CSE 2025 DATA STRUCTURES PROJECT#2

A BST With 3log(4n) Depth Levels That Provides the Best Possible Access Time

Class: Cse 2025 Data Structures

Project Subject/Problem :

A BST With 3log(4n) Depth Levels That Provides the Best Possible Access Time

Name: Mustafa Tunahan

Surname : BAŞ

Student No: 150119055

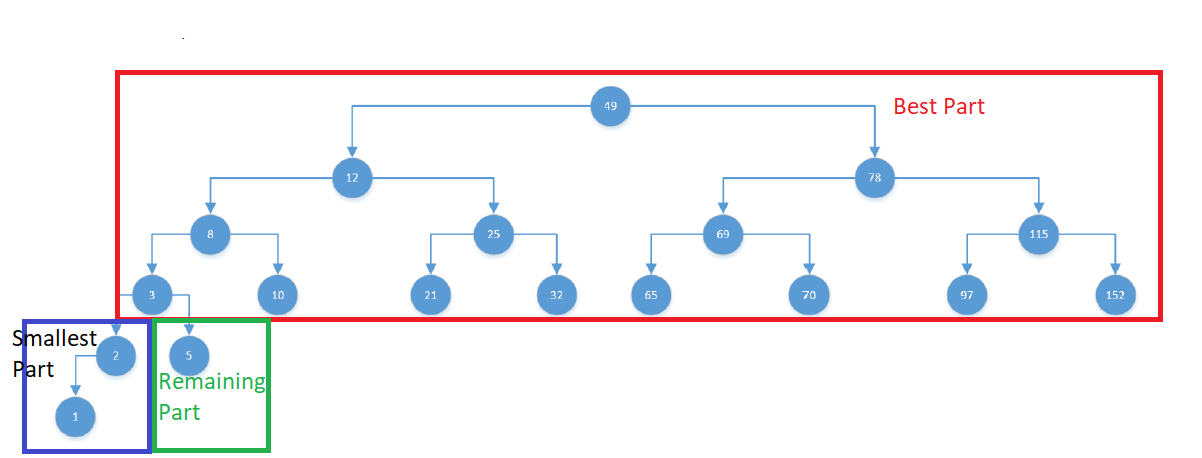
Submitted To : Birol GENÇYILMAZ

Project Due Date : 05.11.2021 17.00

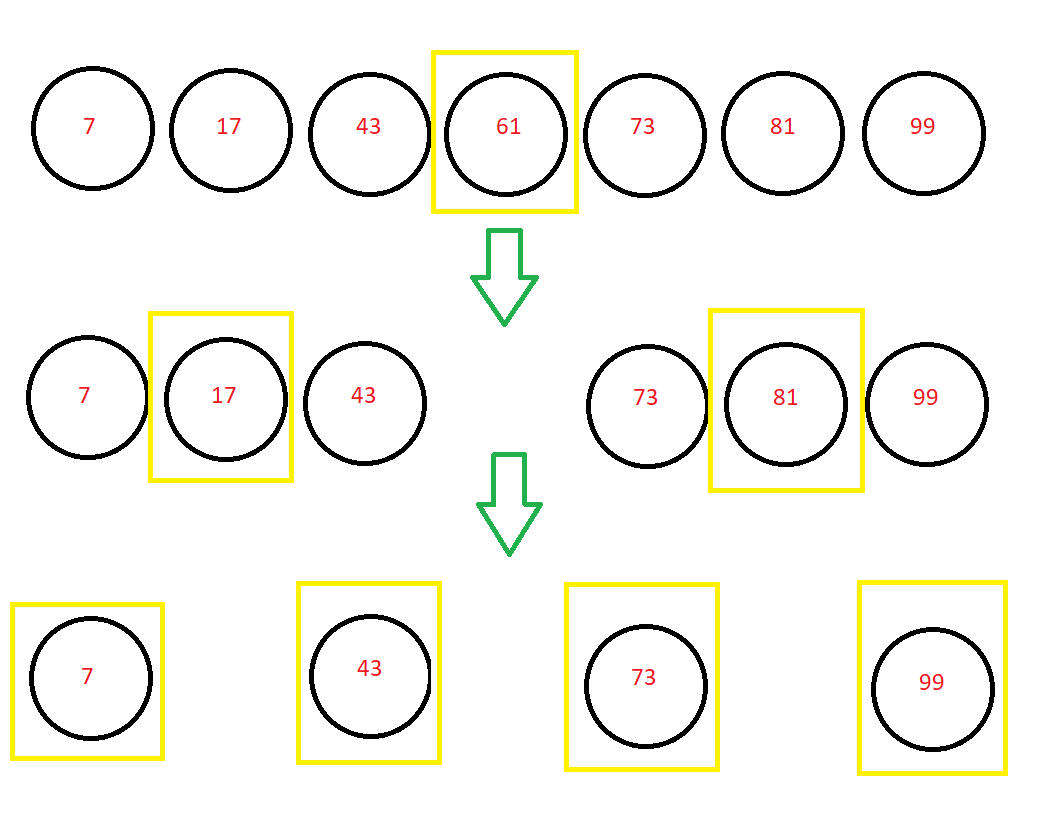
**Problem :** Constructing a BST whose nodes are distributed over 3log(4n) depth levels and that provides the best possible access time.

