МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп'ютерні інформаційні технології»

**Лабораторна робота №5**

**з дисципліни «Основи програмної інженерії»**

**на тему: «**Розробка та створення діалогу типу «питання-відповідь»**»**

Виконав:

студент гр.ПЗ1911

Сафонов Д.Є.

Прийняла:

Куроп'ятник О. С.

Дніпро, 2020

**Тема.** Розробка та створення діалогу типу «питання-відповідь».

**Мета.** Отримати практичні навички створення діалогу типу «питання-відповідь».

**1. Постановка задачі згідно загального та індивідуального завданнями.**

Для задачі лабораторної роботі №1 виконайте:

− побудуйте сценарій ведення діалогу;

− розробіть алгоритм та напишіть програму з діалогом типу «питання відповідь» згідно розробленого сценарію.

17. Відсортуйте в матриці стовпці, що містять елементи, в запису яких є однакові цифри. Інші стовпці переверніть.

**2. Сценарій діалогу програми.**

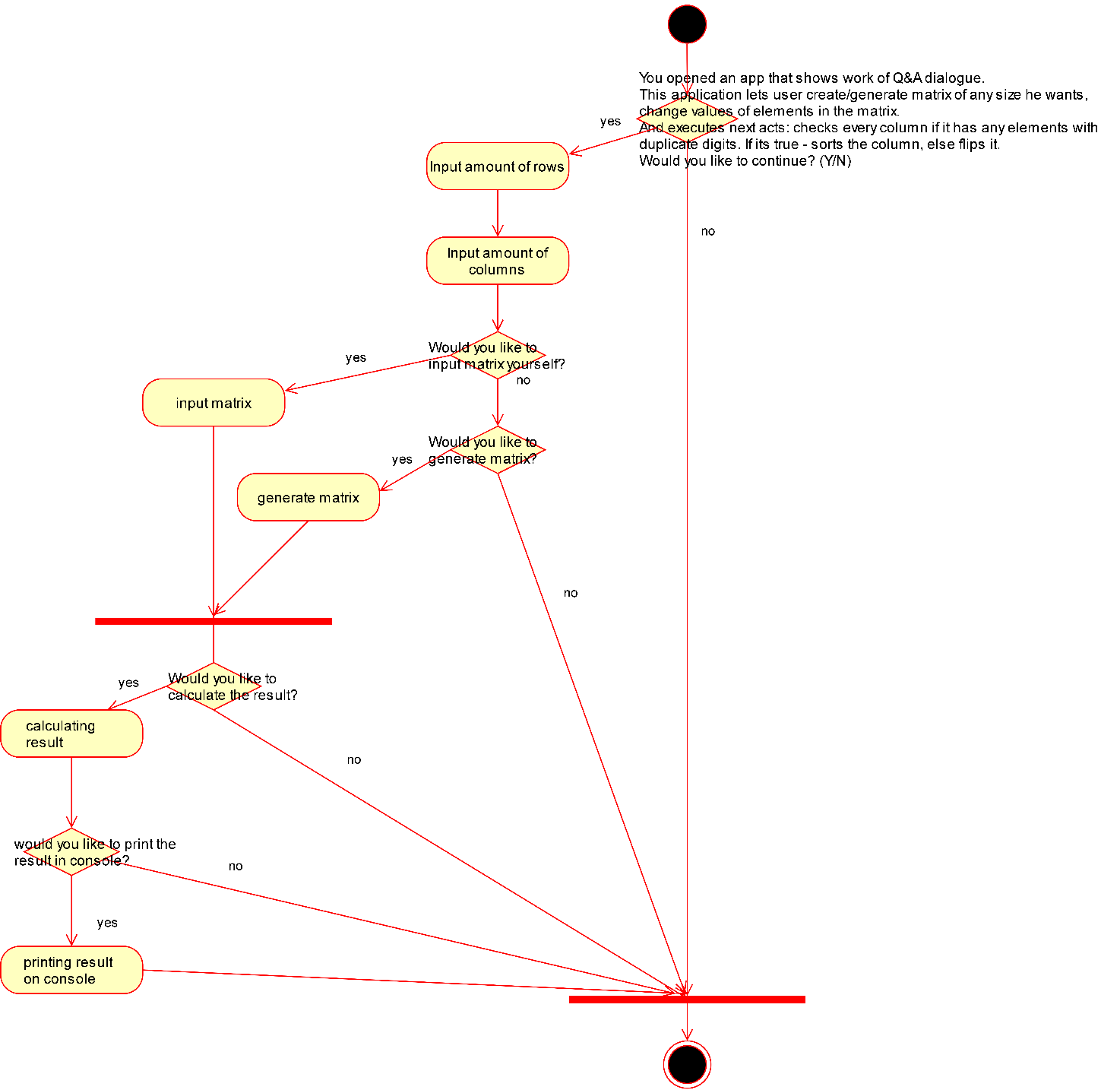


Рисунок 1

**3. Набори тестів для перевірки працездатності програми.**

Перевірятись будуть лише переходи по розгалуженням, бо розрахунки вже перевірені в ЛР1.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва | Вхідні дані | Очікуваний результат |
| Будуть вказані відповіді для переходу для розгалужень.  Для компактності послідовності, які приходять різними путями до одного розгалуження будуть вказані разом, але будуть виконані обидва тести | | | |
| 1 |  | n | Привітання |
| 2 |  | y n n | Привітання,  Запит про введеня матриці,  Запит про генерацію матриці. |
| 3 |  | y y n  y n y n | Введення/генерація матриці. |
| 4 |  | y y y n  y n y y n | Введення/генерація матриці,  Розрахунок результату. |
| 5 |  | y y y y  y n y y y | Введення/генерація матриці,  Розрахунок результату,  Виведення результату. |

**4. Код програми**

“main.cpp”

#include "solution.h"

int main() {

solveOPI5();

return 0;

}

“matrix.h”

#ifndef \_\_MATRIX\_H\_\_

#define \_\_MATRIX\_H\_\_

#include <iostream>

#include <iomanip>

#include <random>

