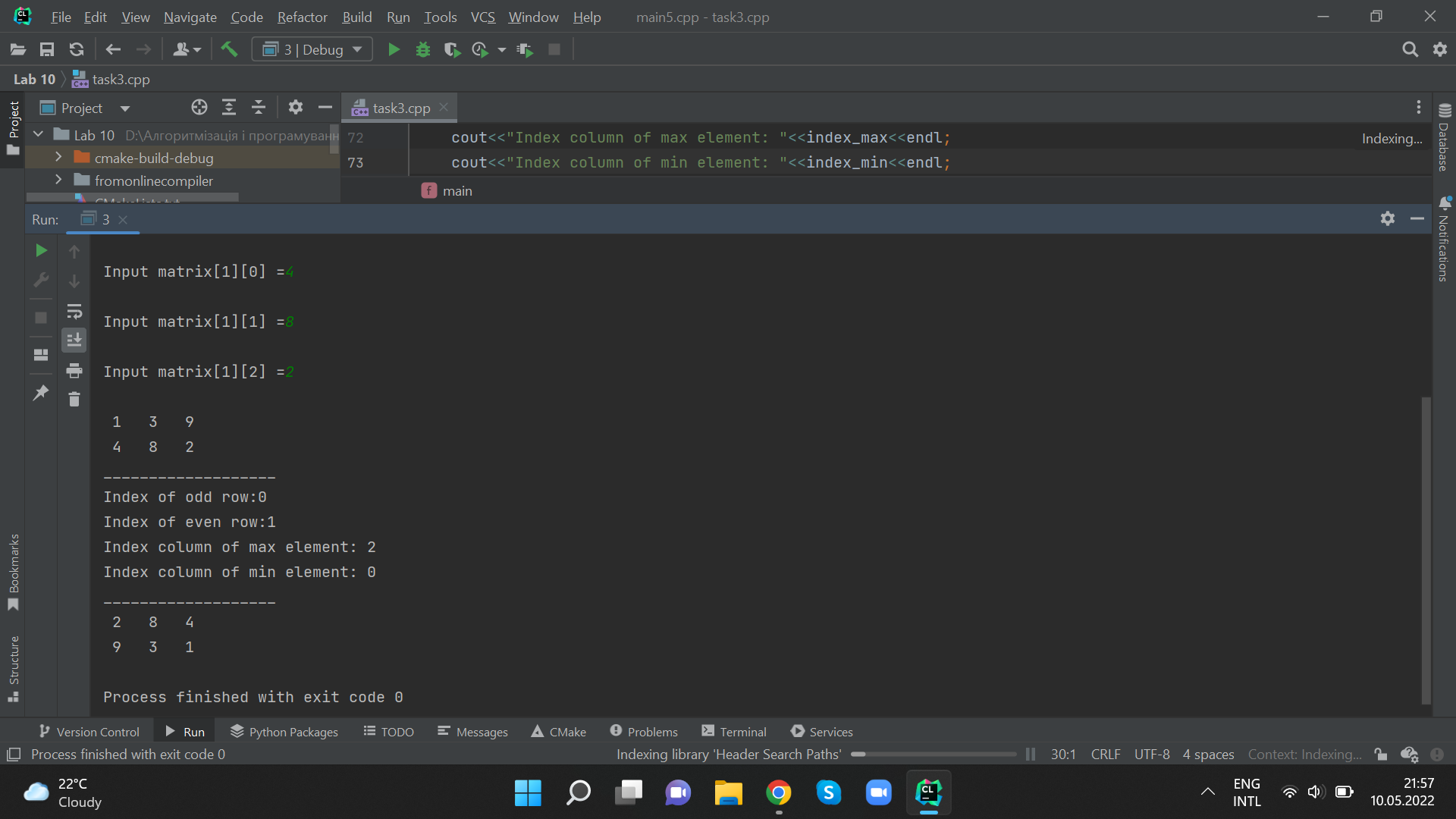
****

Рисунок 5– Тестовий приклад до завдання 3

2. Заповнюємо елементами матрицю 3х3. Міняємо рядки та стовпчики, виводимо змінену матрицю.

