```
//////////////////////////////////////////////////////////////////////////////
//////////////////////////
//   静态链表及其结点的类定义
//   Author：Melissa M.CAO
//   Belong：Section of software theory，School of Computer Engineering & Science，
Shanghai University
//   Version： 1.0
//////////////////////////////////////////////////////////////////////////////
//////////////////////////

#pragma once

template<class type>class StaticLinkList;

//////////////////////////////////////////////////////////////////////////////
//////////////////////////
//   静态链表结点的类定义
//////////////////////////////////////////////////////////////////////////////
//////////////////////////
template<class type> class StaticNode
{
     friend class StaticLinkList<type>;

     private:
         type data;
         int next;

     public:
         StaticNode(int pnext = -1);
         StaticNode(const type &item,int pnext = -1);
         void SetNext(int n) { next = n; }
         void SetData(type x) {  data = x; }
         ~StaticNode( ){}
};


//////////////////////////////////////////////////////////////////////////////
//////////////////////////
//   静态链表的类定义
//////////////////////////////////////////////////////////////////////////////
//////////////////////////
template<class type>class StaticLinkList
{
private:
```

```
    static const int maxlen = 40;
    int maxsize;

public:
    int head; //已用空间
    int avail; //可用空间
    StaticNode<type> *info;

    StaticLinkList(int size = maxlen);
    ~StaticLinkList(void) { if (info) delete[]info;}

    int GetLength( ) const;           //获取表长度
    int Locate(type &x);               //定位
    bool IsEmpty( ) const {return info[head].next == avail;}            //判断链表是否为空
    void Clear( );                    //清除所有结点
    void PrintList();                 //输出链表所有内容
    bool EndofList (int now) const {return now == avail;}    //判断是否达到链表的最后
    void Append(const type &x);       //追加元素到链尾
          ////////////////////////////////补充
   void Insert(const type & x, const type &y, char flag);  //在 y 之前或之后插入 x
   void Insert(const type & x, const int i, char flag);   // 在第 i 个元素之前或之后插入 x
   type Remove(const int i);                          //删除第 i 个结点
   int GetLast();                   //获取最后一个元素的下标
};
```