#include <chrono>

#include <bitset>

#include <string>

//allocate memory for matrix

template<typename T>

T\*\* createMatrix(int rowsAmount, int columnsAmount) {//allocate memory for matrix

T\*\* matrix2d = new T \* [rowsAmount];

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

matrix2d[i] = new T[columnsAmount];

return matrix2d;

}

//free up the memory

template<typename T>

void deleteMatrix(T\*\* matrix2d, int rowsAmount, int columnsAmount) {

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

delete[]matrix2d[i];

delete[]matrix2d;

}

//set random values for matrix

template<typename T>

void genMatrix(T\*\* matrix2d, int rowsAmount, int columnsAmount) {

typedef std::chrono::high\_resolution\_clock myclock;

myclock::time\_point beginning = myclock::now();

std::uniform\_int\_distribution<int> valueDistribution(0, 100);

myclock::duration d = myclock::now() - beginning;

unsigned seed = d.count();

std::default\_random\_engine generator(seed);

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

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

matrix2d[i][j] = valueDistribution(generator);

}

template<typename T>

void printMatrix(T\*\* matrix2d, int rowsAmount, int columnsAmount) {

std::cout << std::left;

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

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

std::cout << std::setw(12) << matrix2d[i][j] << " ";

std::cout << std::endl;

}

std::cout << std::right;

}

//sorts column using insertion sort algorithm

template <typename T>

void sortColumn(T\*\* matrix2d, int column, int rowsAmount) {

for (int i = 1; i < rowsAmount; i++)

if (matrix2d[i][column] < matrix2d[i - 1][column]) {//insertionSort

T temp = matrix2d[i][column];

int j = i - 1;

for (j; (j >= 0) && (temp < matrix2d[j][column]); j--)

matrix2d[j + 1][column] = matrix2d[j][column];//find position and shift to right elements between new and old position

matrix2d[j + 1][column] = temp;

