实验七 使用 STL

实验目的

1．掌握 VC 中 STL 的使用方法;

2．掌握容器（container）、模板（template）、游标（Iterator）、算法（Algorithms）、分配器（Allocator）、向量（vector）等知识的应用。

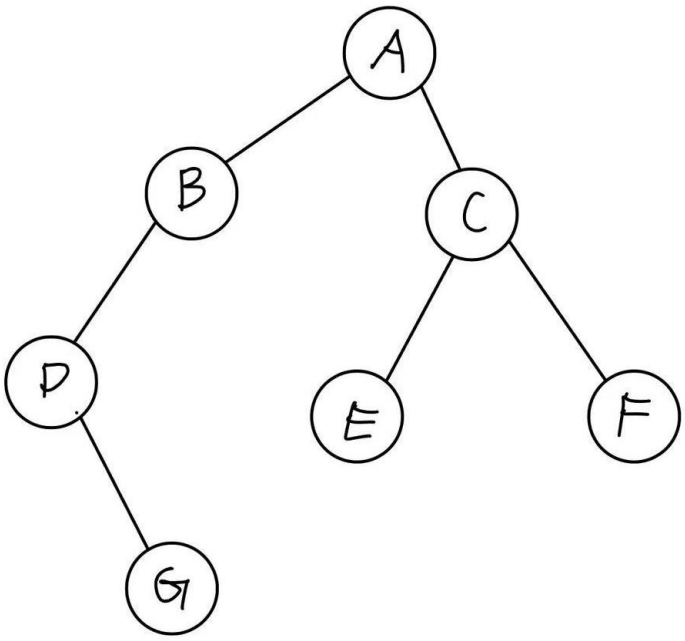
实验内容

1. 句子分解

有一段英文由若干单词组成，单词之间用一个空格分隔。编写程序提取其中所有单词。

```
void solve(string str, vector<string>& words);
int main()
{
    string str = "The following code solves a sentence into a string array";
    cout << str << endl;
    vector<string> words;
    solve(str, words);
    vector<string>::iterator it;
    for (it = words.begin(); it != words.end(); it++) cout << *it << endl;
}
```

2. 构造二叉树



由后序序列 post[] = "GDBEFCA"和中序序列 in[] = "DGBAECF"构造一棵二叉树，写出二叉树的层次遍历算法。要求:

(1) 由后序序列 post[0…n-1]和中序序列 in[0…n-1]构造一棵二叉树。

(2) 实现层次遍历算法所使用的队列选用 STL 中的 queue。

```
template <class T>
struct BTNode
{
    T data;
    BTNode* lchild;
    BTNode* rchild;
};

template <class T>
class BTree
{
    BTNode<T>* root;                    //根节点
public:
    /*由后序序列post[0…n-1]和中序序列in[0…n-1]构造一棵二叉树*/
    BTree(T* post,T* in, int n);        //构造二叉树
    ~BTree()                            //析构二叉树
```

```
{
    DestoryBT(root);
}
void DestoryBT(BTNode<T>* b)   //递归销毁二叉树
{
    if (b != nullptr)
    {
        DestoryBT(b->lchild);
        DestoryBT(b->rchild);
        //cout << b->data << " ";
        delete b;
    }
}
void preOrderTraverse()    //先序遍历二叉树
{
    preOrderTraverse(root);
}
void preOrderTraverse(BTNode<T>* b)//递归先序遍历二叉树
{
    if (b != nullptr)
    {
        cout << b->data << " ";
        preOrderTraverse(b->lchild);
        preOrderTraverse(b->rchild);
    }
}
void inOrderTraverse()//中序遍历二叉树
{
    inOrderTraverse(root);
}
void inOrderTraverse(BTNode<T>* b) //递归中序遍历二叉树
{
    if (b != nullptr)
    {
        inOrderTraverse(b->lchild);
        cout << b->data << " ";
        inOrderTraverse(b->rchild);
    }
}
void postOrderTraverse() //后序遍历二叉树
{
    postOrderTraverse(root);
}
void postOrderTraverse(BTNode<T>* b) //递归后序遍历二叉树
```

```
    {
        if (b != nullptr)
        {
            postOrderTraverse(b->lchild);
            postOrderTraverse(b->rchild);
            cout << b->data << " ";
        }
    }
    void levelOrderTraverse();          //层次遍历二叉树
};

int main()
{
    char in[] = "DGBAECF", post[] = "GDBEFCA";
    BTree<char> t(post, in, 7);
    cout << "preOrderTraverse: "; t.preOrderTraverse(); cout << endl;
    cout << "inOrderTraverse: "; t.inOrderTraverse(); cout << endl;
    cout << "postOrderTraverse: "; t.postOrderTraverse(); cout << endl;
    cout << "levelOrderTraverse: "; t.levelOrderTraverse(); cout << endl;
}
```