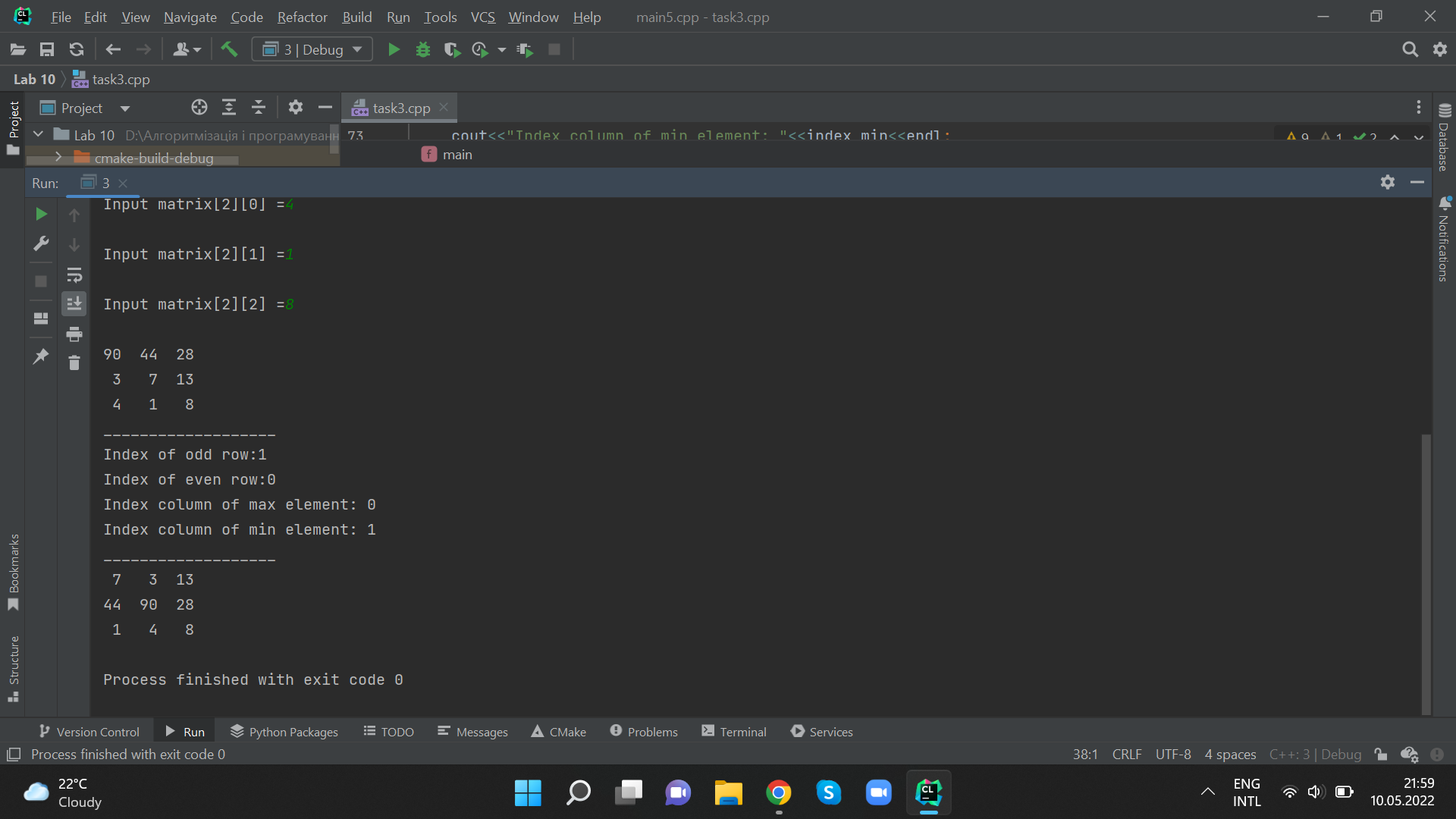


Рисунок 6 - Тестовий приклад до завдання 3

Висновок: результати роботи програми підтверджено тестовими розрахунками.