**My Approach and Algorithm**

1. Get the data from input folder and validate it and convert the string into an integer array
2. Sort the array,and let numbers = sorted array
3. Split the array into 3 arrays (**arrBest,arrSmallest,arrRemaining**) that will form the BST like this :



1. Calculate **a** = Floor() where **a** is the last depth level that the nodes in the Best Part(red rectangle) will fill
2. Calculate **b** = where b is the number of the nodes in the Best Part
3. Calculate **c** = **numbers.Length – b** where **c** is the total number of nodes in the Smallest Part(blue rectangle) and Remaining Part(green rectangle)
4. Calculate **d** = Round(3log(4\****numbers.Length***)) where **d** is the deepest depth level that should be reached to meet the requirements of the topology
5. Calculate **e = d-a** where **e** is the number of nodes in the smallest Part
6. Go to A. if **e - a > c** (If there are not enough remaining nodes [blue+green nodes after calculating the red part]) ***(Only 4 and 8 nodes cause this issue and since we want at least 16 nodes,the program will actually never execute that)***
   1. Assign **a = a-1**
   2. Go to b) to calculate **b** again
   3. Go to c) to calculate **c** again
7. Remove the first **e** elements from the **numbers** array and fill the **arrSmallest** with them (without breaking the order)
8. Calculate **f=c-e** where **f** is the number of the nodes in the remaining part (green part)
9. Remove the first **f** odd indexed elements from the **numbers** array and fill the **arrRemaining** with them (withour breaking the order)
10. Put all the remaining elements in the **numbers** array into **arrBest** in order
11. Realize that **arrBest** is a sorted array and if you insert them right away into the BST you’ll create a linkedlist
12. Goto 6. and call the function with **bestArr**
13. (Recursive) Loop while given array has more than 0 elements
    1. Remove the median from the array and insert to the BST (if BST is null then create it)
    2. Call the function 5. with the array that’s formed with the keys that are on the left side of the removed median
    3. Call the function 5. with the array that’s formed with the keys that are on the right side of the removed median (The order of left and right array doesn’t actually matter)