}

}

template <typename T>

void flipColumn(T\*\* matrix2d, int column, int rowsAmount) {

for (int i = 0, j = rowsAmount - 1; i <= j; i++, j--) {

T temp = matrix2d[i][column];

matrix2d[i][column] = matrix2d[j][column];

matrix2d[j][column] = temp;

}

}

bool hasDuplicateDigits(int a) {

bool retValue = false;

a = abs(a);

if (a > 999999999)//>10digits, at least one is duplicate

retValue = true;

else if (a > 10) {//if >= 2digits

std::bitset<10> digitsUsed;

int currentDigit;

do {

currentDigit = a % 10;

a = (a - currentDigit) / 10;

if (digitsUsed[currentDigit])

retValue = true;

else

digitsUsed[currentDigit] = true;

} while (a > 0 && !retValue);

}

return retValue;

}

//checks if column has any elemnts with duplicate digits

template <typename T>

bool columnElementsDuplicateDigits(T\*\* matrix2d, int column, int rowsAmount) {

bool has = false;

T currentElement;

for (int i = 0; i < rowsAmount && !has; i++)//has = if matrix2d[i][column] has >= 2 same numerals

has = hasDuplicateDigits(matrix2d[i][column]);

return has;

}

//performs a check on every column and sorts it if it has any elements with duplicate digits, flips if not

template <typename T>

void exercise(T\*\* matrix2d, int rowsAmount, int columnsAmount) {

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

if (columnElementsDuplicateDigits(matrix2d, i, rowsAmount))

sortColumn(matrix2d, i, rowsAmount);

else

flipColumn(matrix2d, i, rowsAmount);

}

int inputNum(int leftLim, int rightLim, std::string invitation) {

std::string cinStr;//////////////////////////////////////////

int num = leftLim - 1;

size\_t\* pos = new size\_t{ 0 };

do {//input validation

std::cout << invitation;

std::cin >> cinStr;

if (std::cin.peek() != '\n') {//if istream isn't empty

while (std::cin.get() != '\n');//clear istream

std::cout << "incorrect format" << std::endl;

}

else {

\*pos = 0;//clear old

try {

num = std::stoi(cinStr, pos);//convert input to INT format, set \*pos value to length of used substring

}

catch (std::invalid\_argument) {

std::cout << "incorrect format" << std::endl;

num = leftLim - 1;

}

catch (std::out\_of\_range) {

std::cout << "out of range" << std::endl;

num = leftLim - 1;

}

if (\*pos != 0)

if (\*pos < cinStr.size()) {

std::cout << "incorrect format" << std::endl;

num = leftLim - 1;

}

else if (num < leftLim || num >= rightLim) {

std::cout << "out of range" << std::endl;

num = leftLim - 1;

}

}

} while (num < leftLim || num >= rightLim);

return num;

}

//lets user to chnge any elements by indexes

template <typename T>

void setMatrix(T\*\* matrix2d, int rowsAmount, int columnsAmount) {

system("cls");

printMatrix(matrix2d, rowsAmount, columnsAmount);

char choice;

std::cout << "Do you want to enter any elements?(Y/N)";

std::cin >> choice;

while ((choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') || (std::cin.peek() != '\n')) {

while (std::cin.get() != '\n');//clear istream

std::cout << "wrong input, try again." << std::endl;

std::cout << "Do you want to enter any elements?(Y/N)";

std::cin >> choice;

