```cpp
//BinaryTree
#include<iostream>
#include<stack>
#include<queue>
using namespace std;

class btNode {
    public:
        double value;
        btNode* left;
        btNode* right;
    public:
        btNode() {
            this->value=0.0;
            this->left=NULL;
            this->right=NULL;
        }
        btNode(double m_value=0.0) {
            this->value=m_value;
            this->left=NULL;
            this->right=NULL;
        }
        ~btNode() {
        }
        void setLeft(double m_value) {
            this->left=new btNode(m_value);
//          this->left->value=m_value;
        }
        void setRight(double m_value) {
            this->right=new btNode(m_value);
//          this->right->value=m_value;
        }
//      int printSelf();
        //三种非递归遍历（栈）
        int preGo1();
        int miGo1();
        int postGo1();
        //层序遍历 （队列）
        int levelGo();
        //三种递归遍历
        int preGo2();
        int miGo2();
        int postGo2();
};
```

```
45.
46. int btNode::preGo1() {
47.     stack<pair<btNode*,bool>> s;
48.     btNode *p;
49.     bool visited;
50.     //false 表示此点是第一次进栈
51.     s.push(make_pair(this,false));
52.     while(!s.empty()) {
53.         p=s.top().first;
54.         visited=s.top().second;
55.         s.pop();
56.         if(p==NULL){
57.             continue;
58.         }
59.         if(visited){
60.             cout<<"v: "<<p->value<<endl;
61.         }else{
62.             s.push(make_pair(p->right,false));
63.             s.push(make_pair(p->left,false));
64.             s.push(make_pair(p,true));
65.         }
66.     }
67.     return 0;
68. }
69.
70. int btNode::miGo1() {
71.     stack<pair<btNode*,bool>> s;
72.     btNode *p;
73.     bool visited;
74.     //false 表示此点是第一次进栈
75.     s.push(make_pair(this,false));
76.     while(!s.empty()) {
77.         p=s.top().first;
78.         visited=s.top().second;
79.         s.pop();
80.         if(p==NULL){
81.             continue;
82.         }
83.         if(visited){
84.             cout<<"v: "<<p->value<<endl;
85.         }else{
86.             s.push(make_pair(p->right,false));
87.             s.push(make_pair(p,true));
88.             s.push(make_pair(p->left,false));
```

```
        }
    }
    return 0;
}

int btNode::postGo1() {
    stack<pair<btNode*,bool>> s;
    btNode *p;
    bool visited;
    //false 表示此点是第一次进栈
    s.push(make_pair(this,false));
     while(!s.empty()) {
         p=s.top().first;
         visited=s.top().second;
         s.pop();
         if(p==NULL){
             continue;
         }
         if(visited){
             cout<<"v: "<<p->value<<endl;
         }else{
             s.push(make_pair(p,true));
             s.push(make_pair(p->right,false));
             s.push(make_pair(p->left,false));
         }
    }
    return 0;
}

int btNode::levelGo(){
    queue<btNode*> q;
    q.push(this);
    btNode* p;
    while(!q.empty()){
         p=q.front();
         q.pop();
         cout<<"v: "<<p->value<<endl;
         if(p->left!=NULL){
             q.push(p->left);
         }
         if(p->right!=NULL){
             q.push(p->right);
         }
    }
```

```
133. }
134. 
135. int btNode::preGo2() {
136.     cout<<"v: "<<this->value<<endl;
137.     if(this->left!=NULL) {
138.         this->left->preGo2();
139.     }
140.     if(this->right!=NULL) {
141.         this->right->preGo2();
142.     }
143.     return 0;
144. }
145. 
146. int btNode::miGo2() {
147.     if(this->left!=NULL) {
148.         this->left->miGo2();
149.     }
150.     cout<<"v: "<<this->value<<endl;
151.     if(this->right!=NULL) {
152.         this->right->miGo2();
153.     }
154.     return 0;
155. }
156. 
157. int btNode::postGo2() {
158.     if(this->left!=NULL) {
159.         this->left->postGo2();
160.     }
161.     if(this->right!=NULL) {
162.         this->right->postGo2();
163.     }
164.     cout<<"v: "<<this->value<<endl;
165.     return 0;
166. }
167. 
168. int main() {
169.     double a=0,b=1,c=2,d=3,e=4;
170.     //    0
171.     //  1   2
172.     // 3 4
173.     btNode *root=new btNode(a);
174.     root->setLeft(b);
175.     root->setRight(c);
176.     root->left->setLeft(d);
```

```
177.     root->left->setRight(e);
178.     cout<<"迭代式前序遍历："<<endl;
179.     root->preGo1();
180.     cout<<"迭代式中序遍历："<<endl;
181.     root->miGo1();
182.     cout<<"迭代式后序遍历："<<endl;
183.     root->postGo1();
184.     cout<<"----------------------"<<endl;
185.     cout<<"层序遍历："<<endl;
186.     root->levelGo();
187.     cout<<"----------------------"<<endl;
188.     cout<<"递归式前序遍历："<<endl;
189.     root->preGo2();
190.     cout<<"递归式中序遍历："<<endl;
191.     root->miGo2();
192.     cout<<"递归式后序遍历："<<endl;
193.     root->postGo2();
194. }
```