**Постановка задачи**

Разработать детальные требования, тест план и программу для следующей задачи:

* Напишите функцию (в соответствии с указанным вариантом) для работы с **двумерным массивом**– квадратной матрицей в динамической памяти. Функция, должна иметь два параметра – адрес  
  матрицы (целые числа), размерность матрицы и выполнять действия в соответствии с указанным  
  вариантом.
* Элементы матрицы – целые числа.
* Ввод и вывод матрицы выполнять в отдельных функциях.
* **Входные данные:** размерность матрицы, значения элементов (если требуется) **ввести из файла**.
* **Выходные данные**: в соответствии заданным с вариантом **вывести в файл**. Количество столбцов, элементы которых упорядочены по возрастанию элементов
* Найти минимум среди сумм элементов диагоналей параллельных главной диагонали матрицы.

1. **Детальные требования**
   1. Вводимый файл должны быть.
      1. Вводимые данные – целые числа.
         1. Если файла нет, то вывести сообщение «The file was not open!», завершив программу
         2. Если введенные данные не являются целым числом, или вводимые данные отсутствуют, то сообщение «Either a non-integer number was entered, or a non-number was entered, end of the program!»
         3. Если введенные данные являются целыми числами, то выводим в файл сообщение «The minimum among the sums of the elements of the diagonals parallel to the main diagonal of the matrix: результат»

**Тест-план**

|  |  |  |  |
| --- | --- | --- | --- |
| Требование | Детальные требования | Данные | Ожидаемый результат |
| 1.1. Вводимые данные должны быть. | | | |
| 1.1.1 Вводимые данные – целые числа | 1.1.1.1 Если файла нет, то вывести сообщение «The file was not open!», завершив программу | Отсутствует файл | Сообщение «The file was not open!», |
| 1.1.1 Вводимые данные – целые числа | * + - 1. Если введенные данные не являются целым числом, или вводимые данные отсутствуют, то сообщение «Either a non-integer number was entered, or a non-number was entered, end of the program!» | A  S  Sdf  dss | Cообщение «Either a non-integer number was entered, or a non-number was entered, end of the program!» |
| 1.1.1 Вводимые данные – целые числа | * + - 1. Если введенные данные являются целыми числами, то выводим в файл сообщение «The minimum among the sums of the elements of the diagonals parallel to the main diagonal of the matrix: результат» | 3  3  1 2 3  1 1 1  2 3 1 | Сообщение «The minimum among the sums of the elements of the diagonals parallel to the main diagonal of the matrix: 2» |

**Код:**

#include <iostream>

#include <fstream>

#include <stdlib.h>

const char\* ERROR\_FILE\_NOT\_OPEN = "The file was not open!\n";

const char\* ERROR\_INVALID\_INPUT = "Either a non-integer number was entered, or a non-number was entered, end of the program!\n";

int FindMinSum(int\*\* Mass, int line, int columns);

int\*\* inputArray(std::ifstream& input, int\*\* Mass, int line, int columns);

int main()

{

std::ifstream input("C:/Users/PetrK/Documents/Visual Studio Files/ConsoleApplication16/input.txt");

std::ofstream output("C:/Users/PetrK/Documents/Visual Studio Files/ConsoleApplication16/output.txt");

int line = 0, columns = 0;

try

{

if (!input.is\_open())

{

throw ERROR\_FILE\_NOT\_OPEN;

}

input >> line >> columns;

if ((!input) or input.fail())

{

throw ERROR\_INVALID\_INPUT;

}

}

catch (const char\* error)

{

std::cerr << "\n" << error ;

return -1;

}

int \*\* array = new int\* [line];

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

{

array[i] = new int[columns];

}

output << "The minimum among the sums of the elements of the diagonals parallel to the main diagonal of the matrix: " << FindMinSum(inputArray(input, array, line, columns), line, columns);

input.close();

output.close();

return 0;

}

int \*\* inputArray(std::ifstream& input, int\*\* Mass, int line, int columns)

{

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

{

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

{

input >> Mass[i][j];

}

}

return Mass;

}

int FindMinSum(int\*\* Mass, int line, int columns)

{

int k = 0, sum = 0, k1 = 1, SumMin1 = INT\_MAX, SumMin2 = INT\_MAX;

for (int k = 1; k < columns; k++)

{

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

{

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

{

if ((i + k == j) and (i + k < columns))

{

if (k == k1)

{

sum = sum + Mass[i][j];

}

else

{

SumMin1 = std::min(sum, SumMin1);

sum = Mass[i][j];

k1 = k;

}

if (k == columns - 1)

{

SumMin1 = std::min(sum, SumMin1);

sum = Mass[i][j];

SumMin1 = std::min(sum, SumMin1);

k1 = k;

}

}

}

}

}

int h1 = 1;

sum = 0;

for (int h = 1; h < line; h++)

{

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

{

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

{

if ((j + h == i) and (j + h < line))

{

if (h == h1)

{

sum = sum + Mass[i][j];

}

else

{

SumMin2 = std::min(sum, SumMin2);

sum = Mass[i][j];

h1 = h;

}

if (h == line - 1)

{

SumMin2 = std::min(sum, SumMin2);

sum = Mass[i][j];

SumMin2 = std::min(sum, SumMin2);

h1 = h;

}

}

}

}

}

return std::min(SumMin1, SumMin2);

}