}

while (choice != 'N' && choice != 'n') {

int i = inputNum(0, rowsAmount, "enter row of element you want to change: ");

int j = inputNum(0, columnsAmount, "enter column of element you want to change: ");

std::string elementName = "A[" + std::to\_string(i) + "][" + std::to\_string(j) + "] = ";

matrix2d[i][j] = inputNum(-32768, 32767, elementName);

system("cls");

printMatrix(matrix2d, rowsAmount, columnsAmount);

std::cout << "Do you want to enter any elements?(Y/N)";

std::cin >> choice;

while ((choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') || (std::cin.peek() != '\n')) {

while (std::cin.get() != '\n');//clear istream

std::cout << "wrong input, try again." << std::endl;

std::cout << "Do you want to enter any more elements?(Y/N)";

std::cin >> choice;

}

}

}

//set all elements to 0

template <typename T>

void setMatrix0(T\*\* matrix2d, int rowsAmount, int columnsAmount) {

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

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

matrix2d[i][j] = 0;

}

#endif

“solution.h”

#ifndef \_\_SOLUTION\_H\_\_

#define \_\_SOLUTION\_H\_\_

//asks <question>, if answer isnt (Y || y || N || n) asks again. if (Y || y) returns true. if (N || n) returns false.

bool getBinaryAnswer(const char\* question, const char\* error);

//same as OPI1, but using dialogue scenario

void solveOPI5();

#endif

“solution.cpp”

#include "solution.h"

#include <iostream>

#include "matrix.h"

bool getBinaryAnswer(const char\* question, const char\* error) {

char choice;

std::cout << question;

std::cin >> choice;

bool ret;

while ((choice != 'Y' && choice != 'y' && choice != 'N' && choice != 'n') || (std::cin.peek() != '\n')) {//until correct input

while (std::cin.get() != '\n');//clear istream

std::cout << error << std::endl;

std::cout << question;

std::cin >> choice;

}

if (choice == 'N' || choice == 'n')

ret = false;

else//Y or y

ret = true;

return ret;

}

void solveOPI5() {

std::cout << "You opened an app that showswork of Q&A dialogue." << std::endl;

std::cout << "This application lets user create/generate matrix of any size he wants, change values of elements in the matrix" << std::endl;

std::cout << "And executes following actions: checks every column for elements with duplicate digits." << std::endl;

std::cout << "If it has any - sorts the column, else flips it." << std::endl;

bool exit\_ = !getBinaryAnswer("Would you like to continue? (Y/N)", "wrong input, try again.");

int rowsAmount, columnsAmount;

double\*\* matrix2d = NULL;

if (!exit\_) {

rowsAmount = inputNum(1, 32767, "Enter amount of rows: ");

columnsAmount = inputNum(1, 32767, "Enter amount of columns: ");

matrix2d = createMatrix<double>(rowsAmount, columnsAmount);

if (getBinaryAnswer("Would you like to input matrix yourself? (Y/N)", "wrong input, try again.")) {

setMatrix0(matrix2d, rowsAmount, columnsAmount);

setMatrix(matrix2d, rowsAmount, columnsAmount);

}

else if (!(exit\_ = !getBinaryAnswer("Would you like to generate matrix? (Y/N)", "wrong input, try again."))) {

genMatrix(matrix2d, rowsAmount, columnsAmount);

printMatrix(matrix2d, rowsAmount, columnsAmount);

}

}

if (!exit\_)

exit\_ = !getBinaryAnswer("Would like to calculate the result? (Y/N)", "wrong input, try again.");

if (!exit\_) {

exercise(matrix2d, rowsAmount, columnsAmount);

if (exit\_ = getBinaryAnswer("Would like to print the result in console? (Y/N)", "wrong input, try again."))

printMatrix(matrix2d, rowsAmount, columnsAmount);

