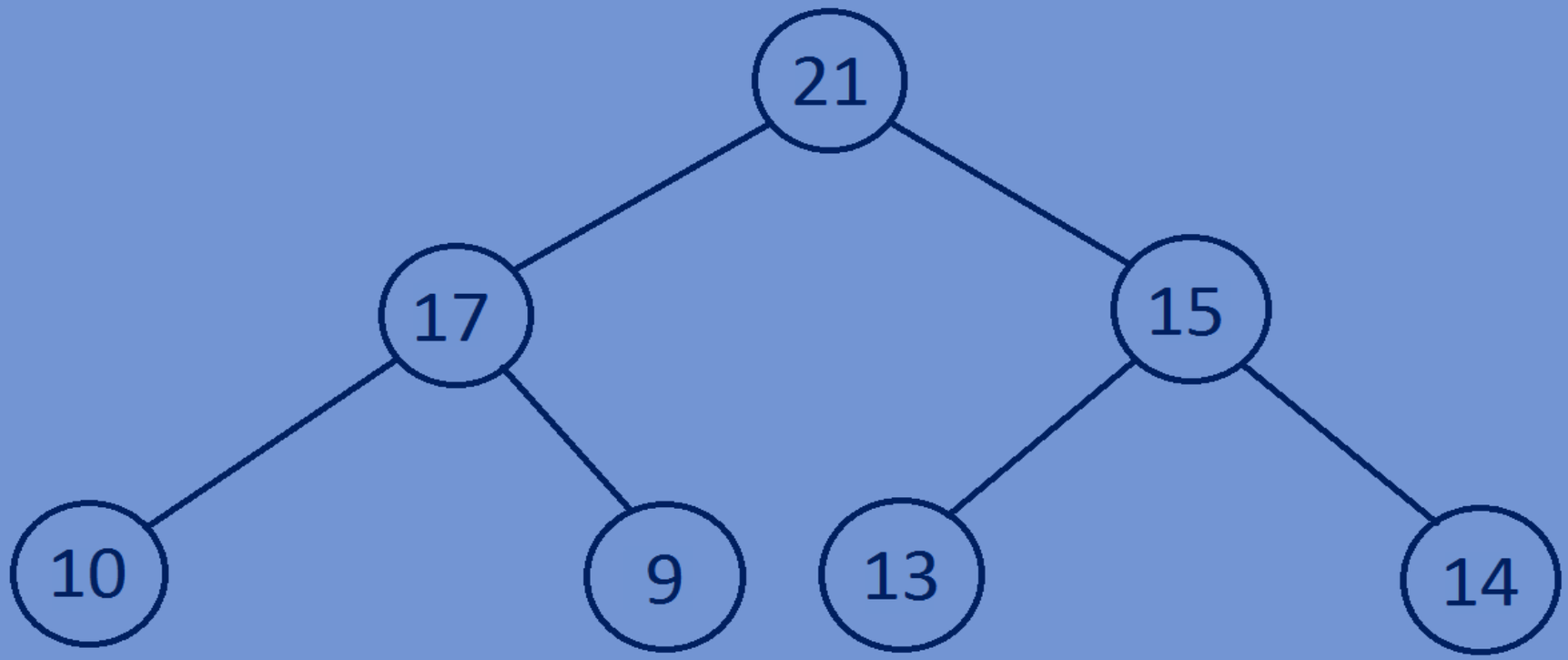
Data Structures

Implementation of Max Heap

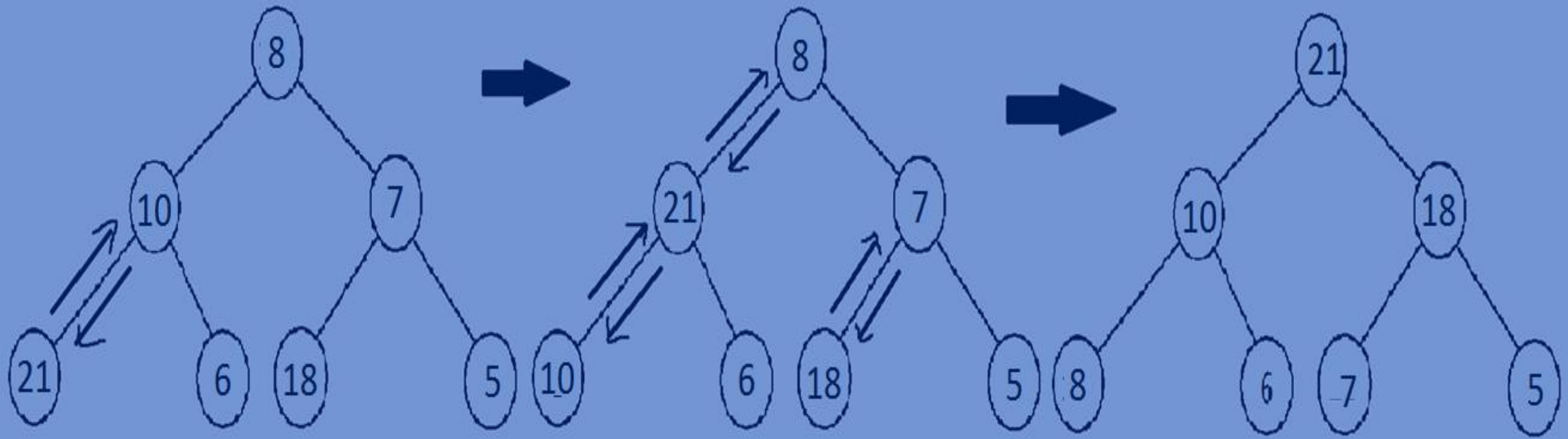
Max binary heap:

- In a max heap the data of the root node must be greater than or equal to its Children nodes data