**Лабораторна робота №9.** **Виключення**

Тема: Виключення.

Мета: Навчитись розробляти програми з реалізацією виключень

**ВИМОГИ**

**1.1 Інформація про розробника:**

* Кліщов Б. Р.
* КІТ 102.8а
  1. **Загальне завдання**

У файлі розміщена інформація про N масивів. В першому рядку міститься інформація про кількість масивів, у кожній наступній – інформація про кількість елементів в кожному масиві та власне дані масиву. Необхідно реалізувати програму, що виконує перераховані нижче дії, причому кожна з них, в окремій функції, поки користувач не введе замість назви файлу рядок \exit Дії, що має виконувати програма такі:

- введення з клавіатури назви вхідного файлу з даними;

- читання даних з файлу;

- виконання індивідуального завдання;

- введення з клавіатури імені вихідного файлу;

- запис результату операції у файл;

- доступ до елемента за індексом слід винести в окрему функцію, що виконує перевірку на можливість виходу за межі масиву.

Слід окремо звернути увагу, що при обробці виключення цикл не повинен перериватись.

**1.3 Додаткові умови виконання завдання:**

- продемонструвати відсутність витоків пам’яті;

- продемонструвати роботу розроблених методів за допомогою модульних тестів (проемулювати роботу користувача с декількома файлами, командою \exit);

- не використовувати конструкцію «using namespace std;», замість цього слід роботи «using» кожного необхідного класу:using std::string, using std::cout

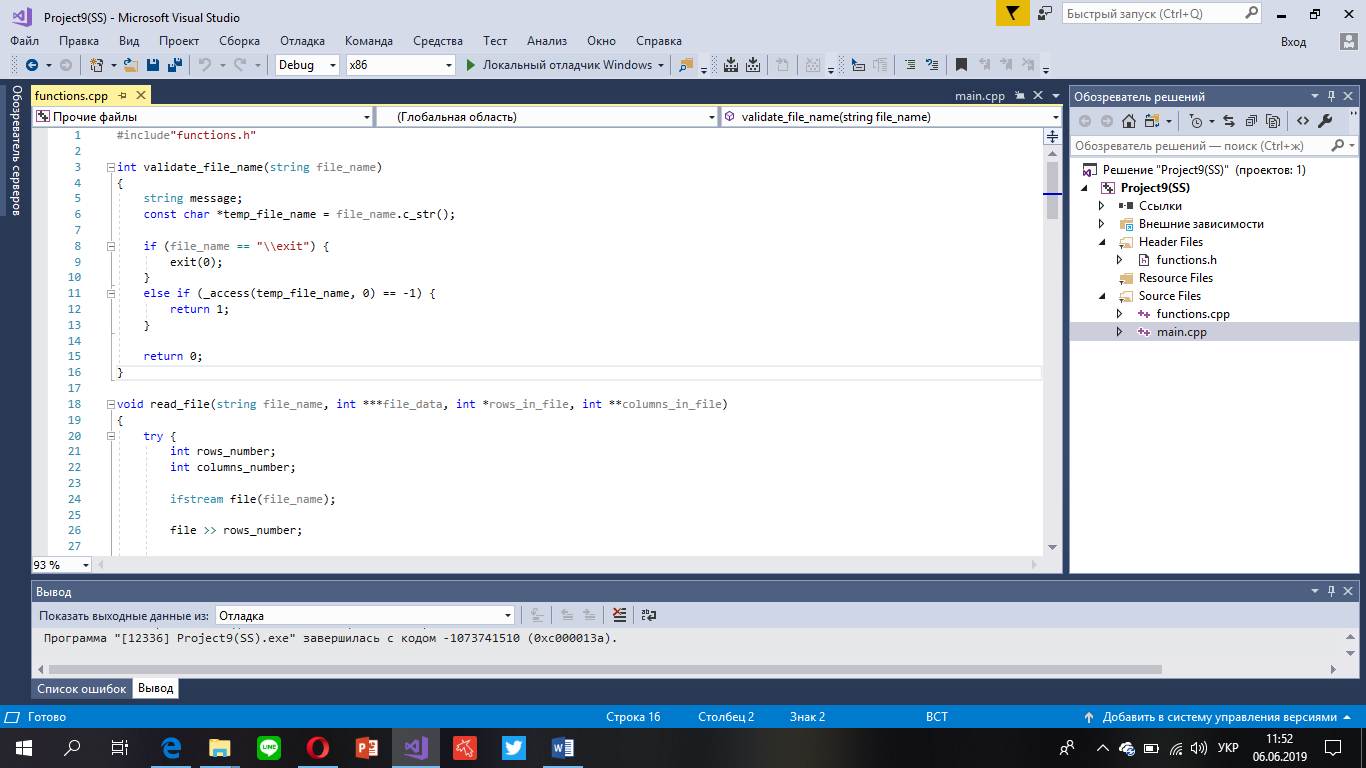
1. **ОПИС ПРОГРАМИ**

**2.1 Функціональне призначення**

Програма призначена щоб считувати двовимірний масив з файлу та вазначати суму двовомірних масивів.