}

}

**5. Висновки щодо розробки діалогу типу «питання-відповідь».**

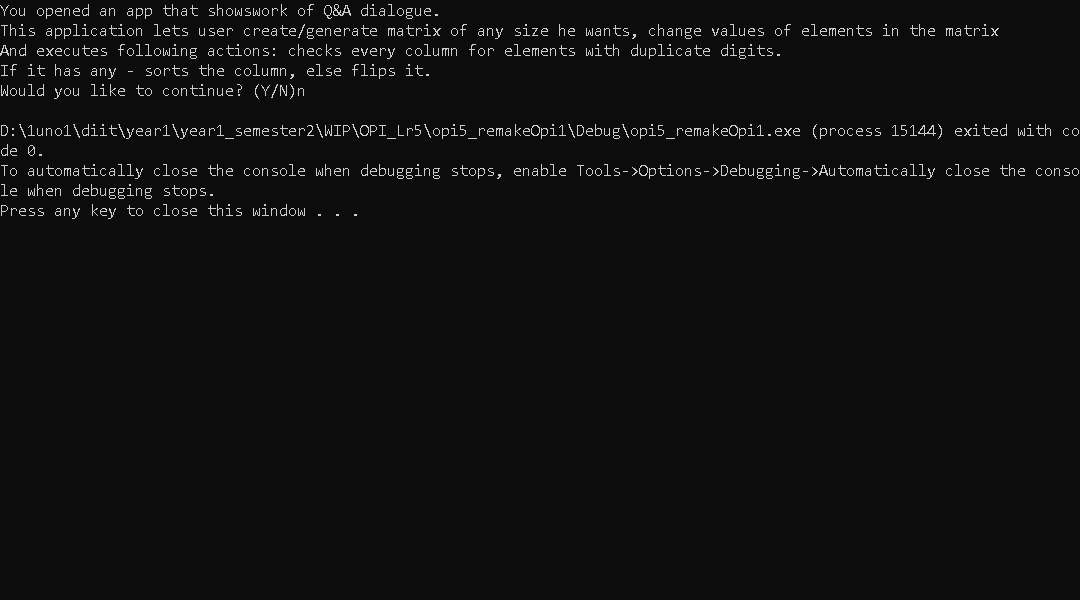


Рисунок 2(Тест 1)

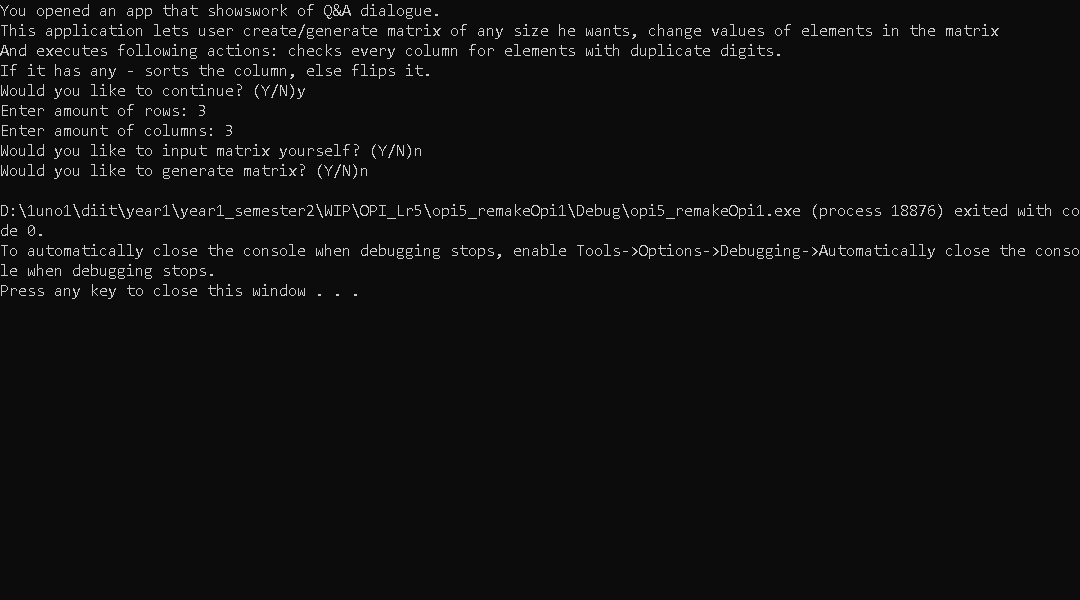


Рисунок 3(Тест 2)

