BTrees :-

```
Class Btreechild
{
    *Keys ; //. array of Keys
    Btreechild ** Child ;
    n // Total Keys
    order
    bool leaf
}

Class Btree    // Friend class of B treechild
{
    insert (int ~~key~~ data)
    {
        if (root == NULL)
            insert value into root
            and increase no. of Keys
    else
        if (root is full)
        {
            Allocate memory for root
            Btreenode *nroot
            nroot → child[0] = root
            (split the old root) -
        }
        else
            root → insert.empty(~~keys~~ data)
}

insertempty (~~keys~~ data)   // called when the
                                 node is not full
{
    if (leaf is true)
    {   i =  n-1
        while( i >=0 && Key[i] > data
         {  Keys [i+1] = Keys [i]
            i-- ;
          }
    Key[i+1] = data
     n++
    }
    else   // If not leaf
    {
        find (child)   // which has new key
        if (Child is full)
            split child ( i+1, child [i+1])

        if (keys[i+1] < data)
            i++
    } child[i+1] → insertempty(data)
}
```

```
split childnode ( i, Btreechild * C1)
{
        Btreechild * C2;
        C2 -> n = Order - 1        // no. of Keys
       g = 0
        While (g < order-1)
         C2 -> Keys[g] = C1 -> Keys[g+order]
        // order-1 Keys of C1 in C2
        // then copy order no. of child of C1 in C2
           C2 -> child[i] = C1 -> child[i + order]

        Child [i+1] = C1
            // move all Keys one space
           than increment total no of Keys.
}
```