```
/* 归并排序 - 递归实现 */

/* L = 左边起始位置, R = 右边起始位置, RightEnd = 右边终点位置*/
void Merge( ElementType A[], ElementType TmpA[], int L, int R, int RightEnd )
{ /* 将有序的A[L]~A[R-1]和A[R]~A[RightEnd]归并成一个有序序列 */
     int LeftEnd, NumElements, Tmp;
     int i;

     LeftEnd = R - 1; /* 左边终点位置 */
     Tmp = L;         /* 有序序列的起始位置 */
     NumElements = RightEnd - L + 1;

     while( L <= LeftEnd && R <= RightEnd ) {
         if ( A[L] <= A[R] )
             TmpA[Tmp++] = A[L++]; /* 将左边元素复制到TmpA */
         else
             TmpA[Tmp++] = A[R++]; /* 将右边元素复制到TmpA */
     }

     while( L <= LeftEnd )
         TmpA[Tmp++] = A[L++]; /* 直接复制左边剩下的 */
     while( R <= RightEnd )
         TmpA[Tmp++] = A[R++]; /* 直接复制右边剩下的 */

     for( i = 0; i < NumElements; i++, RightEnd -- )
         A[RightEnd] = TmpA[RightEnd]; /* 将有序的TmpA[]复制回A[] */
}

void Msort( ElementType A[], ElementType TmpA[], int L, int RightEnd )
{ /* 核心递归排序函数 */
     int Center;

     if ( L < RightEnd ) {
          Center = (L+RightEnd) / 2;
          Msort( A, TmpA, L, Center );               /* 递归解决左边 */
          Msort( A, TmpA, Center+1, RightEnd );      /* 递归解决右边 */
          Merge( A, TmpA, L, Center+1, RightEnd );  /* 合并两段有序序列 */
     }
}

void MergeSort( ElementType A[], int N )
{ /* 归并排序 */
     ElementType *TmpA;
     TmpA = (ElementType *)malloc(N*sizeof(ElementType));

     if ( TmpA != NULL ) {
          Msort( A, TmpA, 0, N-1 );
          free( TmpA );
     }
     else printf( "空间不足" );
}
```