**3.2. Текст програми:**

#include<iostream>

#include <vector>

#include <ctime>

#include <cstdlib>

#include <iomanip>

#include <algorithm>

using namespace std;

int main() {

srand(time(0));

int K, P;

cout << "Enter K:";

cin >> K;

cout << "Enter P:";

cin >> P;

vector<vector<int>> matrix(K, vector<int>(P));

cout <<"MATRIX"<< endl;

for (int i = 0; i < K; i++) {

for (int j = 0; j < P; j++) {

cout << "\nInput matrix["<<i<<"]["<<j<<"] = ";

cin >> matrix[i][j];

}

}

cout<<endl;

//show matrix:)

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

for (int j = 0; j < matrix[i].size(); j++) {

cout << setw(2) << matrix[i][j] << " ";

}

cout << endl;

}

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

//odd or even rows

int index\_odd=-1,index\_even=-1;

for (int i = 0; i < K; i++) {

bool even = 1,odd = 1;

for (int j = 0; j < P; j++) {

if (matrix[i][j]%2 == 0) {

odd = 0;

}

if (matrix[i][j] % 2 == 1) {

even = 0;

}

}

if (odd) {

index\_odd = i;

}

if (even) {

index\_even = i;

}

}

cout<<"Index of odd row:"<<index\_odd<<endl;

cout<<"Index of even row:"<<index\_even<<endl;

// min, max - тесты успешны!

int min = matrix[0][0], max = matrix[0][0];

int index\_min=0, index\_max=0;

for (int i = 0; i < K; i++) {

for (int j = 0; j < P; j++) {

if (matrix[i][j] > max) {

max = matrix[i][j];

index\_max = j;

}

if (matrix[i][j] < min) {

min = matrix[i][j];

index\_min = j;

}

}

}

cout<<"Index column of max element: "<<index\_max<<endl;

cout<<"Index column of min element: "<<index\_min<<endl;

if (index\_odd<0||index\_even<0){

cout<<"Please, input elements due to task"<<endl;

}

//change rows

matrix[index\_even].swap(matrix[index\_odd]);

//change columns

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

swap( matrix[i][index\_max], matrix[i][index\_min] );

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

//show matrix after changing columns and rows

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

for (int j = 0; j < matrix[i].size(); j++) {

cout << setw(2) << matrix[i][j] << " ";

}

cout << endl;

}

return 0;