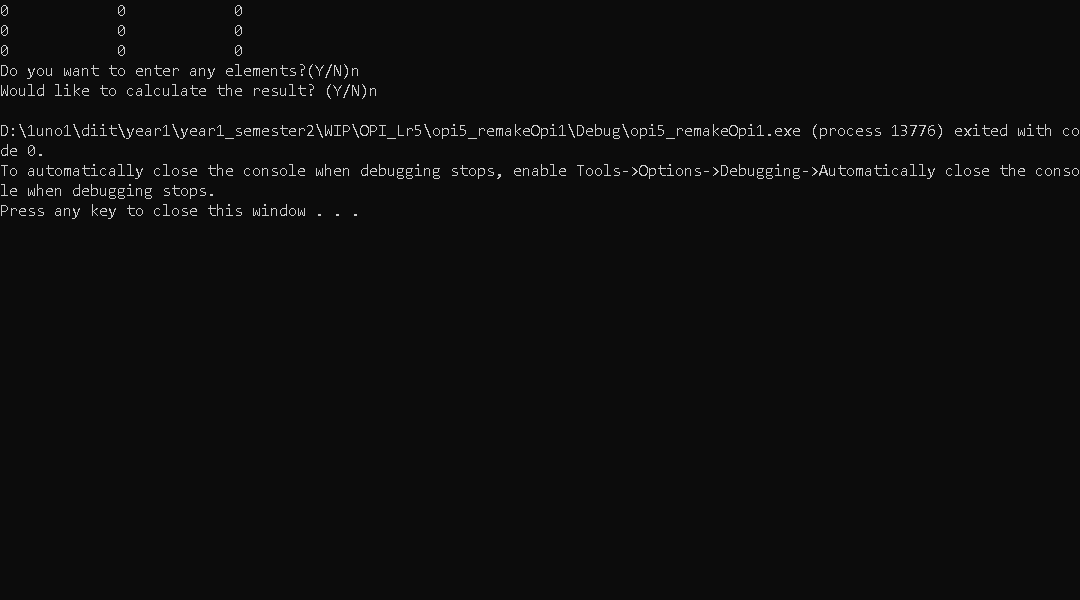


Рисунок 4(Тест 3.1)

