連結リストによるリストの実現

```
typedef  struct {
                elementtype     element ;
                CELL↑           next ;
                                /*  next は CELL 型データへのポインタ  */
        } CELL ;


typedef  struct {
                CELL↑           head ;
                CELL↑           last ;
                                /*  head, last は CELL 型データへのポインタ  */
        } LIST ;


typedef     CELL↑   position ;      /*  position は CELL 型データへのポインタ  */
```

以下のプログラムにおいて, パラメタのリスト *L* は, C 言語では *L* へのポインタを渡すことになる.
例えば, *Tail* は C 言語では

```
position  Tail( LIST＊  ptr_L )
{
        return  (＊ptr_L).last ;
}
```

と書き, これを呼び出す側では,

```
i := Tail( &L ) ;
```

とする. このことは 1.4 節にも書いたので, 今後, この説明は省略する.

```
position  Tail( L )
        入力 ： L  :     LIST
        出力 ： Tail  :  position
{
        return  L.last ;
} /*  Tail  */
```

Makenull(*L*), *Insert*(*x, p, L*), *Delete*(*p, L*), *Previous*(*p, L*) は本文中にあるので省略.

```
position  First( L )
        入力 :  L  :     LIST
        出力 :  First  :  position
{
        return  L.head ;
} /*  First  */
```

```
position  Next( p, L )
        入力 :  p  :     position (値呼び)
                L  :     LIST
        出力 :  Next  :  position
{
        if ( p ≠ NULL )          return  p.next ;
        else      エラーを出力する ;
} /*  Next  */
```

```
position  Locate( x, L )
        入力 :  x  :     elementtype
                L  :     LIST
        出力 :  Locate  :  position
/*  ここでは, elementtype の二つの要素 x, y が同一か否かを判定する関数 Same( x, y )  */
/*  が存在するものとし, それを用いている  */
position  q ;                    /*  要素を順に調べるためのポインタ. 局所的変数  */
{
        q  :=  L.head ;
        while (  (↑q).next  ≠  NULL  ) {
                if (  Same( x, Retrieve( q, L ) ) = true  )      return  q ;
                q  :=  (↑q).next ;
        }
        return  q ;
} /*  Locate  */
```

```
elementtype  Retrieve( p, L )
        入力 :  p  :     position (値呼び)
                L  :     LIST
        出力 :  Retrieve  :  elementtype
/*  ここでは, elementtype の要素を関数の戻し値とできるものとしている  */
{
        if (  (↑p).next  ≠  NULL  )      return  (↑(↑p).next).element  ;
        else      エラーを出力する ;
} /*  Retrieve  */
```

```
void  PrintList( L )
        入力 ： L  :     LIST
        出力 ： L の要素を順に印刷したもの
/* ここでは, elementtype の要素 x を出力する関数 PrintElement( x ) が存在する */
/* ものとし, それを用いている */
position q ;                    /* 要素を順に調べるためのポインタ. 局所的変数 */
{
        q := L.head ;
        while ( (↑q).next ≠ NULL ) {
                PrintElement ( Retrieve( q, L ) ) ;
                q := (↑q).next ;
        }
} /* PrintList */
```