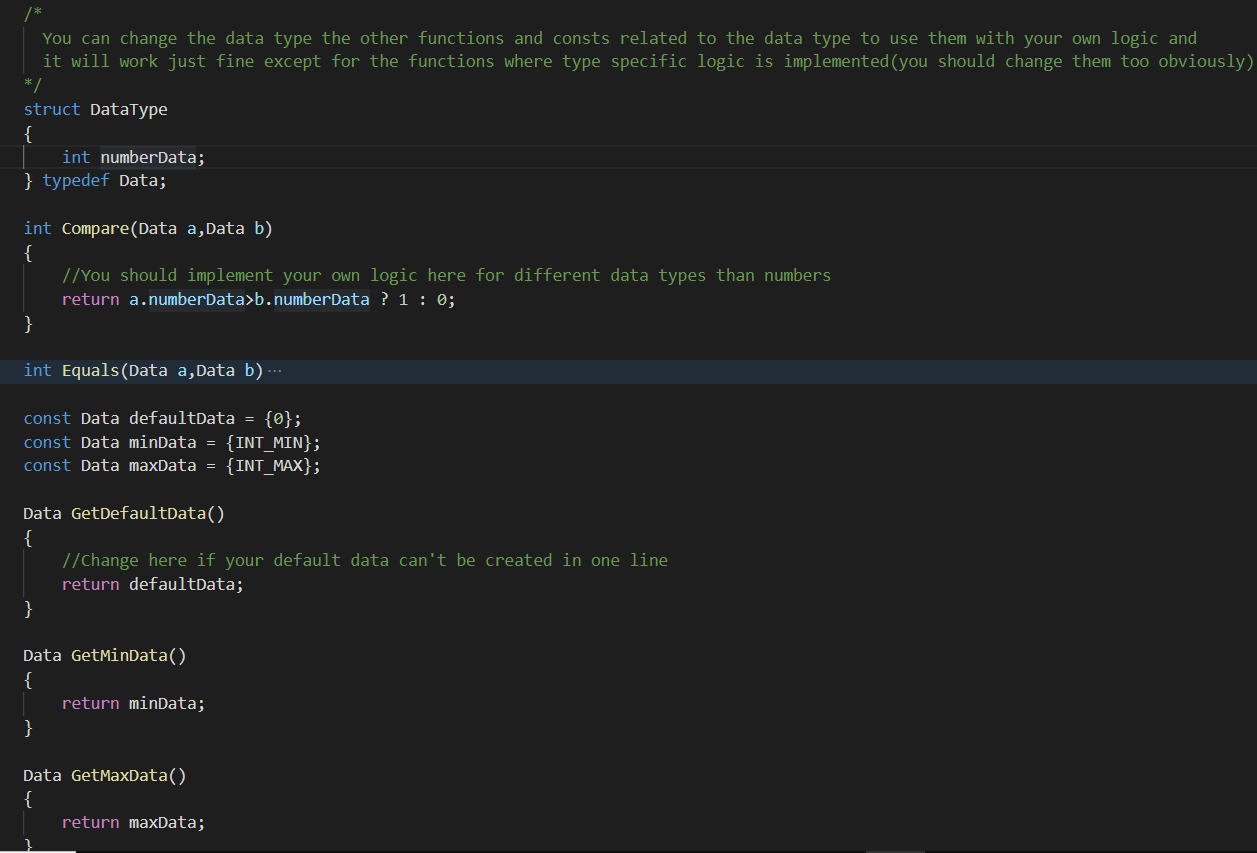
1. Insert the keys in the **arrSmallest** to the BST (in order)
2. Insert the keys in the **arrRemaining** to the BST (in order)