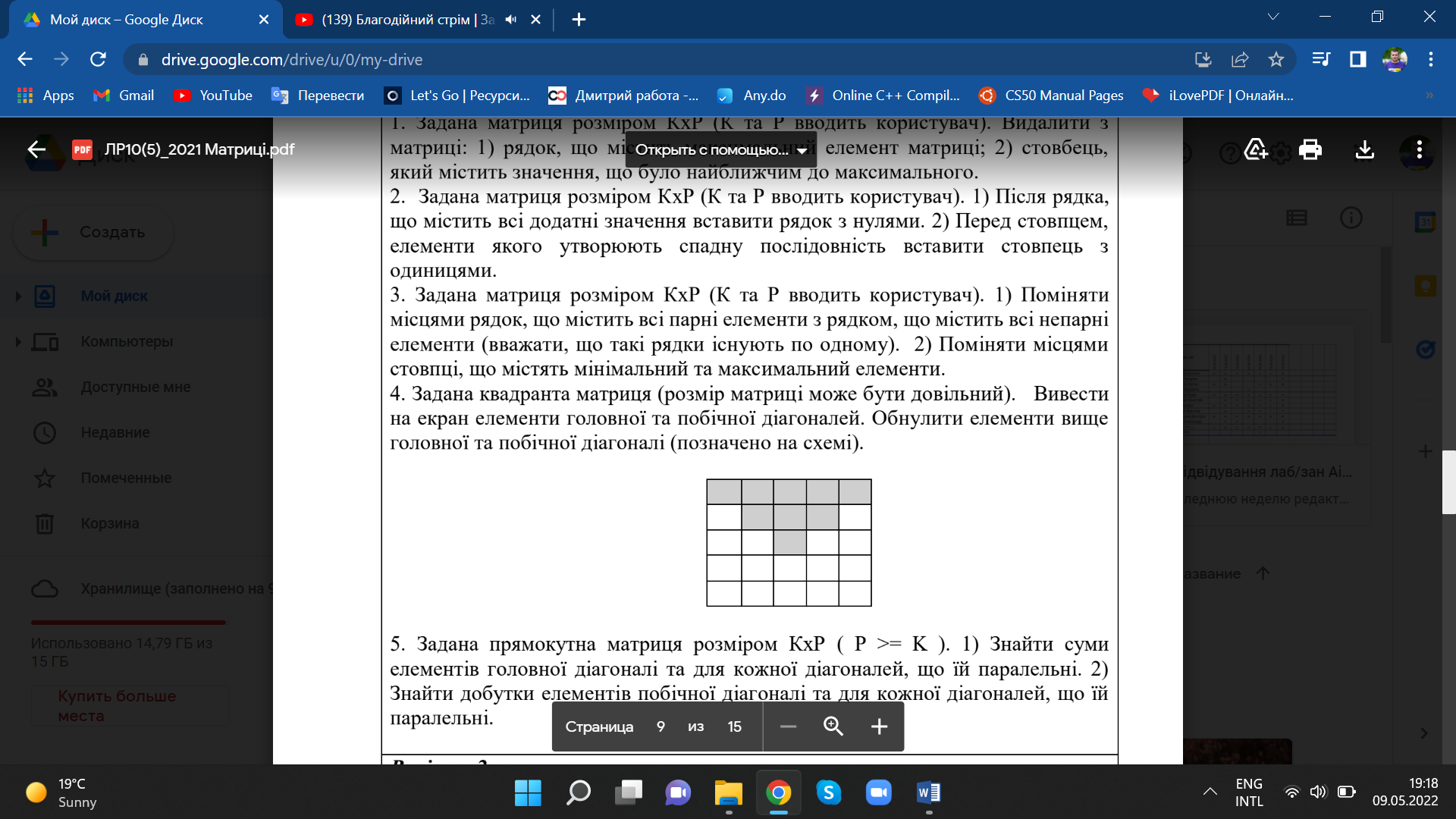
}

**Завдання 4**

Задана квадранта матриця (розмір матриці може бути довільний). Вивести

на екран елементи головної та побічної діагоналей. Обнулити елементи вище

головної та побічної діагоналі (позначено на схемі).



**4.1. Тестові приклади роботи програми:**

1. Утворимо рандомну матрицю, знайдемо елементи головної та побічної діагоналі, заповнимо нулями простір за умовою.