and this property holds for all the nodes of the max Binary heap.
- In case of max heap, the root node will have the largest value compared to All other nodes.



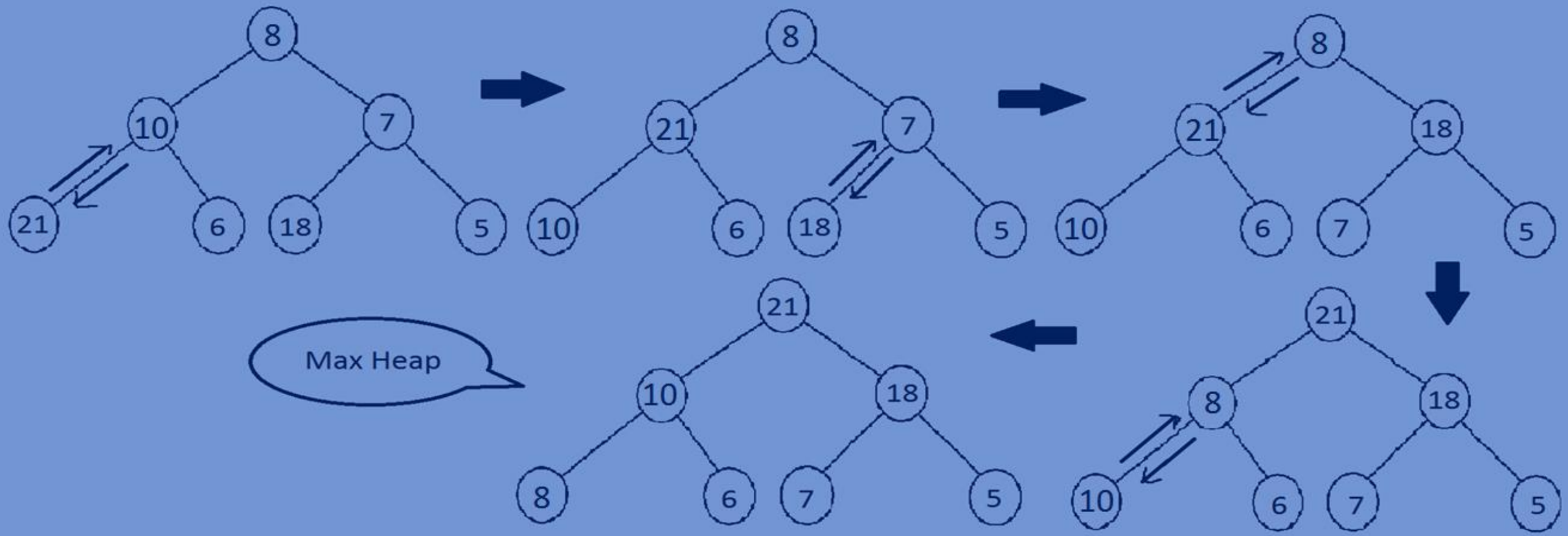
Implementation of Max binary heap:

