**Cây nhị phân tìm kiếm:**

void insert(node \*root, int x){

if(root->data > x){

if(root->left) insert(root->left, x);

else root->left = new node (x);

}

if(root->data < x){

if(root->right) insert(root->right, x);

else root ->right = new node(x);

}

}

**Cây nhị phân tìm kiếm cân bằng:**

node \*buildTree(int a[], int left, int right){

if(left>right) return NULL;

int mid = (left+right)/2;

node \*root = new node(a[mid]);

root->left = buildTree(a, left, mid-1);

root->right = buildTree(a,mid+1, right);

return root;

}

**Cây nhị phân cơ bản:**

void makeNode(node \*root, int u, int v, char c){

if(c=='L') root->left = new node(v);

else root->right= new node(v);

}

void insertNode(node \*root, int u, int v, char c){

if(root == NULL) return;

if(root->data == u){

makeNode(root, u,v,c);

}

else{

insertNode(root->left, u, v, c);

insertNode(root->right, u,v,c);

}

}

**LevelOrder**

void levelOrder(node \*root){

queue<node\*> q;

q.push(root);

while(!q.empty()){

node \*tmp = q.front(); q.pop();

if(tmp != NULL){

cout << tmp->data <<" ";

q.push(tmp->left);

q.push(tmp->right);

**Biến đổi trung tố**

for (int i=0;i<s.size();i++){

if(s[i] == '('){

st.push(s[i]);

}

else if(!st.empty() && s[i] == ')'){

while(st.top() != '('){

res += st.top();

st.pop();

}

st.pop();

}

else if(isalpha(s[i])){

st.push(s[i]);

}

else{

while(!st.empty() && priority(st.top()) >= priority(s[i])){

char c = st.top();

st.pop();

res += c;

}

st.push(s[i]);

}

}

while(!st.empty()){

res += st.top();

st.pop();

}