//BST Non-recursive

#include<stdio.h>

#include<stdlib.h>

typedef struct Node

{

struct Node \*left;

int info;

struct Node \*right;

}bst;

bst\* insert(bst \*rt,int key)

{

bst \*p=NULL,\*ptr=NULL,\*prnt;

p=(bst\*)malloc(sizeof(bst));

if(p!=NULL)

{

p->info=key ;

p->left=NULL;

p->right=NULL;

if(rt==NULL)

rt=p;

else

{

ptr=rt;

prnt=NULL;

while(ptr!=NULL)

{

prnt=ptr;

if(key<ptr->info)

ptr=ptr->left;

else

ptr=ptr->right;

}

if(key<prnt->info)

prnt->left=p;

else if(key>prnt->info)

prnt->right=p;

else

printf("invalid info");

}

}

return rt;

}

void inorder(bst \*rt)

{

if(rt!=NULL)

{

inorder(rt->left);

printf("%d ",rt->info);

inorder(rt->right);

}

}

void preorder(bst \*rt)

{

if(rt!=NULL)

{

printf("%d ",rt->info);

preorder(rt->left);

preorder(rt->right);

}

}

void postorder(bst \*rt)

{

if(rt!=NULL)

{

postorder(rt->left);

postorder(rt->right);

printf("%d ",rt->info);

}

}

int main()

{

bst \*rt=NULL,\*temp=NULL;

int ch=1,k=0;

do

{

printf("\n1.insert 2. Inorder traversal 3.Preorder traversal 4.Postorder traversal");

printf("\n enter choice");

scanf("%d",&ch);

switch(ch)

{

case 1:printf("\n enter key/info to insert");

scanf("%d",&k);

rt=insert(rt,k);

break;

case 2:if(rt==NULL)

printf("empty........");

else

{

printf("\n BST Inorder traversal\n");

inorder(rt);

}

break;

case 3:if(rt==NULL)

printf("empty........");

else

{

printf("\n BST Preorder traversal\n");

preorder(rt);

}

break;

case 4:if(rt==NULL)

printf("empty........");

else

{

printf("\n BST Postorder traversal\n");

postorder(rt);

}

}

}while(ch<=4);

return 0;

}

//BST Recursive single ptr and double ptr

#include<stdio.h>

#include<stdlib.h>

typedef struct Node

{

struct Node \*left;

int info;

struct Node \*right;

}bst;

bst\* insert(bst \*rt,int key)

{

bst \*p=NULL;

if(rt==NULL)

{

p=(bst\*)malloc(sizeof(bst));

p->info=key ;

p->left=NULL;

p->right=NULL;

rt=p;

return rt;

}

if(key<rt->info)

rt->left=insert(rt->left,key);

else if(key > rt->info)

rt->right=insert(rt->right,key);

else

printf("Invalid/Duplicate key..........");

return rt;

}

void inorder(bst \*rt)

{

if(rt!=NULL)

{

inorder(rt->left);

printf("%d ",rt->info);

inorder(rt->right);

}

}

void preorder(bst \*rt)

{

if(rt!=NULL)

{

printf("%d ",rt->info);

preorder(rt->left);

preorder(rt->right);

}

}

void postorder(bst \*rt)

{

if(rt!=NULL)

{

postorder(rt->left);

postorder(rt->right);

printf("%d ",rt->info);

}

}

int main()

{

bst \*rt=NULL,\*temp=NULL;

int ch=1,k=0;

while(ch)

{

printf("\n enter key/info to insert");

scanf("%d",&k);

rt=insert(rt,k);

printf("\n press 1 to continue/0 to exit");

scanf("%d",&ch);

if(ch==0)

break;

}

printf("\n BST Inorder traversal\n");

inorder(rt);

printf("\n BST Preorder traversal\n");

preorder(rt);

printf("\n BST Postorder traversal\n");

postorder(rt);

}

// dbl ptr

#include<stdio.h>

#include<stdlib.h>

typedef struct Node

{

struct Node \*left;

int info;

struct Node \*right;

}bst;

void insert(bst \*\*rt,int key)

{

bst \*p=NULL;

if(\*rt==NULL)

{

p=(bst\*)malloc(sizeof(bst));

p->info=key ;

p->left=NULL;

p->right=NULL;

\*rt=p;

return;

}

if(key<(\*rt)->info)

insert(&(\*rt)->left,key);

else if(key>(\*rt)->info)

insert(&(\*rt)->right,key);

else

printf("Invalid/Duplicate key..........");

}

void inorder(bst \*rt)

{

if(rt!=NULL)

{

inorder(rt->left);

printf("%d ",rt->info);

inorder(rt->right);

}

}

void preorder(bst \*rt)

{

if(rt!=NULL)

{

printf("%d ",rt->info);

preorder(rt->left);

preorder(rt->right);

}

}

void postorder(bst \*rt)

{

if(rt!=NULL)

{

postorder(rt->left);

postorder(rt->right);

printf("%d ",rt->info);

}

}

int main()

{

bst \*rt=NULL,\*temp=NULL;

int ch=0,k=0;

do

{

printf("\n1.insert\n2.Inorder\n3. Preotder\n4.Postorder");

printf("\n enter your choice.....");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("\n enter key/info to insert");

scanf("%d",&k);

insert(&rt,k);

break;

case 2:

printf("\n BST Inorder traversal\n");

inorder(rt);

break;

case 3:

printf("\n BST Preorder traversal\n");

preorder(rt);

break;

case 4:

printf("\n BST Postorder traversal\n");

postorder(rt);

}

}while(ch<=4);

return 0;

}