Insertion Into AVL Tree

Code Id	14
Date Updated	3/7/2010
Title	Insertion into AVL Tree

Description

This is a program for insertion into an AVL tree.

Codes Snippet

```
 #include
#include
typedef enum { FALSE, TRUE} bool;
struct node
{
int info;
int balance;
struct node *lchild;
 struct node *rchild;
} ;
struct node *insert (int, struct node *, int *);
struct node* search(struct node * ,int);
main( )
{
bool ht_inc;
int info;
int choice;
struct node *root = (struct node *)malloc(sizeof(struct node);
 root = NULL;
while(1 )
{
printf(" 1.Insertn");
printf("2.Displayn");
 printf("3 .Quitn "); printf("Enter your choice: ");
scanf("%d" ,&choice);

switch( choice)
{
case 1:
printf("Enter the value to be inserted: ");
scanf("%d", &info);
if( search(root,info) == NULL )
root = insert(info, root, &ht_inc);
else
printf("Duplicate value ignoredn");
break;
case 2:
if(root NULL)
{
printf("Tree is emptyn");
continue;
}
printf("Tree is :n");
display(root, 1);
printf("nn");
printf("Inorder Traversal is: ");
inorder(root);
printf("n");
break;
case 3:
        exit(1);
        default:
        printf("Wrong choicen");
        }/*End of switch*/
        }/*End of while*/
}/*End of main( )*/
struct node* search(struct node *ptr,int info)
 {      if(ptr!=NULL)
                if(info < ptr->info)
                ptr=search(ptr->lchild,info );
                else if( info> ptr->info)
```


Recent Posts

Types of Cloud Computing
What is Cloud Computing ?
How to pass a multi-dimensional array to a function?
Memory Layout of a C Program
PHP and Its Advantages

```
pptr->1child = NULL;
pptr->rchild = NULL;
pptr->balance = 0;
*ht_inc = TRUE;
return (pptr);
}
if(info < pptr->info)
{
pptr->1child = insert(info, pptr->1child, ht_inc);
if(*ht_inc-- TRUE)
{
switch(pptr->balance)
 {
                        case -1: /* Right heavy */
                                pptr->balance = 0;
*ht_inc = FALSE;
                                break;
case 0: /* Balanced */
pptr->balance = 1;
break;
case 1: /* Left heavy */
aptr = pptr->lchild;
 if(aptr->balance == 1)
{
printf("Left to Left Rotationn");
 pptr->lchild= aptr->rchild;
 aptr->rchild = pptr;
pptr->balance = 0;
aptr->balance=0;
pptr = aptr;
}
else
 {
printf("Left to right rotationn");
 bptr = aptr->rchild;
 aptr->rchild= bptr->lchild;
 bptr->lchild = aptr;
 pptr->lchild = bptr->rchild;
bptr->rchild = pptr;
if(bptr->balance == 1 )
                                                pptr->balance = -1;

else
                                                        pptr->balance = 0;
if(bptr->balance== -1)
                                                        aptr->balance = 1;
else
aptr->balance = 0;
 bptr->balance=0;
 pptr=bptr;
}
*ht_inc = FALSE;

                        } /*End of switch * / .
}/*End of if*/
}/*End of if*/
if(info > pptr->info)
 {
pptr->rchild = insert(info, pptr->rchild, ht_inc);
if(*ht_inc==TRUE)
{
switch(pptr->balance)
{
case 1: /* Left heavy */
 pptr->balance = 0;
*ht_inc = FALSE;
                                        break;

case 0: /* Balanced */
pptr->balance = -1;
 break;
case -1: /* Right heavy * /
aptr = pptr->rchild;
if(aptr->balance = -1)
{
printf("Right to Right Rotationn");
pptr->rchild= aptr->lchild;
aptr->lchild =pptr;
pptr->balance = 0;
```

```
rptr->balance = 1;
else
                                pptr->balance = 0;
if(bptr->balance== 1)
                                aptr->balance = -1;
else
aptr->balance = 0;
bptr->balance=0;
pptr = bptr;
}/*End of else*/
*ht_inc = FALSE;
        } /*End of switch * /
}/*End of if*/
 }/*End of if*/
 return(pptr );
}/*End of insert( )*/

display(struct node *ptr,int level)
{
int i;
if( ptr!=NULL)
{
display(ptr->rchild, level+ 1);
printf("n");
        for (i = 0; i < level; i++)
        printf(" ");
        printf("%d", ptr->info);
        display(ptr->lchild, level+ 1);
        }/*End of if */
}/*End of display( )*/
inorder(struct node *ptr)
 {
if(ptr!=NULL)
{
inorder(ptr -> lchild);
 printf("%d ,ptr->info);
 inorder(ptr ->rchild);
}
}/*End of inorder( )*/
```