* 1. **Опис логічної структури**

На рисунку № 2 зоображена структура програми:



Малюнок №2: Cтруктура програми

**2.3 Важливі фрагменти програми**

Код програми:

Main():

#include"functions.h"

int main()

{

int \*\*file\_data = nullptr;

int \*\*file\_data1 = nullptr;

int \*\*addition\_result = nullptr;

int rows\_in\_file;

int \*columns\_in\_file = nullptr;

int rows\_in\_file1;

int \*columns\_in\_file1 = nullptr;

int \*\*result;

int exception;

string message;

string file\_name;

do {

try

{

cout << "Input name of the first file: ";

cin >> file\_name;

exception = validate\_file\_name(file\_name);

if (exception == 1)

throw message = "Invalid name";

read\_file(file\_name, &file\_data, &rows\_in\_file, &columns\_in\_file);

cout << "Input name of the second file: ";

cin >> file\_name;

exception = validate\_file\_name(file\_name);

if (exception == 1)

throw message = "Invalid name";

read\_file(file\_name, &file\_data1, &rows\_in\_file1, &columns\_in\_file1);

cout << "The result of addition is:" << endl;

add\_arrays(&file\_data, &file\_data1, &result, &rows\_in\_file, &columns\_in\_file, &rows\_in\_file1, &columns\_in\_file1);

cout << "Input name of the file to write to: ";

cin >> file\_name;

exception = validate\_file\_name(file\_name);

if (exception == 1)

throw message = "Invalid name";

write\_file(file\_name, &result, &rows\_in\_file, &columns\_in\_file, &rows\_in\_file1, &columns\_in\_file1);

}

catch (...)

{

cout << message << endl;

}

} while (true);

return 0;

}

Functions():

#include"functions.h"

int validate\_file\_name(string file\_name)

{

string message;

const char \*temp\_file\_name = file\_name.c\_str();

if (file\_name == "\\exit") {

\_CrtSetReportMode(\_CRT\_WARN, \_CRTDBG\_MODE\_FILE);

\_CrtSetReportFile(\_CRT\_WARN, \_CRTDBG\_FILE\_STDERR);

\_CrtSetReportMode(\_CRT\_ERROR, \_CRTDBG\_MODE\_FILE);

\_CrtSetReportFile(\_CRT\_ERROR, \_CRTDBG\_FILE\_STDERR);

\_CrtSetReportMode(\_CRT\_ASSERT, \_CRTDBG\_MODE\_FILE);

\_CrtSetReportFile(\_CRT\_ASSERT, \_CRTDBG\_FILE\_STDERR);

\_CrtDumpMemoryLeaks();

exit(0);

}

else if (\_access(temp\_file\_name, 0) == -1) {

return 1;

}

return 0;

}

void read\_file(string file\_name, int \*\*\*file\_data, int \*rows\_in\_file, int \*\*columns\_in\_file)

{

try {

int rows\_number;

int columns\_number;

ifstream file(file\_name);

file >> rows\_number;

\*rows\_in\_file = rows\_number;

\*columns\_in\_file = new int[rows\_number]();

\*file\_data = new int \*[rows\_number]();

for (int i = 0; i < rows\_number; i++)

{

file >> columns\_number;

(\*columns\_in\_file)[i] = columns\_number;

(\*file\_data)[i] = new int[columns\_number]();

for (int j = 0; j < columns\_number; j++)

{

file >> (\*file\_data)[i][j];

cout << (\*file\_data)[i][j] << "\t";

}

cout << endl;

}

file.close();

}

catch (...) {

cout << "Oh shit, here we go again..." << endl;

}

}

void add\_arrays(int \*\*\*file\_data, int \*\*\*file\_data1, int \*\*\*result, int \*rows\_in\_file, int \*\*columns\_in\_file, int \*rows\_in\_file1, int \*\*columns\_in\_file1)

{

int biggest\_row;

int biggest\_column;

if (\*rows\_in\_file >= \*rows\_in\_file1)

{

biggest\_row = \*rows\_in\_file;

}

else

{

biggest\_row = \*rows\_in\_file1;

}

\*result = new int \*[biggest\_row]();

for (int i = 0; i < biggest\_row; i++)

{

if ((\*columns\_in\_file)[i] >= (\*columns\_in\_file1)[i])

{

(\*result)[i] = new int[(\*columns\_in\_file)[i]]();

}

else

{

(\*result)[i] = new int[(\*columns\_in\_file1)[i]]();

}

}

for (int i = 0; i < \*rows\_in\_file; i++)

{

for (int j = 0; j < (\*columns\_in\_file)[i]; j++)

{

(\*result)[i][j] = (\*file\_data)[i][j];

}

}

for (int i = 0; i < \*rows\_in\_file1; i++)

{

for (int j = 0; j < (\*columns\_in\_file1)[i]; j++)

{

(\*result)[i][j] += (\*file\_data1)[i][j];

}

}

for (int i = 0; i < biggest\_row; i++)

