

```

1  /*
2   * Class name: BSTNode.h
3   * Class description: Templatized node for the associated binary search tree class (BSTree.h). Stores a
4   * single value and contains pointers to two child nodes, where the left child is less than the contained value
5   * and the right child is greater than the contained value.
6   *
7   * Programmer: Chad Philip Johnson
8   * Date Created: Friday, April 26th, 2013
9   * Last Date Modified: Thursday, May 09th, 2013
10  *
11  * Sources Used:
12  *     N/A
13  */
14
15 using namespace std;
16
17 #ifndef BSTNODE_H
18 #define BSTNODE_H
19
20 template<typename T>
21 class BSTNode
22 {
23     public:
24         /**
25          * Default constructor for the BSTNode class. Sets the left child and right child pointers to NULL and
26          * assigns a given value to the node.
27          * @param tmpData The value to be assigned to the node.
28          */
29     BSTNode( T tmpData )
30     : tmpData( tmpData ), pobjBSTNodeLeftChild( NULL ), pobjBSTNodeRightChild( NULL )
31     {
32         /* empty */
33     }
34
35     /**
36      * Destructor for the BSTNode class. Currently unused.
37      */
38     virtual ~BSTNode()
39     {
40         /* empty */
41     }
42
43     /**
44      * Changes the value associated with the current node.
45      * @param tmpData The value to be assigned to the node.
46      */
47     void setData( T tmpData )
48     {
49         this->tmpData = tmpData;
50     }

```

```
51
52 /**
53 * Changes the associated left node (value is less than current node) with the current node as its
54 *      left child.
55 * @param pobjBSTNodeLeftChild The pointer to the left child node.
56 */
57 void setLeftChild( BSTNode* pobjBSTNodeLeftChild )
58 {
59     this->pobjBSTNodeLeftChild = pobjBSTNodeLeftChild;
60 }
61
62 /**
63 * Changes the associated right node (value is greater than current node) with the current node as its
64 *      right child.
65 * @param pobjBSTNodeRightChild The pointer to the right child node.
66 */
67 void setRightChild( BSTNode* pobjBSTNodeRightChild )
68 {
69     this->pobjBSTNodeRightChild = pobjBSTNodeRightChild;
70 }
71
72 /**
73 * Return the value of the node.
74 * @return The value of the node.
75 */
76 T getData() const
77 {
78     return this->tmpData;
79 }
80
81 /**
82 * Return the memory address of the variable storing the value of the node.
83 * @return The memory address of the variable storing the value of the node.
84 */
85 T& getData()
86 {
87     return this->tmpData;
88 }
89
90 /**
91 * Return the pointer value for the associated left child node.
92 * @return The pointer value for the left child node.
93 */
94 BSTNode<T>* getLeftChild() const
95 {
96     return this->pobjBSTNodeLeftChild;
97 }
98
99 /**
100 * Return the memory address of the pointer for the associated left child node.
```

```
101     * @return The memory address of the pointer for the associated left child node.  
102     */  
103     BSTNode<T>*& getLeftChild()  
{  
105         return this->pobjBSTNodeLeftChild;  
106     }  
108  
109     /**  
110      * Return the pointer value for the associated right child node.  
111      * @return The pointer value for the right child node.  
112      */  
113     BSTNode<T>* getRightChild() const  
{  
115         return this->pobjBSTNodeRightChild;  
116     }  
117  
118     /**  
119      * Return the memory address of the pointer for the associated right child node.  
120      * @return The memory address of the pointer for the associated right child node.  
121      */  
122     BSTNode<T>*& getRightChild()  
{  
123         return this->pobjBSTNodeRightChild;  
124     }  
125  
126     private:  
127         BSTNode<T>* pobjBSTNodeLeftChild;  
128         BSTNode<T>* pobjBSTNodeRightChild;  
129         T tmpData;  
130     };  
131 #endif  
132  
133
```