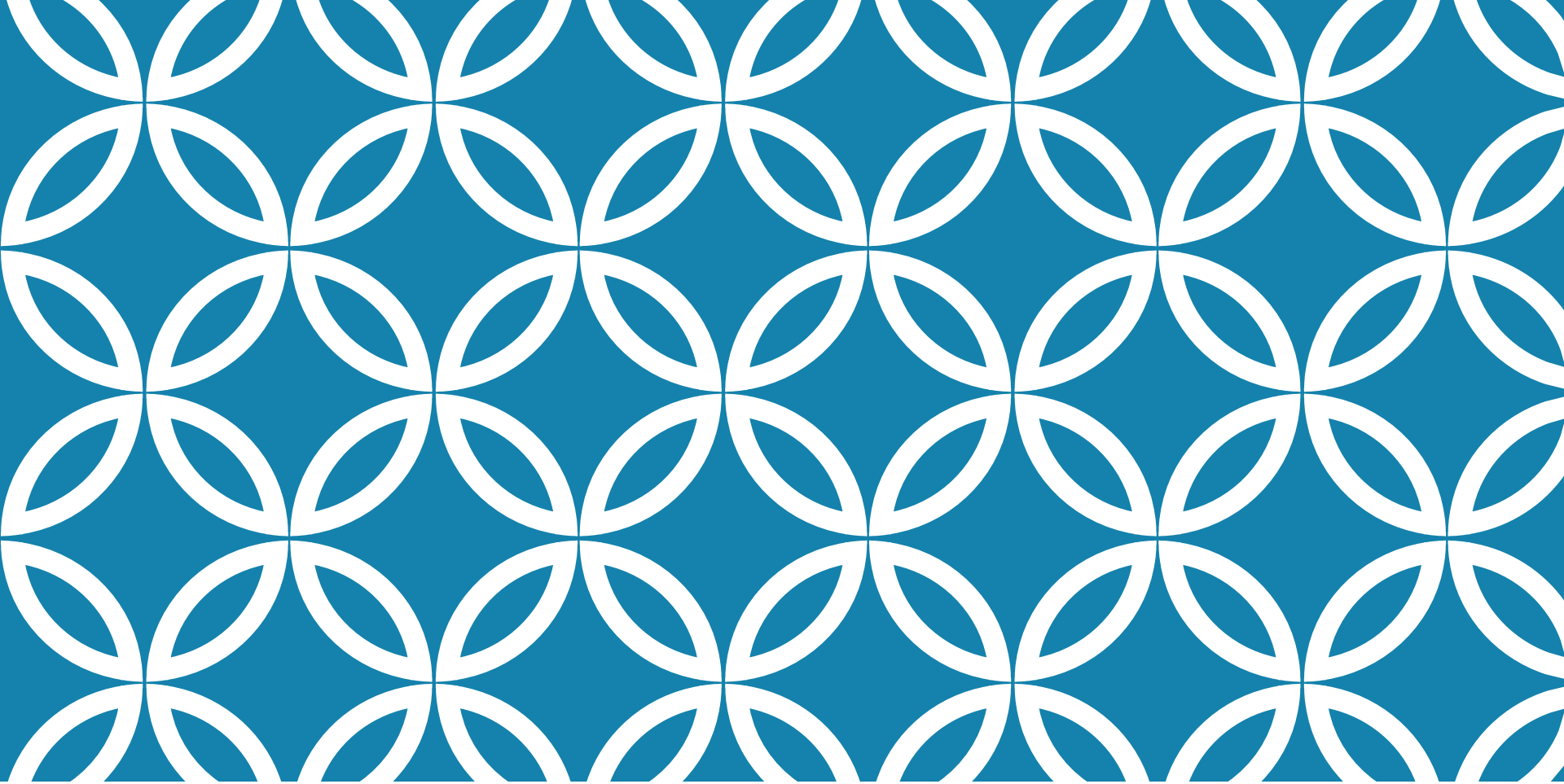
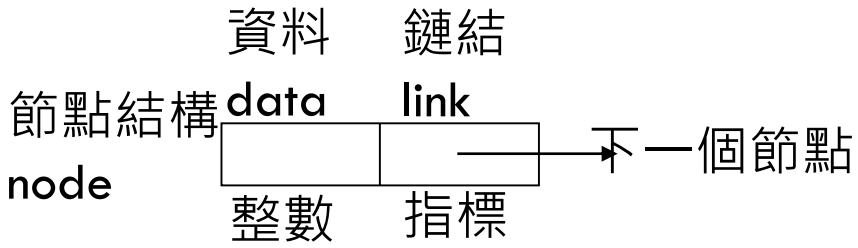