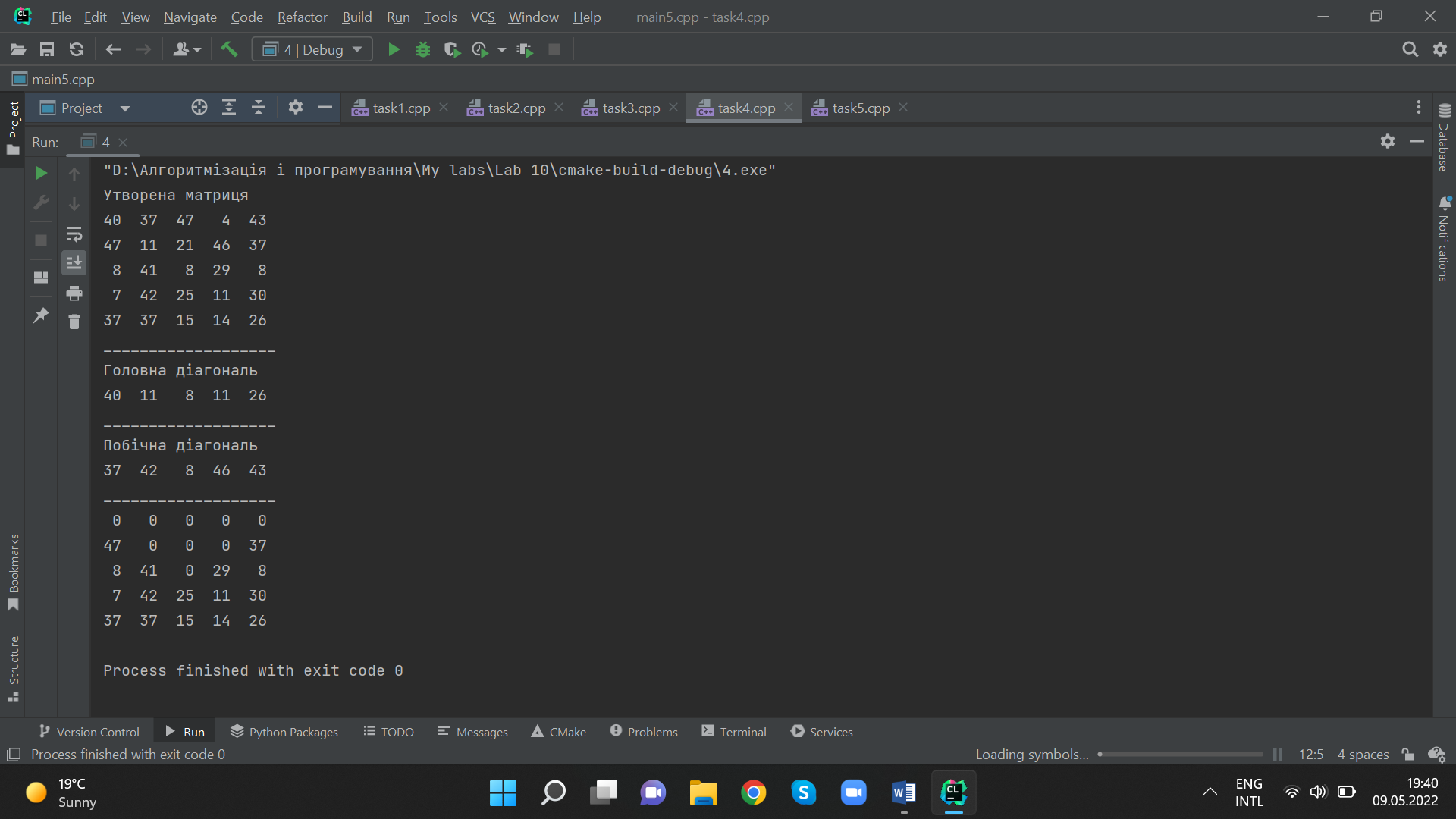


Рисунок 7 - Тестовий приклад до завдання 4

Висновок: результати роботи програми підтверджено тестовими розрахунками.

**4.2. Текст програми:**

#include <iostream>

#include <iomanip>

#include <vector>

#include <ctime>

#include <cstdlib>

using namespace std;

int main() {

int N = 5; //кількість рядків

int M = N; //кількість стовпців

srand(time(0));

vector <vector<int> > A(N, vector<int>(M, 0));

cout <<"Утворена матриця"<< endl;

for (int i = 0; i < N; i++) {

for (int j = 0; j < M; j++) {

A[i][j] =1+ rand() % 50;

cout << setw(2) << A[i][j] << " ";

}

cout << endl;

}

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout <<"Головна діагональ"<< endl;

for (int i = 0; i < N; i++) {

cout << setw(2) << A[i][i] << " " ;

}

cout << endl ;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout <<"Побічна діагональ"<< endl;

for (int i = N-1; i >=0; i--) {

cout << setw(2) << A[i][N - i - 1] << " " ;

}

cout << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

for (int i = 0; i < N; i++) {

for (int j = 0; j < M; j++) {

if ((i <= j) && (i <= N - 1 - j)) {

A[i][j] = 0;

}

cout << setw(2) << A[i][j] << " ";

}

cout << endl;

}

return 0;}

**Завдання 5**

Задана прямокутна матриця розміром КхР ( P >= K ). 1) Знайти суми

елементів головної діагоналі та для кожної діагоналей, що їй паралельні. 2)

Знайти добутки елементів побічної діагоналі та для кожної діагоналей, що їй

паралельні.

