Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 8 з дисципліни
«Основи програмування-1.
Базові конструкції»

«Багатовимірні масиви»

Варіант 26

Виконав студент ІП-11 Рябов Юрій Ігорович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вітковська Ірина Іванівна
(прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота№8

Багатовимірні масиви

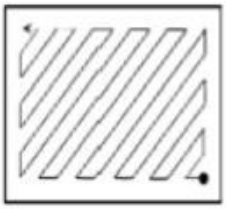
Мета

Опанувати технологію використання двовимірних масивів даних(матриць), навчитися розробляти алгоритми та програми із застосуванням матриць.

Індивідуальне завдання

Варіант 26

26. Побудувати квадратну матрицю з елементами 1, 2, ..., n, розміщеними наступним чином, починаючи з правого нижнього кута. Знайти максимальне значення кутових елементів отриманої матриці.



Постановка задачі

Необхідно за допомогою підпрограм руху по діагоналі вліво вниз та вправо вгору заповнити матрицю числами від n до 1 за заданою схемою. В залежності від того до якої “стінки” матриці дійшов алгоритм наступним буде елемент відповідної стінки і рух буде виконуватись по заданій діагоналі.

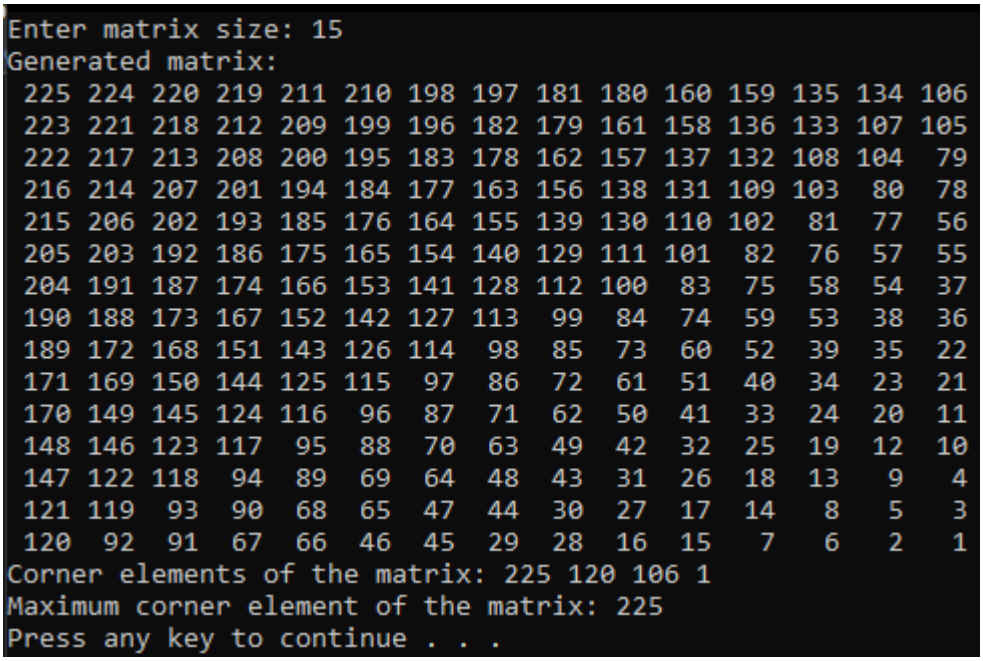
Програма на мові C++:

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;

int** CreateMatrix(int rowNumber, int columnNumber);
void FillByPattern(int** matrix, int matrixSize);
void MoveDiagonallyUp(int** matrix, int& currentNumber, int& currentRow, int& currentColumn, int size);
void MoveDiagonallyDown(int** matrix, int& currentNumber, int& currentRow, int& currentColumn, int size);
void OutputMatrix(int** matrix, int matrixSize);
void OutputCorners(int** matrix, int matrixSize);
void DeleteMatrix(int** matrix, int rowNumber);

int main()
{
	int matrixSize;
	int** matrix;

	cout << "Enter matrix size: ";
	cin >> matrixSize;

	matrix = CreateMatrix(matrixSize, matrixSize);
	FillByPattern(matrix, matrixSize);
	cout << "Generated matrix:\n";
	OutputMatrix(matrix, matrixSize);
```