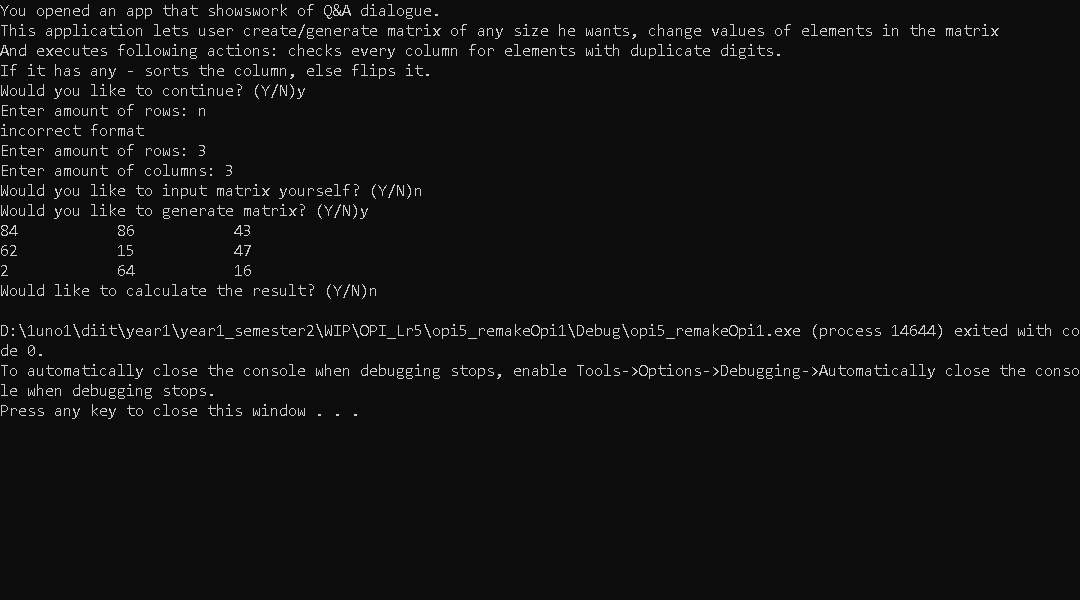


Рисунок 5(Тест 3.2)

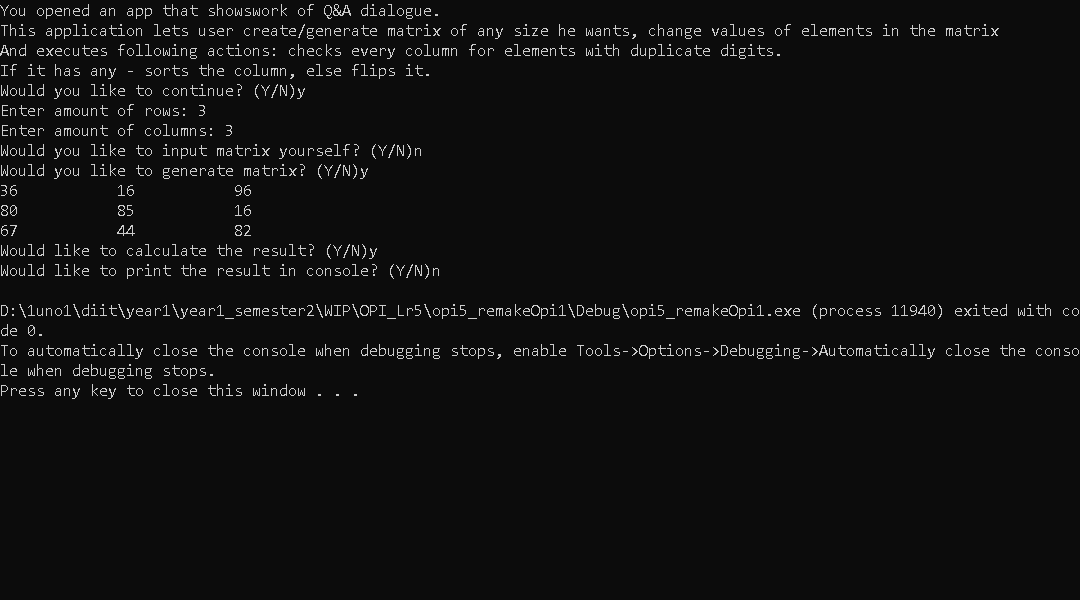


Рисунок 6(Тест 4.1)