**5.1. Тестові приклади роботи програми:**

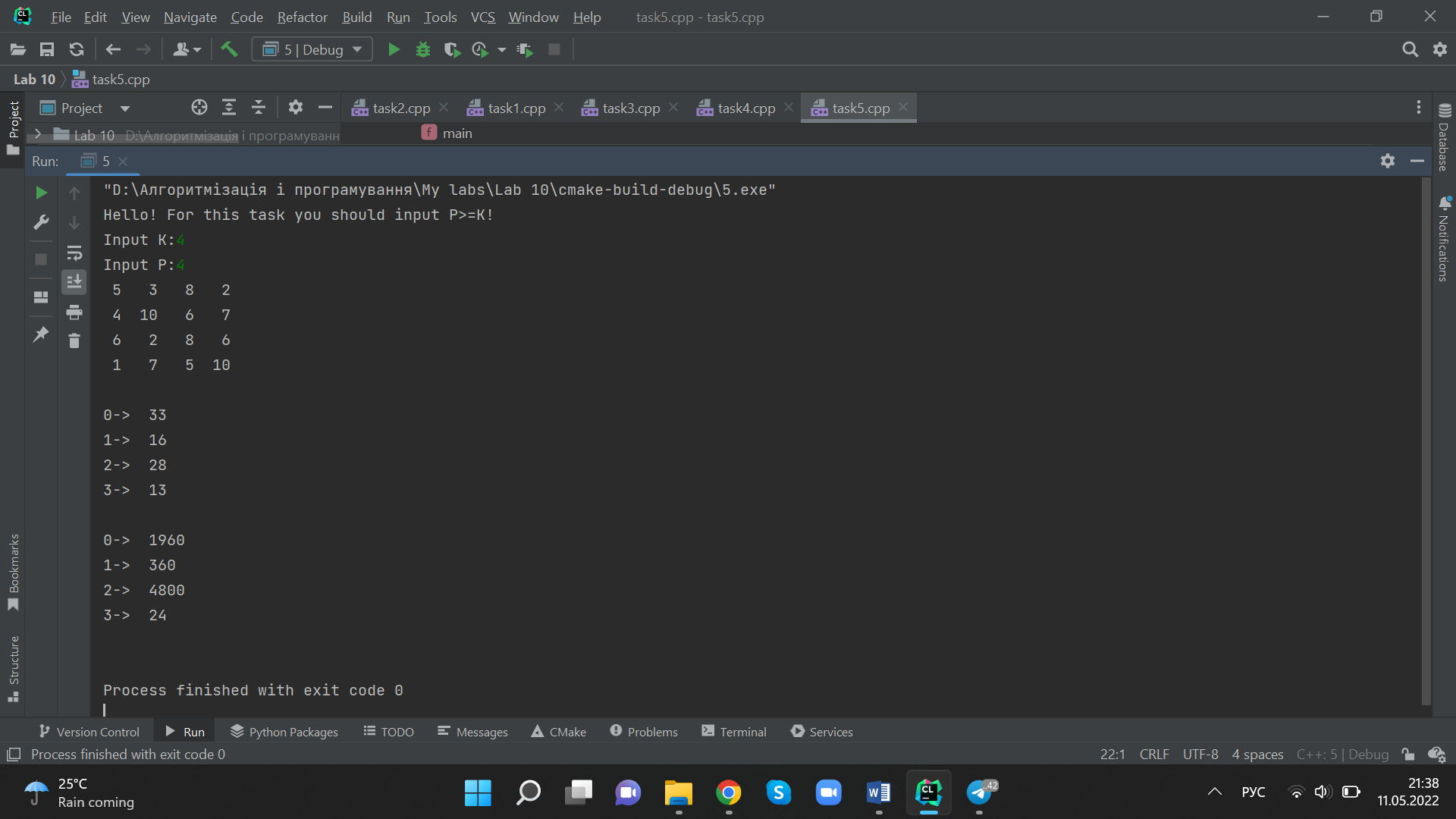


Рисунок 8 - Тестовий приклад до завдання 5

Висновок: результати роботи програми підтверджено тестовими розрахунками.

**5.2. Текст програми:**

#include <iostream>

#include <iomanip>

#include <vector>

#include <ctime>

#include <cstdlib>

using namespace std;

int main() {

int K = 4; //кількість рядків

int P = 5; //кількість стовпців

int r;

srand(time(0));

cout<< "Hello! For this task you should input P>=K!"<<endl;

cout << "Input K:";

cin >> K;

cout << "Input P:";

cin >> P;

vector <int> goldiagonal(P), pobdiagonal(P);

vector <vector<int> > matrix(K, vector<int>(P, 0));

for (int i = 0; i < K; i++) {

for (int j = 0; j < P; j++) {

matrix[i][j] = 1 + rand() % 50;

cout << setw(2) << matrix[i][j] << " ";

}

cout << endl;

}

cout << endl;

for (int j = 0; j < P; j++) {

for (int i = 0; i < K; i++) {

goldiagonal[j] += matrix[i][(i + j) % P];

}

}

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

cout << i << "-> " << goldiagonal[i] << endl;

}

cout << endl;

for (int j = P-1; j >=0; j--) {

pobdiagonal[j] = 1;

for (int i = 0; i < K; i++) {

pobdiagonal[j] \*= matrix[i][(j-i+P) % P];

}

}

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

cout << i << "-> " << pobdiagonal[i] << endl;

}

cout << endl;

return 0;

}