ALGORITHM

1. Start
2. Set typedef struct Node
    2.1. Declare info
    2.2. Set struct Node *lchild, *rchild
3. Set node
4. Set node *root=NULL
5. newnode(val)
    5.1. Set node*p=(node*)malloc(sizeof(node))
    5.2. Set p->info=val
    5.3. Set p->lchild=p->rchild=NULL
    5.4. return p
6. End function
7. insert(val)
    7.1. Check if(root=NULL)
        7.1.1. Set root=newnode(val)
    7.2. else
        7.2.1. Set node* par=NULL
        7.2.2. Set node* curr=root
        7.2.3. while(curr!=NULL)
            7.2.3.1. Set par=curr
            7.2.3.2. Check if(val<curr->info)
                7.2.3.2.1. Set curr=curr->lchild
            7.2.3.3. else
                7.2.3.3.1. Set curr=curr->rchild
        7.2.4. Check if(val<par->info)
            7.2.4.1. Set par->lchild=newnode(val)
        7.2.5. else
            7.2.5.1. Set par->rchild=newnode(val)
8. End function
9. inorder(node* p)
    9.1. Check if(p!=NULL)

Date: 30/10/24

PROGRAM NO: 19
BINARY SEARCH TREE

Aim: To implement binary search tree

PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
typedef struct Node
{
   int info;
   struct Node *lchild, *rchild;
}node;
node *root=NULL;

node* newnode(int val)
{
   node*p=(node*)malloc(sizeof(node));
   p->info=val;
   p->lchild=p->rchild=NULL;
   return p;
}

void insert(int val)
{
   if(root==NULL)
      root=newnode(val);
   else
   {
      node* par=NULL;
      node* curr=root;
      while(curr!=NULL)
      {
         par=curr;
         if(val<curr->info)
            curr=curr->lchild;
         else
            curr=curr->rchild;
      }
      if(val<par->info)
         par->lchild=newnode(val);
      else
         par->rchild=newnode(val);
   } }

void inorder(node* p)
{
   if(p!=NULL) {
      inorder(p->lchild);
      printf("%d\t", p->info);
      inorder(p->rchild); } }
```

9.1.1. Set inorder(p->lchild)

9.1.2. Print, p->info

9.1.3. Set inorder(p->rchild)

10. End function

11. main()

11.1. Declare choice, val

11.2. do

11.2.1. Print, 1. Insert 2. Display 7. Quit

11.2.2. Print, Enter your choice

11.2.3. Read the choice

11.2.4. switch(choice)

11.2.4.1. case 1: Print, Enter the element to be inserted

11.2.4.1.1. Read the element

11.2.4.1.2. Set insert(val)

11.2.4.2. case 2: inorder(root)

11.2.4.3. case 3: Print, Exiting

11.2.4.4. default: Print, Invalid choice.

11.3. while(choice!=3);

12. End function

13. Stop

OUTPUT

```
1. Insert
2. Display
3. Quit: 1
Enter the element to be inserted:  1
1. Insert
2. Display
3. Quit: 1
Enter the element to be inserted:  2
1. Insert
2. Display
3. Quit: 1
Enter the element to be inserted:  0
1. Insert
2. Display
3. Quit: 2
0 1 2
1. Insert
2. Display
3. Quit: 3  Exiting…
```

```
void main()
{
   int choice,val;
     do
     {
     printf("\n1. Insert\n2. Display\n3. Quit:\n");
     printf("Enter your choice\n");
     scanf("%d", &choice);
     switch(choice)
     {
          case 1:
                    printf("Enter the element to be inserted:\n");
                    scanf("%d", &val);
                    insert(val);
                    break;
          case 2:
             inorder(root);
             break;
          case 3: printf("Exiting...\n");
                    break;
          default:
                    printf("Invalid choice.\n Enter a valid choice from 1 to 3\n");
     }
     }while(choice!=3);
     }
```

Result:

The program is executed successfully and output is obtained