Ministry of Education of Republic of Moldova

Technical University of Moldova

Faculty of Computers, Informatics and Microelectronics

Software Engineering Department

Computer Programming

Laboratory work #3

Two-Dimensional Array Operations and Processing

Author:
Mihai Caraman
std. gr. FAF-233

Verified:
Alexandru Furdui

Chișinău 2023

Theory Background

A two-dimensional array is a data structure that organizes data in a grid-like format, consisting of rows and columns. Each element in a 2D array is identified by two indices: a row index and a column index. Two-dimensional arrays are fundamental in programming and are used for various applications, including representing images, matrices, grids, game boards, and more. Understanding their operations and processing is crucial for efficient manipulation and utilization of structured data.

The Task

Describe your task, and enumerate the task/tasks you have implemented:

1. Task: Hard - Constructing a Mountain Matrix(worth 10 points)

Technical implementation

Pseudocode

```
Define function InsertionSort(array, length)
    For i = 1 to length - 1
        Set key = array[i]
        Set j = i - 1
        While j >= 0 and array[j] > key
            Move array[j] to array[j + 1]
            Set j = j - 1
        End While
        Set array[j + 1] = key
    End For
End Function

Define function printSpiral(rows, cols, matrix, array)
    Set top = 0, bottom = rows - 1, left = 0, right = cols -
       1
    Set index = 0
    While top <= bottom and left <= right
        For i = left to right
            Set matrix[top][i] = array[index++]
        End For
        Increment top
```

```
        For i = top to bottom
            Set matrix[i][right] = array[index++]
        End For
        Decrement right

        If top <= bottom
            For i = right to left
                Set matrix[bottom][i] = array[index++]
            End For
            Decrement bottom
        End If

        If left <= right
            For i = bottom to top
                Set matrix[i][left] = array[index++]
            End For
            Increment left
        End If
    End While
End Function

Define function main()
    Read rows from user
    Read cols from user
    Create 2D matrix with dimensions rows and cols
    For i = 0 to rows - 1
        For j = 0 to cols - 1
            Read matrix[i][j] from user
        End For
    End For
    Convert 2D matrix to 1D array
    Call InsertionSort with array and length
    Print sorted 2D array in a matrix-like format
    Call printSpiral with rows, cols, matrix, and array
    Print matrix in a clockwise spiral pattern
End Function
```

Results

The program takes user input for the number of rows and columns, followed by input for the elements of a matrix. It then performs the following operations:

Insertion Sort:

- Converts the 2D matrix into a 1D array.
- Sorts the 1D array using the Insertion Sort algorithm.

Print Sorted 2D Array as a Matrix:

- Prints the sorted 1D array as a matrix, using the specified number of columns.

Print Matrix in a Clockwise Spiral Pattern:

- Calls a function (printSpiral) to fill and print the matrix in a clockwise spiral pattern.

User Input:

- User provides the number of rows (e.g., 3) and columns (e.g., 4).
- User inputs elements for a 3x4 matrix.

Insertion Sort and 1D Array Conversion:

- The program converts the 2D matrix to a 1D array, which is then sorted using Insertion Sort.

Print Sorted 2D Array as a Matrix:

- The program prints the sorted 1D array in a matrix-like format using the specified number of columns.

Print Matrix in a Clockwise Spiral Pattern:

- The program fills and prints the original matrix in a clockwise spiral pattern using the sorted array.

Output:

- The sorted 1D array is printed in a matrix-like format.
- The original matrix is printed in a clockwise spiral pattern.

The actual output, such as the sorted 1D array and the matrix in a clockwise spiral pattern, will depend on the specific input provided by the user. The program demonstrates the sorting of a 1D array, conversion of a 2D matrix to a 1D array, and a visualization of the original matrix in a spiral pattern.

Conclusion

Functionality: Code performs sorting using Insertion Sort on a 1D array obtained from a 2D matrix. It visualizes the sorted array as a matrix and displays the original matrix in a clockwise spiral pattern.

Input Processing: Accepts user input for matrix dimensions and elements, demonstrating user interaction.

Modular Design: Well-structured code with separate functions for sorting and matrix traversal, promoting code reuse and readability.

Visualization: Effectively displays the sorted array in a matrix-like format and visualizes the original matrix in a clockwise spiral pattern.

Optimization Potential: Room for optimization, especially in memory usage and code readability.

User Interaction and Feedback: Provides clear prompts for user input, enhancing user understanding and interaction.

Bibliography

1. https://stackoverflow.com/
2. https://www.geeksforgeeks.org/
3. https://github.com/caramisca/Lab-3 (github link with all docs, source code, pdf file, .tex file)

```
Enter the number of rows: 11
Enter the number of columns: 10
Enter the elements of the matrix:
187 153 175 159 183 146 103 205 199 139 1
105 127 140 172 192 170 114 209 179 151 2
150 207 148 174 117 195 169 141 100 163 1

Sorted 2D array using Insertion Sort (as
100 101 102 103 104 105 106 107 108 109
110 111 112 113 114 115 116 117 118 119
120 121 122 123 124 126 127 128 129 130
131 132 133 134 135 136 138 139 140 141
142 143 144 145 146 147 148 149 150 151
152 153 154 155 156 157 158 159 160 161
162 163 164 165 166 167 168 169 170 171
172 173 174 175 176 177 178 179 180 181
182 183 184 185 186 187 188 189 190 191
192 193 194 195 196 197 198 199 200 201
202 203 204 205 206 207 208 209 210 211

Matrix in a clockwise spiral pattern:
100 101 102 103 104 105 106 107 108 109
139 140 141 142 143 144 145 146 147 110
138 169 170 171 172 173 174 175 148 111
136 168 191 192 193 194 195 176 149 112
135 167 190 205 206 207 196 177 150 113
134 166 189 204 211 208 197 178 151 114
133 165 188 203 210 209 198 179 152 115
132 164 187 202 201 200 199 180 153 116
131 163 186 185 184 183 182 181 154 117
130 162 161 160 159 158 157 156 155 118
129 128 127 126 124 123 122 121 120 119
```

Figure 1: The results from the console