//5.2.6节 算法 GetValue

```
//利用辅助堆栈S求表达式的值，t为表达式对应二叉树根指针,value为求得的表达式值
void ExprTree:: GetValue(BinTreeNode<string> *t, int &value )
{
    if ( t==NULL)       return;
    AStack<int> caculator;
    AStack<AssBinTreeNode<string>*> s;
    AssBinTreeNode<string>* ass,*lass,*rass;
    ass=new AssBinTreeNode<string>();
    ass->SetPtr(t);      ass->SetFlag(0);
    s.Push(ass) ;
    int i=0,operand1=0,operand2=0,result=0;
    while (!s.IsEmpty())
    {
        s.Pop(ass);
        t=ass->GetPtr();      i=ass->GetFlag();
        if (i==0)
        {
            ass->SetFlag(1);
            s.Push( ass) ;
            if((t->GetLeft()) !=NULL)
            {
                lass=new AssBinTreeNode<string>(t->GetLeft() ,0);
                lass->SetFlag(0);
                s.Push(lass) ;
            }
        }
        if (i==1)
        {
            ass->SetFlag(2);
            s.Push( ass) ;
            if((t->GetRight())!=NULL)
            {
            rass=new AssBinTreeNode<string>(t->GetRight(),0);
            rass->SetFlag(0);
            s.Push( rass) ;
            }
        }
        if (i==2)
        {
           if(t->GetLeft() ==NULL && t->GetRight()==NULL)
           {
                int a=0;
                const char *strin ;
                strin=(t->GetData()).c_str() ;
                a=atoi(strin);
                caculator.Push( a ) ;
           }
           else
           {
                caculator.Pop(operand2);
                caculator.Pop(operand1);
                string op;
                op=t->GetData();
                int result;
                if (op==    "+" ) { result = operand1 + operand2 ;}
                if (op==    "-" ) { result = operand1 - operand2 ;}
                if (op==    "*" ) { result = operand1 * operand2 ;}
```

```
                    if (op==    "/" )
                    {
                        if(operand2 == 0)
                        {
                                cerr << " Divided by 0 ! " << endl;
                                s.clear();    exit(1);
                                }else
                                        result = operand1 / operand2 ;
                    }
                    caculator.Push(result);
                }
            }
        }
        caculator.Pop(value);
}
```