Tutorial – Heaps

The heap is one of the complicated data structures to implement out of all the data structures that you will cover this year. In order for a heap to work correctly, you need to ensure that it remains in the correct order within the array, and that you are able to correctly move between different nodes in the heap correctly.

For this tutorial, you will be require to create a binary heap, using a dynamically-sized array to store the information.

Getting Started

1. Create a new C++ console application, and create a new Heap Class. For now, our heap class will be created to store just int-type variables, but you can follow along using templates if you feel that you are able to.

   For this tutorial, this is what the class definition will look like:

```
class Heap
{
public:
	Heap();
	~Heap();

	//Functions for looking at/removing the values at the top of the heap
	int Peek();
	void Pop();

	//Clear all values from the heap
	void Clear();

	//Push/Remove values from the heap
	void Push(int value);
	void Remove(int value);

	//Get information about the heap itself
	unsigned int GetSize();
	unsigned int GetMaxSize();

	//Set the minimum amount of space needed for this heap
	void SetMaxSize(int minSize);
private:

	int* m_pValueArray;
	unsigned int m_uiArraySize;
	unsigned int m_uiCurrentNumberOfValues;
};
```

   You may also like to add an “IsEmpty” function in to quickly check if the heap is empty.

2. Using the information provided in the slides, implement all of the functions in this class.

3. Test your Heap by pushing on a set of 15 random numbers. When you Peak/Pop, they should all come off the heap in the correct order (either ascending or descending, based on the type of binary heap that you created).