**Implementation in C**

**a.Data Structures**

* DataType
* DataArrayHolder
* BSTNode

**1-DataType**



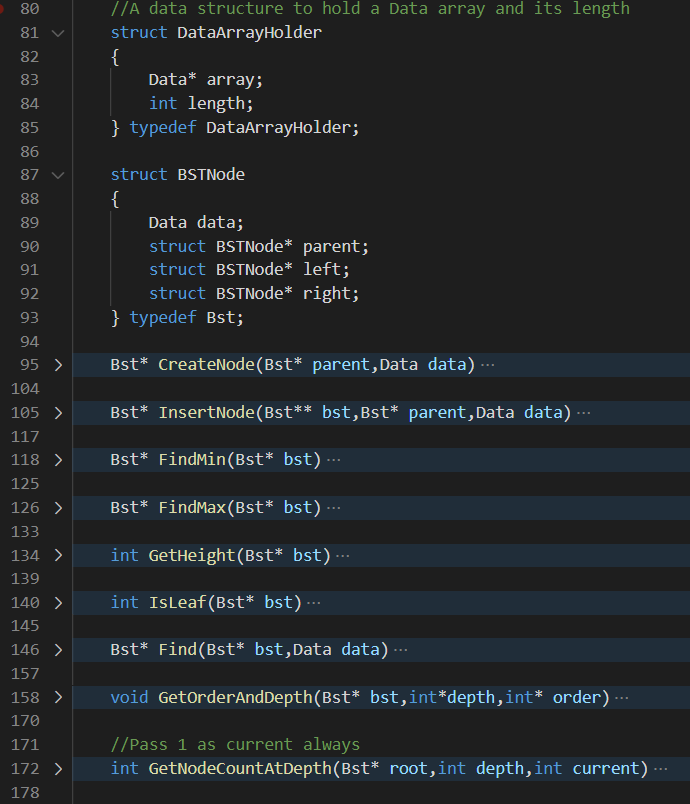
DataType(or Data) is the struct that is used as a container for the data.The reason I encapsulated the data inside it is for flexibility in case someone wanted to construct trees of different types.In that case all he has to do would be just change the field inside DataType struct and other functions’ bodies related to DataType (Compare,Equals,GetDefaultData,GetMinData,GetMaxData).

**Functions**

1. **int Compare(Data,Data)** compares two Data variables and returns 0 or 1 depending on the logic implemented(it currently returns 1 if the first number is greater or equal to the second variable,otherwise 0)
2. **int Equals(Data,Data)** compares two Data variables and returns 1 if they’re equal depending on the implemented logic,otherwise returns 0
3. **Data GetDefaultData()** returns the default data
4. **Data GetMinData()** returns the minimum data
5. **Data GetMaxData()** returns the maximum data

**2-DataArrayHolder**

This is a struct that consists of a Data array and its length,I think encapsulating an array and its length inside one struct helps increase the code readability especially of the function signatures that has lots of returns as parameters.



**3-BSTNode**

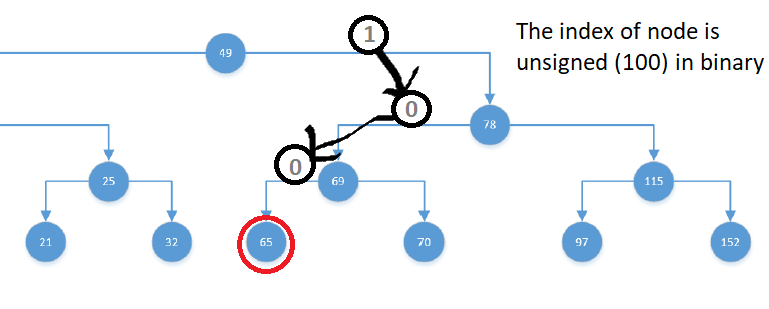
BSTNode (or Bst) is a simple binary search tree (BST) node with an extra field of parent node pointer inside it.Parent field is used for counting depths and order of a node in its depth level.

**Functions**

1. **Bst\* CreateNode(Bst\* parent,Data data)** creates a BST node and assigns its data and parent node
2. **Bst\* InserNode(Bst\* bst,Bst\*\* parent,Data data)** inserts a node into a tree given by ‘Bst\* bst’ and returns the inserted node
3. **Bst\* FindMin(Bst\* bst)** goes to the leftmost child(deepest left child) of a given node and retrieves its data
4. **Bst\* FindMax(Bst\* bst)** goes to the rightmost child of a given node and retrieves its data
5. **int GetHeight(Bst\* bst)** recursively gets the height of its child nodes and calculates their maximum to get the total height of the tree
6. **int IsLeaf(Bst\* bst)** returns 1 if a given node is a leaf,0 otherwise
7. **Bst\* Find(Bst\* bst,Data data)** function finds the node that holds the given data recursively(if it exists) and returns the pointer to it
8. **void GetOrderAndDepth(Bst\* bst,int\* depth,int\* order)** function recursively goes towards the parent and calculates its order and depth
9. **int GetNodeCountAtDepth(Bst\* root,int depth,int current)** function recursively adds up the nodes at the given depth level and returns their count