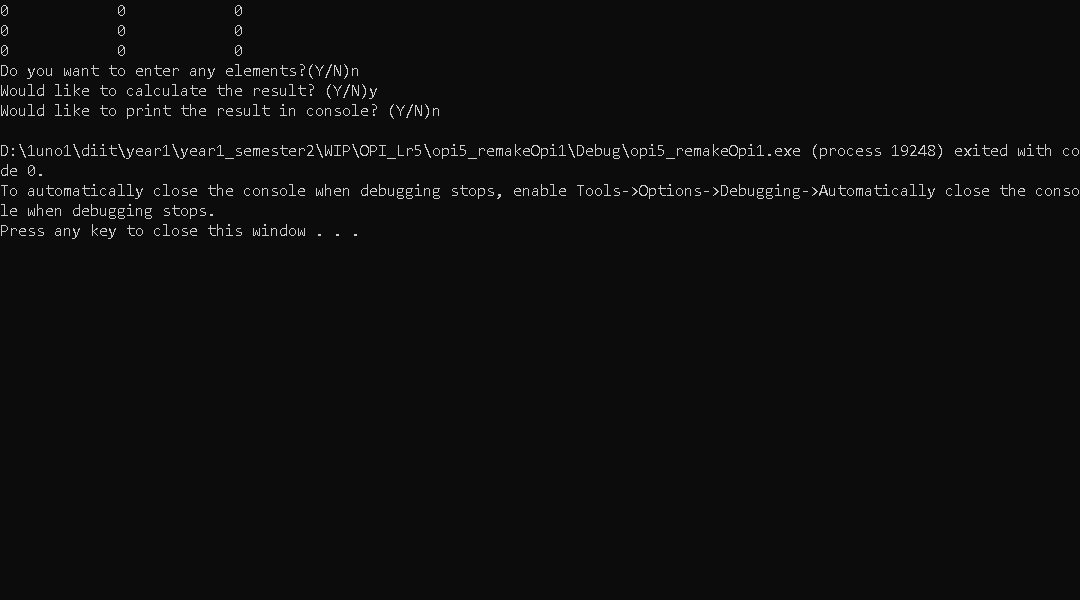


Рисунок 7(Тест 4.2)



Рисунок 8(Тест 5.1)

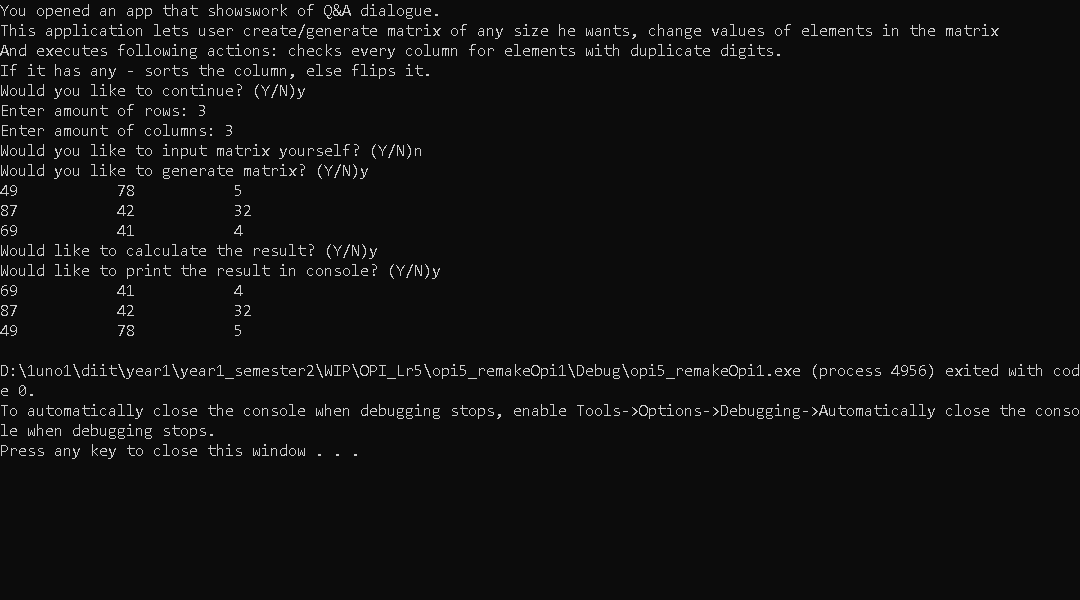


Рисунок 9(Тест 5.2)

Цей спосіб розробки достатньо точно дає зрозуміти де саме потрібно розбити код на окремі функції. Але треба бути обережним з кількістю розгалужень бо користувачу може бути незручно проходити найдовшим шляхом до отримання результату, але у той самий час потрібно забезпечити повний обхід взаємно можливих розгалужень та миттєве зупинення програми.