```
////////////////////////////////////////////////////////////////////////////////
//////////////////////////
//   线性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 LinkList;
template<class type>class LinkQueue;
template<class type>class LinkSortList;
class polynomial;
template <class type> class Sparsematrix;
template<class type>class CirList;

////////////////////////////////////////////////////////////////////////////////
//////////////////////////
//   线性链表结点的类定义
////////////////////////////////////////////////////////////////////////////////
//////////////////////////
template<class type>class node
{
     friend class LinkList<type>;       //单链表
     friend class LinkQueue<type>;      //链式队列
     friend class LinkSortList<type>;   //排序链表
     friend class Sparsematrix<type>;   //稀疏矩阵十字链表
     friend class CirList<type>;        //单循环链表

     friend void  ExtractOdd(LinkList<int>& L, LinkList<int>& LA );

     private:
         node<type>* next;
     public:
         type data;
         node(node<type>* pnext = NULL);
         node(const type &item,node<type>* pnext = NULL);
         void SetNext(node<type>* p) { next = p; }
         void SetData(type x) {  data = x; }
         type GetData() {  return data; }
         ~node( ){}
```

```
};
//////////////////////////////////////////////////////////////////////////////
//////////////////////////
//   线性链表的类定义
//////////////////////////////////////////////////////////////////////////////
//////////////////////////
template<class type> class LinkList
{
    friend class LinkSortList<type>;    //排序链表
    friend class polynomial;            //一元多项式
    friend class Sparsematrix<type>;    //稀疏矩阵十字链表

    friend void  ExtractOdd(LinkList<int>& L, LinkList<int>& LA );

    private:
          node<type> *head;             //头结点
          node<type> *pcurrent;
       public:
          LinkList( );//一般构造函数
          LinkList(type a[], int n);    //为调试程序，增加通过数组构造链表的构造
函数
          ~LinkList( );
          int GetLength( ) const;           //获取表长度
          type GetCurrent( ) const;         //获取当前结点
          node<type>* Locate(type &x);      //定位
          void InsertBefor(const type &x);  //当前结点之前插入
          void InsertAfter(const type &x);  //当前结点之后插入
          type DeleteCurrent( );            //删除当前结点
          int IsEmpty( ) const;             //判断链表是否为空
          void Clear( );                    //清除所有结点
          node<type>* ResetCurrent(int i);  //将当前结点设置为位置 i
          node<type>* CurrentToNext( );     //将当前结点设置为下一个
          void PrintList();                 //输出链表所有内容
          node<type>* Reset(int i);         //重置第 i 项
          node<type>* Next();               //返回下一个结点
          int EndofList () const;           //判断是否达到链表的最后
          void Append(const type &x);       //追加元素到链尾
          void InsertAsFirst(const type &x);//插入元素作为表中的第一个元素
          ////////////////////////////////补充
          void InsertAfter (const type & x, const type &y);  //在 x 前插入 y
          void InsertAfter (const type & x, const int i);   // 在第 i 个元素后插入
y
          type Remove(const int i);                          //删除第 i 个结点
          int Reverse ( );                                   // 逆置
```

```
        //////----------------------------------------------------------
        //以下部分为习题
        void OrderMerge(int t, LinkList<type> &b); //合并 b 表到自身，参数 t 为合并类型，1--同序，2--反序。13 题
};

//#include "LinkList.cpp"
```