```
	cout << "Corner elements of the matrix: ";
	OutputCorners(matrix, matrixSize);
	cout << "\nMaximum corner element of the matrix: " << matrix[0][0] << "\n";
	DeleteMatrix(matrix, matrixSize);


	system("pause");
}

int** CreateMatrix(int rowNumber, int columnNumber)
{
	int** matrix = new int* [rowNumber];
	for (int i = 0; i < rowNumber; i++)
	{
		matrix[i] = new int[columnNumber];
	}
	return matrix;
}

void FillByPattern(int** matrix, int matrixSize)
{
	matrix[0][0] = (int)pow(matrixSize, 2);
	int currentNumber = matrix[0][0] - 1;
	int currentRow = 0;
	int currentColumn = 0;


	while (currentNumber > 0)
	{
		if (currentRow == 0 && currentColumn != matrixSize - 1)
		{
			currentColumn += 1;
			MoveDiagonallyDown(matrix, currentNumber, currentRow, currentColumn, matrixSize);
		}
		else if (currentColumn == 0 && currentRow != matrixSize - 1)
		{
			currentRow += 1;
			MoveDiagonallyUp(matrix, currentNumber, currentRow, currentColumn, matrixSize);
		}
		else if (currentColumn == matrixSize - 1)
		{
			currentRow += 1;
			MoveDiagonallyDown(matrix, currentNumber, currentRow, currentColumn, matrixSize);
		}
		else if (currentRow == matrixSize - 1)
		{
			currentColumn += 1;
			if (currentNumber != 1)
			{
				MoveDiagonallyUp(matrix, currentNumber, currentRow, currentColumn, matrixSize);
			}
			else
			{
				matrix[currentRow][currentColumn] = 1;
				currentNumber -= 1;
			}
		}
	}
}

// Function which fills a diagonal with decreasing values from bottom to top moving to the right
void MoveDiagonallyUp(int** matrix, int& currentNumber, int& currentRow, int& currentColumn, int size)
{
	for (currentRow, currentColumn; currentRow >= 0 && currentColumn <= size - 1; currentRow--, currentColumn++)
	{
		matrix[currentRow][currentColumn] = currentNumber;
		currentNumber -= 1;
	}
	currentRow++;
```

```
		currentColumn--;
}

// Function which fills a diagonal with decreasing values from top to bottom moving to the left
void MoveDiagonallyDown(int** matrix, int& currentNumber, int& currentRow, int& currentColumn, int size)
{
		for (currentRow, currentColumn; currentRow <= size - 1 && currentColumn >= 0; currentRow++, currentColumn--)
		{
			matrix[currentRow][currentColumn] = currentNumber;
			currentNumber -= 1;
		}
		currentRow--;
		currentColumn++;
}

void OutputMatrix(int** matrix, int matrixSize)
{
		for (int i = 0; i < matrixSize; i++)
		{
			for (int j = 0; j < matrixSize; j++)
			{
				cout << setw((int)log10(pow(matrixSize, 2)) + 2) << matrix[i][j];
			}
			cout << "\n";
		}
}

void OutputCorners(int** matrix, int matrixSize)
{
		cout << matrix[0][0] << " " << matrix[matrixSize - 1][0] << " " << matrix[0][matrixSize - 1] << " " <<
matrix[matrixSize - 1][matrixSize - 1];
}

void DeleteMatrix(int** matrix, int rowNumber)
{
		for (int i = 0; i < rowNumber; i++)
		{
			delete(matrix[i]);
		}
		delete(matrix);
}
```

Виконання коду на мові C++:



Висновок

Отже, ми опанували технологію використання двовимірних масивів даних та навчились застосовувати їх при розробці програм, запрограмувавши заповнення матриці елементами за заданою схемою.