**Algorithm behind GetOrderAndDepth**

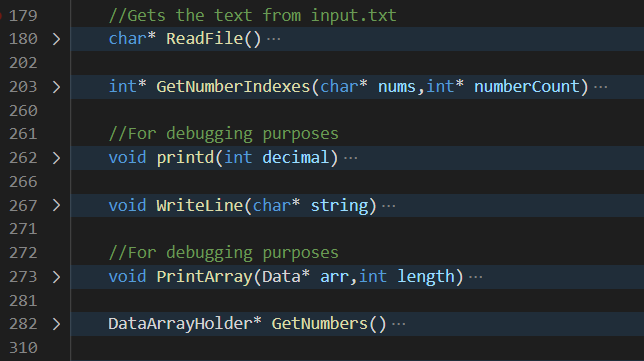
1. Given a node,say depth = 0 at its level and create an unsigned binary integer index = 0 and put a cursor to the rightmost digit of it
2. Loop while bst is not NULL and do;
3. Assign 1 to the cursor digit of the index if bst is its parent’s right child,otherwise 0,
4. Move the cursor to left by one
5. Assign bst.parent to bst



So in this case the index of our node 65 is which is equal to 4,so if the index of the node is 4 then its order in the depth must be 5.

**b.File-IO**

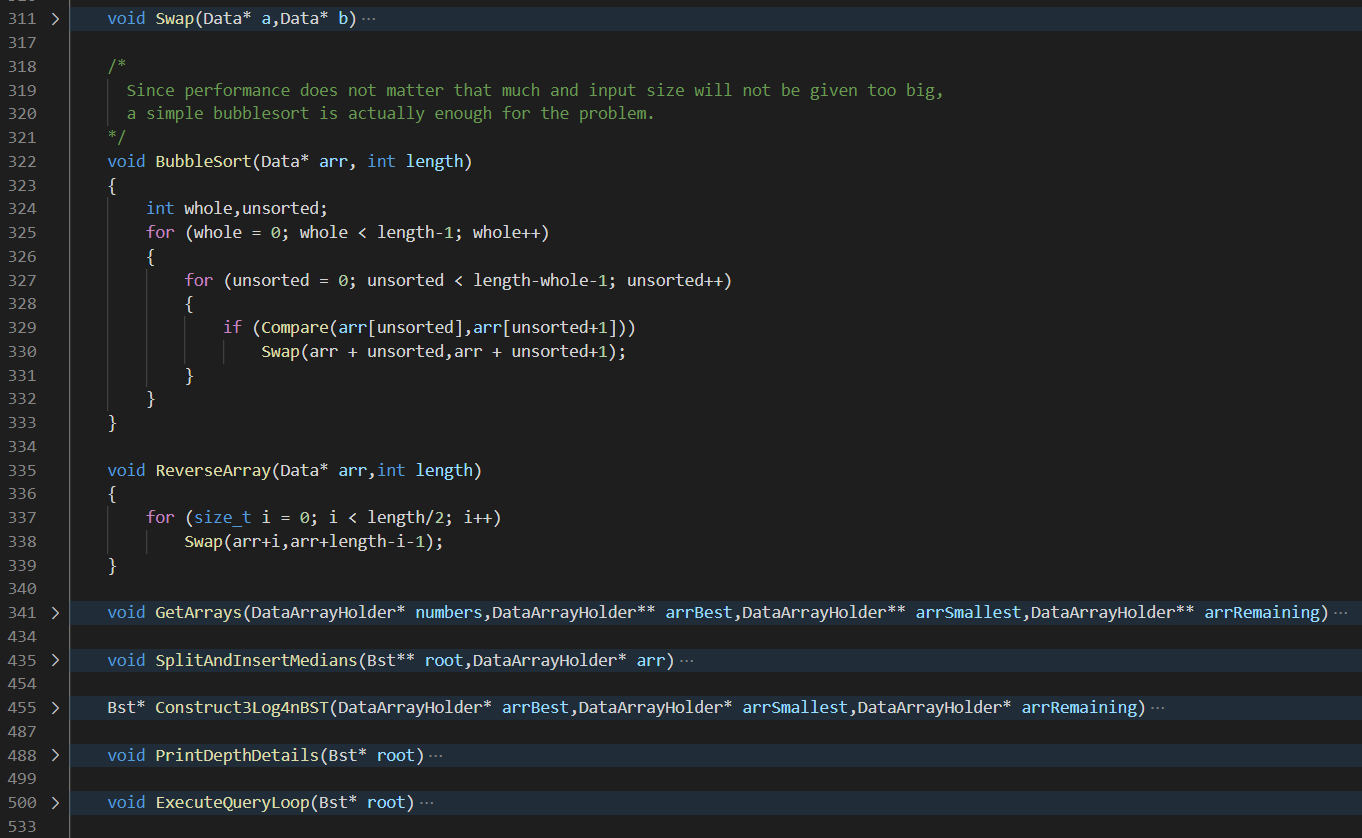
Keys to insert to BST are taken from a text file input.txt,in there the numbers are separated by spaces and all of them should be strictly positive,and there should be at least 16 numbers.



