**顺序存储的线性表基本操作**

```
// 顺序表的构造函数
template<class T>
LinearList<T>::LinearList(int MaxListSize=10)
{
    MaxSize=MaxListSize;
    element=new T[MaxSize];                          // 申请规模为 MaxSize 的数组空间
    length=0;                                        // 初始时没有真正表结点，故表长度为 0
};
//存取：将下标为 k 的结点的字段值赋给 item 并返回 true，若不存在则返回 false
template<class T>
bool LinearList <T> :: Find ( int k, T & item ) const {
    if ( k <0 || k >length-1 || length = = 0)
        {cout<< "unreasonable position or empty list!"<<endl; return false ; }// 下标为 k 的结点不存在
    item = element [k] ;
    return true ;
}
//查找：在表中查找字段值为 item 的结点并返回其下标；若表中没有 item，则返回-1
template<class T>
intLinearList <T> :: Search ( const T & item ) const {
        for ( i = 0 ; i < length ; i ++ )
            if (element[i] = = item ) return i ;
        return-1 ;
}
// 删除表中下标为 k 的结点并将其值赋给 item
template<class T>
voidLinearList <T> :: Delete ( int k, T & item ) {
        if ( find ( k, item ) ) { // 若找到下标为 k 的结点，则将其后面所有结点均向前移动一个位置
            for ( int i = k+1 ; i < length ; i ++ )
                element [i-1] = element[i] ;
            length -- ; // 表长度相应减 1
            return ;
        }
}
// 在下标为 k 的结点后插入 item
template<class T>
void LinearList <T> :: Insert ( int k, const T & item ) {
        if ( IsFull () ) { cout <<"The list is full ! "<< endl ; exit (1) ; } // 若表已满无法插入新结点
    // 若下标为 k 的结点不存在
    if ( k<0 || k > length-1 ) { cout <<"The node does not exist !"<< endl ; exit (1) ; }
        // 从后向前将下标大于等于 k+1 的结点均向后移动一个位置
    for ( i = length-1 ; i >= k+1 ; i - - )
        element[i+1] = element[i] ;
    element[k+1] = item ;
    length++ ;
    return ;
}
```