Note 03: Generic Data Structure

Instruction

We are going to create one of the most elementary data structures, namely, the array. We are provided the header file '`Object.h`' that contains the interface named *Object* that consists of the method:

- `toString()` - displays the array.

Array

Try to create the array data structure in the following stages.

Stage 1: Create a header file named '`Array.h`' that contains a header guard and includes the libraries *iostream*, *string*, *sstream*, *stdexcept*, and '`Object.h`'.

Stage 2: Within the namespace *dsn*, create an empty generic class named *Array* that publicly inherits *Object*.

Stage 3: Within the class *Array*, declare a private generic pointer field name *data*, and a private unsigned integer field named *capacity*.

Stage 4: Within the class *Array*, define the default constructor so that it allocates *data* to an array of size 10, initializes each of *data* to the generic type default value, and assigns 10 to *capacity*.

Stage 5: Within the class *Array*, define an overloaded constructor that takes an unsigned integer parameter and allocates *data* to an array of size equal to the parameter, initializes each of *data* to the generic type default value, and assigns the parameter to *capacity* if the parameter is greater than 0; otherwise, it uses 10 instead of th parameter in the statements.

Stage 6: Within the class *Array*, define the copy constructor so that it performs a deep copy.

Stage 7: Within the class *Array*, define the assignment operator so that it performs a deep copy.

Stage 8: Within the class *Array*, define the destructor so that it deallocates *data*.

Stage 9: Within the class *Array*, define the unsigned integer constant method named `size()` that takes no parameters and returns *capacity*.

Stage 10: Within the class *Array*, define the unsigned integer constant method named `length()` that takes no parameters and returns *capacity*.

Stage 11: Within the class *Array*, define an overloaded constant subscript operator that takes an unsigned integer parameter and returns the element of *data* whose index matches the parameter if the parameter is a valid index; otherwise, it throws an out-of-range exception.

Stage 12: Within the class *Array*, define an overloaded subscript operator that takes an unsigned integer parameter and returns the element of *data* whose index matches the parameter if the parameter is a valid index; otherwise, it throws an out-of-range exception.

Stage 13: Within the class *Array*, override `toString()` so that it displays a list of the elements of *data* all enclosed in square braces.

Stage 14: Create a '`main.cpp`' file, include the header file '`Array.h`'.

Stage 15: In the '`main.cpp`' file, define a generic function named `Minimum()` that takes a generic *Array* reference parameter and returns the minimum element of the parameter.

Stage 16: With the main function of the '`main.cpp`' file, create two *Array* objects of different types, populate them, display them, and display the call of `Minimum()` for each object as the argument.