**Functions**

1. **Char\* ReadFile()** function read the entire file and constructs a string out of it and simply returns it
2. **int\* GetNumberIndexes(char\* nums,int\* numberCount)** takes in the input string and locates the start indexes of numbers inside it and converts every separator character(space) to NULL so that I can treat every start index of number in the nums text as a separate string and turn them into an integer array
3. **void printd(int decimal)** prints out an integer to the console
4. **void PrintArray(Data\* arr,int length)** prints a given amount of elements of an array
5. **DataArrayHolder\* GetNumbers()** calls the ReadFile and GetNumberIndexes functions and constructs the numbers and returns them in an integer array to be sorted laters

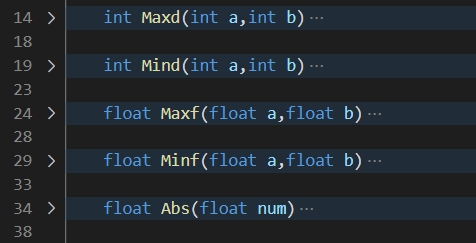
**c.Algorithms**



**Functions**

1. **void Swap(Data\*,Data\*)** function takes in two Data pointers and swaps them, it’s used in Bubblesort implementation and ReverseArray
2. **void BubbleSort(Data\* arr,int length)** is just a simple implementation of bubblesort which is a basic sorting algorithm.The reason I used it is because it’s simple to implement and performance didn’t matter that much
3. **void ReverseArray(Data\* arr,int length)** function takes in a Data array and its length and reverses it
4. **void GetArrays(DataArrayHolder\* numbers,DataArrayHolder\*\* arrBest,DataArrayHolder\*\* arrSmallest,DataArrayHolder\*\* arrRemaining)** is the function that takes in the numbers array returned from GetNumbers function and splits them into 3 arrays (arrSmallest,arrBest,arrRemaining) with an algorithm that when inserted in right order guarantees 3log(4n) depth levels in the BST
5. **void SplitAndInsertMedians(Bst\*\* root,DataArrayHolder\* arr)** function takes in a Data array (arrBest) and a double pointer to a BST node and inserts the array elements recursively with an algorithm that guarantees (if the BST is empty) the BST to be a best access time tree with its last depth level completely filled (arrBest hass elements).That function is called from Construct3Log4nBST
6. **Bst\* Construct3Log4nBST(DataArrayHolder\* arrBest,DataArrayHolder\* arrSmallest,DataArrayHolder\* arrRemaining)** function takes in the three sorted arrays returned from GetArrays function and inserts them in an empty BST in an order that will ensure 3log(4n) depth levels and best possible access time
7. **void PrintDepthDetails(Bst\*)** takes in the BST root and prints the deepest depth level and node count in each depth level
8. **void ExecuteQueryLoop(Bst\*)** takes in the BST root and puts the program in an infinite loop where the user asked to search for a particular node in the BST and is prompted either the node doesn’t exist or the location of the node in the BST till the user decides to enter 0 (terminator value) and exit the program