鏈結串列的製作

資料結構
鍾宜玲

定義節點結構型態

假設節點如圖所示：



則節點結構定義如下：

```
struct node{
        int data;
   struct node *link;
};
```

定義節點結構

另外可用 typedef 將 struct node 改為新型態名稱NODE：

```
typedef struct node NODE;
```

或可將typedef與節點的定義合併一起寫：

```
typedef struct node{
   int data;
   struct node *link;
} NODE;
```

宣告開頭指標list 的程式如下：

```
NODE  *list;
```

產生新節點之函數NEW( )

```
NODE  *new()                              /*傳回值為指向型態NODE之指標*/
{
 NODE *p;
 p=(NODE  *)malloc(sizeof(NODE));  /*動態分配記憶體*/
 if (p == NULL){                          /*若記憶體不足，則分配失敗*/
     printf("記憶體不足" );
     exit(1);
 }
 return(p);                               /*傳回指向新節點的指標*/
}
```

建立鏈結串列

假設節點結構如：

```
typedef struct node{
   int data;
   struct node *link;
   } NODE;
```

則建立鏈結串列之函數create() 可寫為：

鏈結串列之函數CREATE()

```
NODE *create()          /*傳回值為指向型態NODE之指標*/
{
 NODE *list, *p, *q; /*指標list為串列之開頭指標*/
 int i, n, d;

 list=new();              /*建立一個只含首節點的空鏈結串列*/
 list->link=NULL;
 p=list;                  /*指標p指向尾端節點*/
 printf("輸入資料筆數：" );
 scanf("%d", &n);
```

```
for(i=1; i<=n; i++) {          /*利用迴圈將節點加入串列尾端*/
    printf("輸入第%d筆資料：" , i);
    scanf("%d", &d);
    q=new();                   /*分配新節點空間給指標q*/
    q->data=d;                 /*儲存資料值*/
    q->link=NULL;
    p->link=q;                 /*新節點q加入於尾端節點p之後*/
    p=q;                       /*指標p改指向新的尾端節點q*/
 }
 return(list);                 /*傳回建立好的鏈結串列開頭指標*/
}
```

CREATE() 函數解說

1. 建立一個只含首節點的空鏈結串列

```
list = new();
list -> link = NULL;
```

2. 指標p表串列之尾端節點

```
p = list;
```

3. 製作一個新節點q

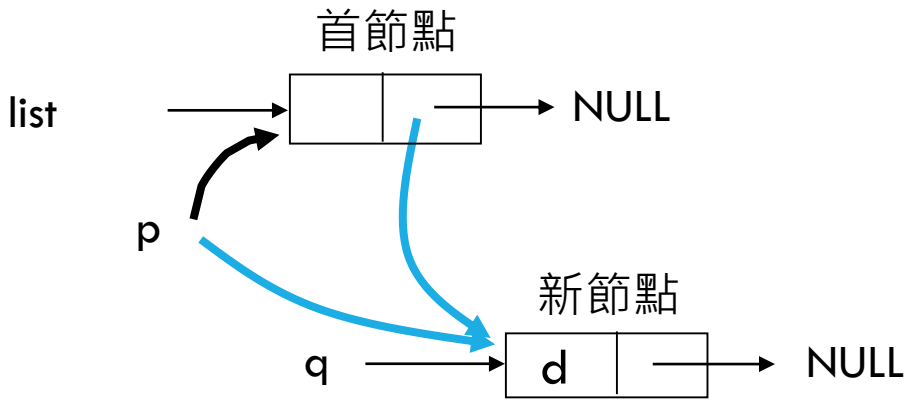
```
q = new();
q -> datd = d;
q -> link = NULL;
```

