

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)

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

Київ 2021

Лабораторна робота №8

Багатовимірні масиви

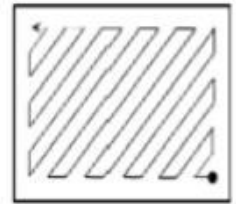
Мета

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 $1, 2, \dots, n$, 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++:

```
Enter matrix size: 15
Generated matrix:
 225 224 220 219 211 210 198 197 181 180 160 159 135 134 106
 223 221 218 212 209 199 196 182 179 161 158 136 133 107 105
 222 217 213 208 200 195 183 178 162 157 137 132 108 104 79
 216 214 207 201 194 184 177 163 156 138 131 109 103 80 78
 215 206 202 193 185 176 164 155 139 130 110 102 81 77 56
 205 203 192 186 175 165 154 140 129 111 101 82 76 57 55
 204 191 187 174 166 153 141 128 112 100 83 75 58 54 37
 190 188 173 167 152 142 127 113 99 84 74 59 53 38 36
 189 172 168 151 143 126 114 98 85 73 60 52 39 35 22
 171 169 150 144 125 115 97 86 72 61 51 40 34 23 21
 170 149 145 124 116 96 87 71 62 50 41 33 24 20 11
 148 146 123 117 95 88 70 63 49 42 32 25 19 12 10
 147 122 118 94 89 69 64 48 43 31 26 18 13 9 4
 121 119 93 90 68 65 47 44 30 27 17 14 8 5 3
 120 92 91 67 66 46 45 29 28 16 15 7 6 2 1
Corner elements of the matrix: 225 120 106 1
Maximum corner element of the matrix: 225
Press any key to continue . . .
```

Висновок

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