- Given a complete binary tree convert the complete binary tree into a max binary heap.
- For a given node, heapify the left subtree,similarly heapify the right subtree And after that if necessary bubble down the current node such that the tree Rooted with the current node satisfies max heap property.



Implementation of Max binary heap using a complete binary tree:

- First traverse through each node of a complete binary tree in preorder fashion and while traversing compare the Data in the left child of the node with data of the node and if data in left child is greater than data in the parent node then swap the Data and call the recursive function with left subtree,if the data in right Child is greater than data in the parent node swap the data and call the recursive function with the right subtree.



Function for heapifying a complete binary tree:

```
//function for heapifying of a complete binay tree into max heap
void Maxheapify(Bstree *root){
    if (root == NULL) {
        return;
    }
    Maxheapify(root->left);
    Maxheapify(root->right);
    BubbleDown(root);
}
//recursive function for bubbling the nodes based on comparison.
void BubbleDown(Bstree *node){
    Bstree *largestNode;
    if(node==NULL){
        return;
    }
    if(node->left!=NULL && node->left->data > node->data){
        largestNode=node->left;
        swapData(node,largestNode);
        BubbleDown(largestNode);
```

```
    }
    if(node->right!=NULL && node->right->data > node->data){
        largestNode=node->right;
        swapData(node,largestNode);
        BubbleDown(largestNode);
    }
}
//swaping data of two nodes
void swapData(Bstree *temp1,Bstree *temp2){
    int temp3;
    temp3=temp1->data;
    temp1->data=temp2->data;
    temp2->data=temp3;
}
```

Whole program:

```c
#include<stdio.h>
#include<stdlib.h>
//creating a node
typedef  struct Bstree{
    int data;
    struct Bstree *left;
    struct Bstree *right;
}Bstree;
//creating new nodes
Bstree *createnewnode(int data){
    Bstree *newnode=(Bstree*)malloc(sizeof(Bstree));
    newnode->data=data;
    newnode->left=NULL;
    newnode->right=NULL;
    return newnode;
}
//swaping data of two nodes
void swapData(Bstree *temp1,Bstree *temp2){
```

```
    int temp3;
    temp3=temp1->data;
    temp1->data=temp2->data;
    temp2->data=temp3;
}
//recursive function for bubbling the nodes based on comparison.
void BubbleDown(Bstree *node){
    Bstree *largestNode;
    if(node==NULL){
        return;
    }
    if(node->left!=NULL && node->left->data > node->data){
        largestNode=node->left;
        swapData(node,largestNode);
        BubbleDown(largestNode);
    }
    if(node->right!=NULL && node->right->data > node->data){
        largestNode=node->right;
        swapData(node,largestNode);
        BubbleDown(largestNode);
    }
}
```

```c
//function for heapifying of a complete binay tree into max heap
void Maxheapify(Bstree *root){
    if (root == NULL) {
        return;
    }
    Maxheapify(root->left);
    Maxheapify(root->right);
    BubbleDown(root);
}
//print all the nodes of a tree in preorder fashion
void preorder(Bstree *root){
    if(root){
        printf("%d ",root->data);
        preorder(root->left);
        preorder(root->right);
    }
}
//main function
int main(){
    Bstree *tree=NULL;
    tree=createnewnode(8);
    tree->left=createnewnode(10);
```

```
    tree->right=createnewnode(9);
    tree->left->left=createnewnode(2);
    tree->left->right=createnewnode(7);
    tree->right->left=createnewnode(1);
    tree->right->right=createnewnode(3);
    preorder(tree);printf("\n");
    Maxheapify(tree);
    preorder(tree);
    return 0;
}
```

Output:

8 10 2 7 9 1 3

10 8 2 7 9 1 3