4. 節點p之鏈結欄改向新節點q

```
p -> link = q;
```

5. 將指標p改向尾端節點q

```
p = q;
```

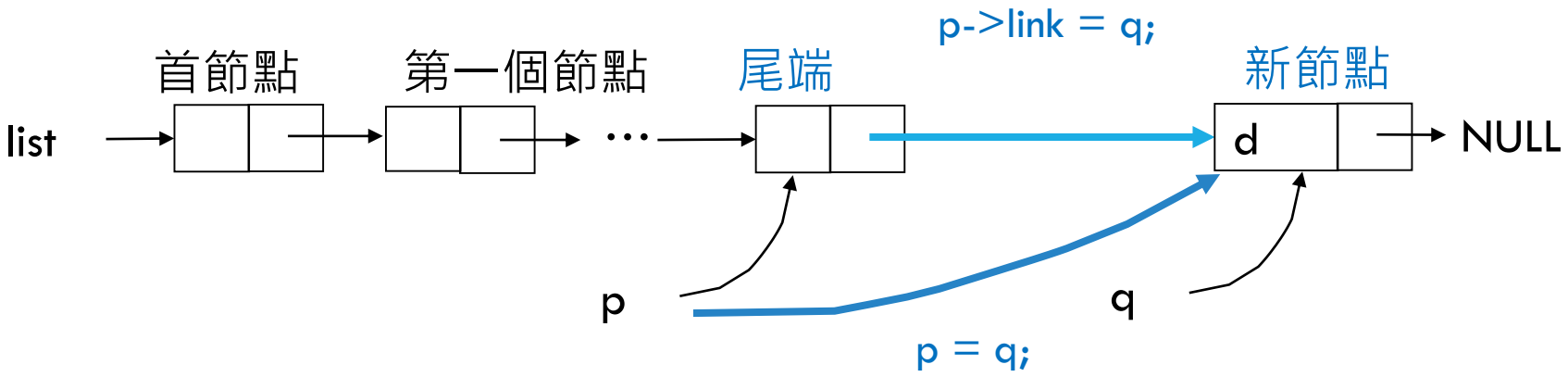


CREATE() 函數解說(續)