{

if ((\*columns\_in\_file)[i] >= (\*columns\_in\_file1)[i])

{

biggest\_column = (\*columns\_in\_file)[i];

}

else

{

biggest\_column = (\*columns\_in\_file1)[i];

}

for (int j = 0; j < biggest\_column; j++)

{

cout << (\*result)[i][j] << "\t";

}

cout << endl;

}

}

void write\_file(string file\_name, int \*\*\*result, int \*rows\_in\_file, int \*\*columns\_in\_file, int \*rows\_in\_file1, int \*\*columns\_in\_file1)

{

int biggest\_row;

int biggest\_column;

ofstream file(file\_name);

if (\*rows\_in\_file >= \*rows\_in\_file1)

{

biggest\_row = \*rows\_in\_file;

}

else

{

biggest\_row = \*rows\_in\_file1;

}

for (int i = 0; i < biggest\_row; i++)

{

if ((\*columns\_in\_file)[i] >= (\*columns\_in\_file1)[i])

{

biggest\_column = (\*columns\_in\_file)[i];

}

else

{

biggest\_column = (\*columns\_in\_file1)[i];

}

for (int j = 0; j < biggest\_column; j++)

{

file << (\*result)[i][j] << "\t";

}

file << endl;

}

file.close();

}

1. **ВАРІАНТИ ВИКОРИСТАННЯ**

**3.1 Результат роботи функцій**

На рисунку № 3 зоображено результат роботи програми

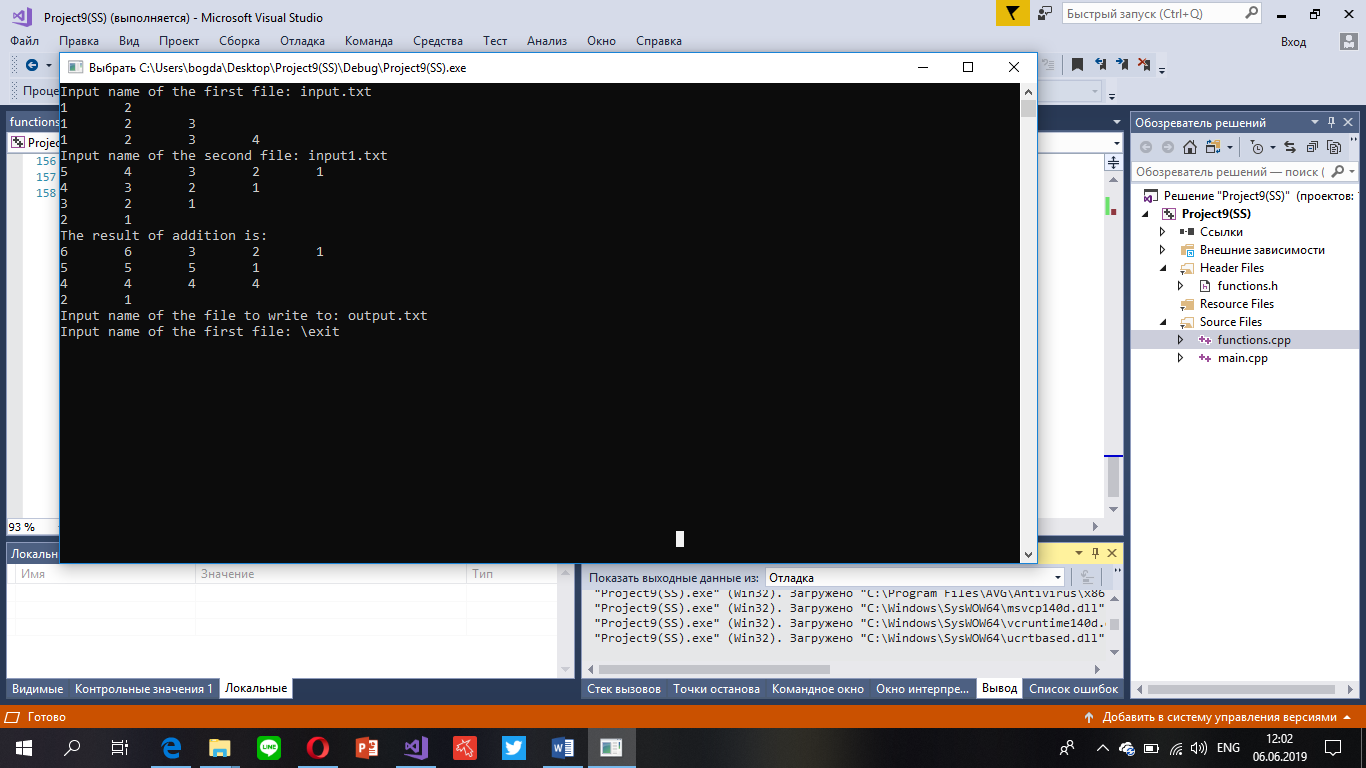


Рисунок № 3. Результат роботи програми

**Висновок:** Навчився розробляти програми з реалізацією виключень.