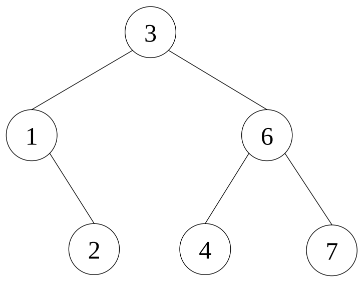
Task 1

1. Construct a parameterizable binary search tree for storing pointers to different objects. The class must include functions for inserting a new element and displaying all the elements sorted in ascending order.

A *binary tree* is a hierarchical data structure composed of a number of nodes (one of them represents the root of the tree) interlinked by edges. Each intermediate node has exactly one ancestor and up to two descendants. Different values are usually stored in the nodes of a binary tree.
In a *binary search tree* the nodes are organized so as the following properties are satisfied for each node *n*:

- n's value is greater than all values in its left subtree;
- n's value is less than all values in its right subtree;
- both the left and the right subtrees are binary search trees.

As a result, all the nodes are kept sorted according to their values. An example of a binary search tree keeping integer values is shown in the figure.



2. Test the developed class with the aid of the following *main* function:

```
int main (int argc, char* argv [])
{
	CTree<int> treeInt;
	int array[] = { 3, 1, 6, 7, 4, 2};
	unsigned size = sizeof(array) / sizeof(array[0]);
	for (unsigned i = 0; i < size; i++)
		treeInt += array+i;
	cout << treeInt;

	CTree<char> treeChar;
	char car[] = { 'a', 'x', 'g', 'm', 'b', 'q'};
	size = sizeof(car) / sizeof(car[0]);
	for (unsigned i = 0; i < size; i++)
		treeChar += car+i;
	cout << treeChar;

	CTree<CFigure> treeFig; //the class CFigure is from the lesson 11
	CCircle c1(2.3);        CCircle c2(5.3);
	CRectangle r(2,4);      CSquare s(2.5);
	treeFig += &c1;   treeFig += &c2;
	treeFig += &r;    treeFig += &s;
	cout << treeFig;

	CTree<String> treeStr; //the class String is from the lesson 8
	String str[] = {"Aveiro", "Porto", "Minsk", "Lisboa", "Coimbra"};
	size = sizeof(str) / sizeof(str[0]);
	for (i = 0; i < size; i++)
		treeStr += str+i;
	cout << treeStr;

	return 0;
}
```