加入第 i 個節點

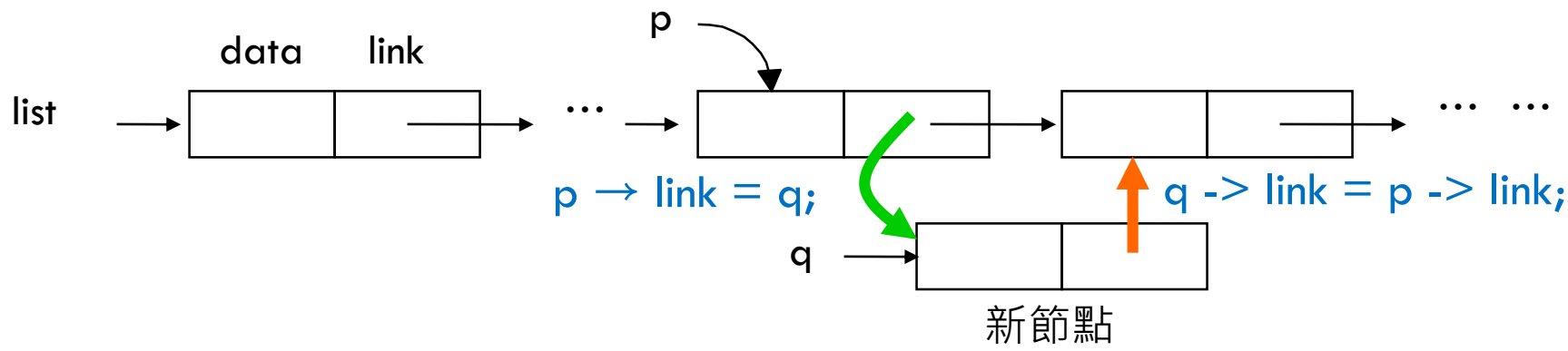
(a)先將尾端節點p之鏈結欄改向新節點q

(b)再將指標p改向新的尾端節點q



插入新節點

則欲插入一新節點 q 於串列 list 中節點 p 之後，改變的過程如下：

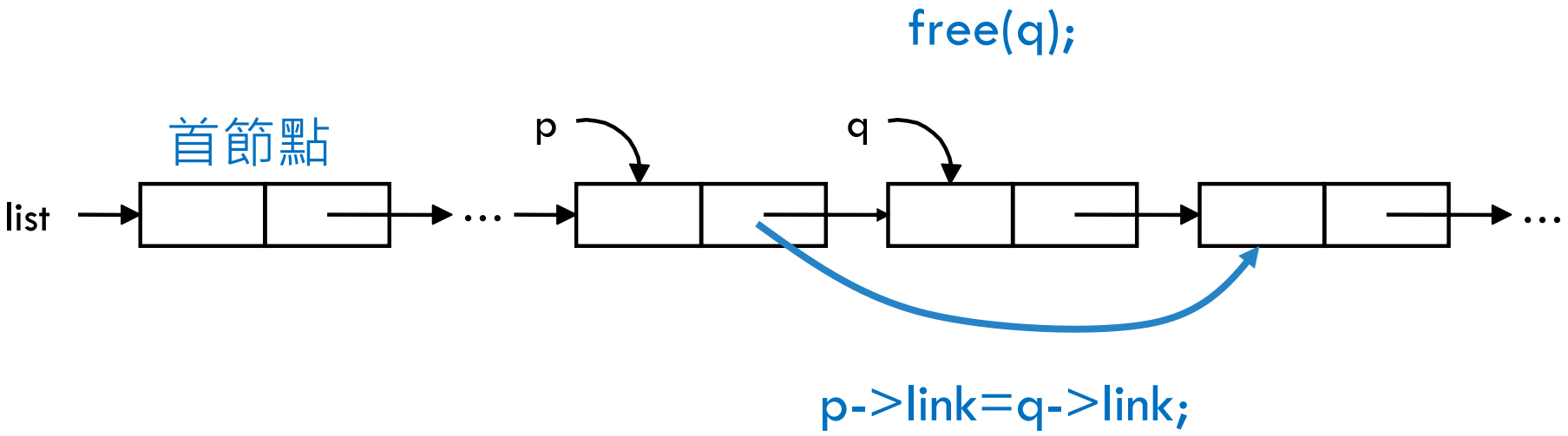


插入新節點的函數INSERT()

```
/*加入新節點q於節點p之後*/
void insert(NODE  *p, NODE  *q)
{
   q->link = p->link; /*節點q的鏈結欄指向p的下一節點*/
   p->link = q;       /*節點q的鏈結欄指向q*/
}
```

刪除節點

假如串列list包含首節點，則欲刪除串列list中節點p的下一個節點q，則刪除的過程如下：



刪除節點的函數DELETE()

```
/*刪除節點p之後的節點q*/
void delete ( NODE  *p, NODE  *q )
{
  p->link = q->link;
  free(q);
}
```

尋找節點的函數SEARCH()

```
NODE *search (NODE  *list,  int  d)
{
 NODE  *p;
 p = list->link;              /*指標p指向串列的第一個節點*/

/*若指標p不為NULL且所指節點之資料不為d*/
  while( p != NULL && p->data != d)
      p = p->link;                /*則p改指向下一個節點*/
 return(p);
}
```

計算串列長度的函數LENGTH( )

```
int length( NODE  *list )
{
  NODE  *p;
  int counter=0;          /*計數變數初值為0*/
  p=list->link;           /*指標p指向串列中的第一個節點*/
  while( p!=NULL ){       /*若指標p不為NULL，則繼續計數*/
       counter ++;        /*counter值加1*/
       p=p->link;         /*指標p指向下一個節點*/
  }
  return(counter);        /*傳回計數值*/
}
```