

## Note 04: Nodes & Arrays

### Instruction

We are going to create the second elementary data structures, namely, the linked list. It requires creating a structure called a *node*.

#### Node

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

Stage 1: Create a header file named ‘`Node.h`’ that contains a header guard and includes the libraries `iostream`, `string`, `sstream`, and `stdexcept`.

Stage 2: Within the namespace `dsn`, create an empty generic class named `Node`.

Stage 3: Within the `Node` class, declare a private generic field name `content`, and a private generic `Node` pointer array field named `links` with a size of 2.

Stage 4: Within the `Node` class, define the default constructor so that it assigns the generic default value to `content` and null to each element of `links`.

Stage 5: Within the `Node` class, define an overloaded constructor that takes a constant generic type reference parameter and assigns the parameter to `content` and null to each element of `links`.

Stage 6: Within the `Node` class, define the copy constructor so that it copies `content` field and assigns null to each element of `links`.

Stage 7: Within the `Node` class, define the assignment operator so that it copies `content` field.

Stage 8: Within the `Node` class, define an empty destructor.

Stage 9: Within the `Node` class, define constant generic type reference constant method named `data()` that takes no parameters and returns `content`.

Stage 10: Within the `Node` class, define generic type reference method named `data()` that takes no parameters and returns `content`.

Stage 11: Within the `Node` class, define generic `Node` pointer reference method named `prev()` that takes no parameters and returns the first element of `links`.

Stage 12: Within the `Node` class, define generic `Node` pointer reference method named `next()` that takes no parameters and returns the second element of `links`.

Stage 13: Within the `Node` class, define a friend generic `Node` pointer function named `clone()` that takes a generic `Node` pointer parameter and returns a copy of the list pointed to by the parameter.

Stage 14: Within the `Node` class, define a friend void function named `clear()` that takes a generic `Node` pointer reference parameter and deletes the list pointed to by the parameter.

Stage 15: Create a ‘`main.cpp`’ file, include the header file ‘`Node.h`’.

Stage 16: In the ‘`main.cpp`’ file, define a generic function named `Minimum()` that takes a generic `Node` pointer parameter and returns the minimum element of the linked list pointed to by the parameter.

Stage 17: With the main function of the ‘`main.cpp`’ file, create two linked lists of different types, populate them, display them, and display the call of `Minimum()` for each object as the argument.