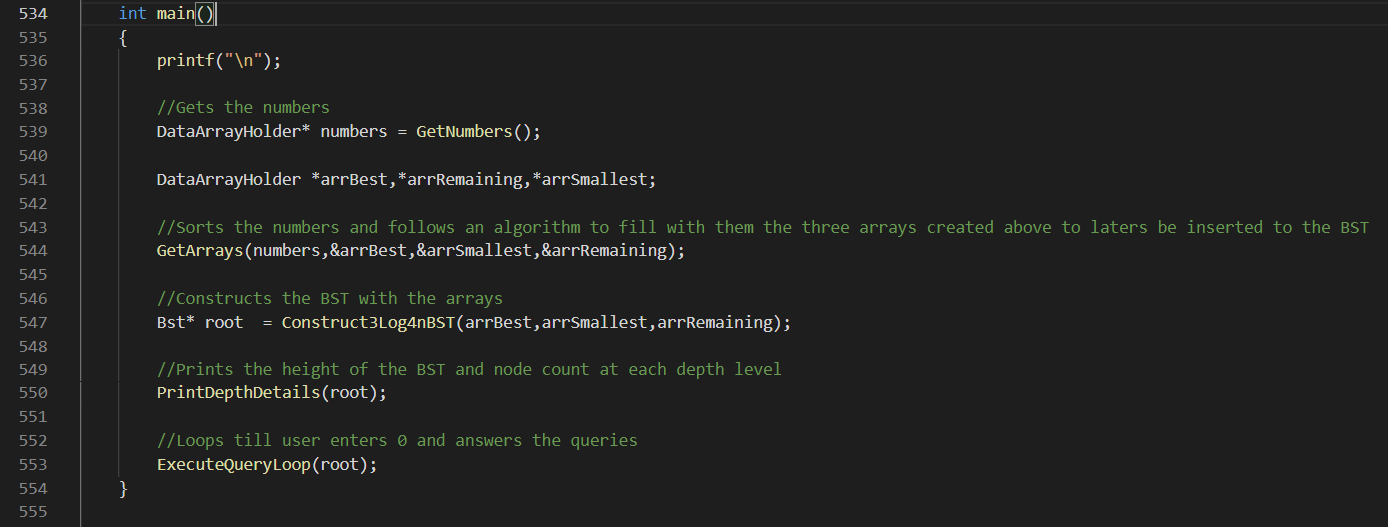
**d.Helper Math Functions**



**Functions**

1. **int Maxd(int a,int b)** function takes in two integers and returns the greater one
2. **int Mind(int a,int b)** function takes in two integers and returns the smaller one
3. **float Maxf(float a,float b)** function takes in two floats and returns the greater one
4. **float Minf(float a,float b)** function takes in two floats and returns the smaller one
5. **float Abs(float num)** function takes a float and returns its absolute value

**e.The main Function and Program Flow**

****

* First,an indentation from above is printed for a better view
* Then,**numbers** are retrieved from GetNumbers function
* Then,**arrSmallest,arrBest** and **arrRemaining** are calculated calling theGetArrays function
* Then the BST is constructed in the Construct3Log4nBST function
* Last depth level and node counts at each depth level are printed in the PrintDepthDetails function
* Finally the infinite loop with which the user will be asked for a data to seacrh in the BST and either is prompted that that data doesn’t exist or its locations in the BST
* Program will terminate whenever the user enters 0 (zero)

**f.An Example Run With 100000 Numbers**

