

Exercise 3: Template

Double-subscripted arrays are considered as two-dimensional arrays.

To identify a particular element in the array, we need to specify two subscripts:

- The first subscript identifies the element's row
- The second subscript identifies the element's column.

Example:

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Diagram illustrating the structure of a double-subscripted array. The array is represented as a grid of elements. The first subscript (row index) identifies the row, and the second subscript (column index) identifies the column. The array name 'a' is shown in the first column of each row. Arrows point to the row index, column index, and array name labels.

Double-subscripted array with three rows and four columns.

In the exercise:

- The representation of the **double-subscripted** array is a **single-subscripted** array
- The size of the array is (rows*columns) elements
- By default, the double-subscripted array has (10*10) elements

Class DoubleSubscriptedArray

1- Private data members:

```
int rowSize;           // number of rows in the array
int columnSize;        // number of columns in the array
int *ptr;               // pointer to the first element in the array
```

2- Overloaded operators:

```
operator(int, int)      // access element at given row and column, this function has two versions*
operator=               // assign one array to another one
operator==              // check if two arrays are equal
operator!=              // check if two arrays are different
```

<code>operator<<</code>	<code>// input the content of the array</code>
<code>operator<<</code>	<code>// output the array in row and column format</code>

*: There are two versions of the double subscript **operator()** depending on the return:

1. int& operator()(int, int)

Function call for **non-const** objects, it returns **lvalue** (reference)

2. const int& operator()(int, int)

Function call for **const** objects, it returns **rvalue** (value)