//5.4 节 哈夫曼树类 HuffmanTree 的定义

```
/*Huffman 树结点类 TreeNode 声明*/
template<class T>
class HuffmanNode
{
private:
    HuffmanNode<T> *LLINK ;          // 指向左、右孩子的指针
    HuffmanNode<T> *RLINK ;
    T INFO ;                         //数据域
    int weight;                      //权值
public:
    HuffmanNode() { }              // 构造函数
    HuffmanNode<T> * GetLeft(void)const { return LLINK ; }
    void SetLeft(HuffmanNode<T> *L){ LLINK = L ; }
    HuffmanNode<T> * GetRight(void)const { return RLINK ; }
    void SetRight(HuffmanNode<T> *R){ RLINK = R ;}
    T& GetData() {    return    INFO;   }
    void SetData(const T & item){ INFO = item ; }
    int GetWeight() {    return    weight;   }                    // 返回 weight 域
    void SetWeight(const int w){ weight = w ; }            // 将 weight 域更新为 w
} ;
/*Huffman 树类 HuffmanTree 声明*/
template <class T>
class HuffmanTree
{
private:
    int m; //Huffman 树结点个数
    HuffmanNode<T>   * * H ;// Huffman 树结点的数组

public:
    HuffmanTree(){ m=0; H = NULL; }   // 构造函数
    void CreatHuffanTree(T data[],int weight[],int n ) ; //创建 Huffman 树
    //其他操作
    HuffmanNode<T> *GetRoot() { return m==0 ? NULL : H[m] ; }
    void PreOrder(HuffmanNode<T> *t) const ;
    void InOrder(HuffmanNode<T> *t) const ;
};
```

//5.4.3 节 算法 Huffman

```
//创建 Huffman 树
template <class T>
void HuffmanTree<T>::CreatHuffanTree(T data[],int weight[],int n )
{
    m = n;
    H = new HuffmanNode<T>   * [m+1];
    HuffmanNode<T> *p1,*p2,*p,*t;
    int i, j;
    for (i=1 ;i<= m ;i++ ) //初始化
    {
        H[i]=new HuffmanNode<T>();
        H[i]->SetData(data[i]);
        H[i]->SetWeight(weight[i]);
          H[i]->SetLeft(NULL);
          H[i]->SetRight(NULL);
    }
    for (i=1 ;i< m ;i++)          //组合过程
    {
        t =new HuffmanNode<T>();
```

```
            p1= H[i];
            p2= H[i+1];
            t->SetWeight(p1->GetWeight() + p2->GetWeight());
            t->SetLeft(p1);
            t->SetRight(p2);
            p=t;
            //把新组合结点 t 的地址 p 插入到数组 H 中，使得 Weight(H[i+1]) ≤Weight(H[m])
            j = i + 2;
            while( j <= m && ( p->GetWeight()) > (H[j]->GetWeight()) )
            {
                  H[j-1] = H[j];
                  j = j+1;
            }
            H[j-1] = p;
      }
};
```