鏈結串列自編教材（六）

- 使用 dummy node 的環狀雙向鏈結串列（大量簡化處理程序）：

```
struct ANODE{
    int val;
    struct ANODE *next,*prev;
};
```

環狀雙向鏈結串列

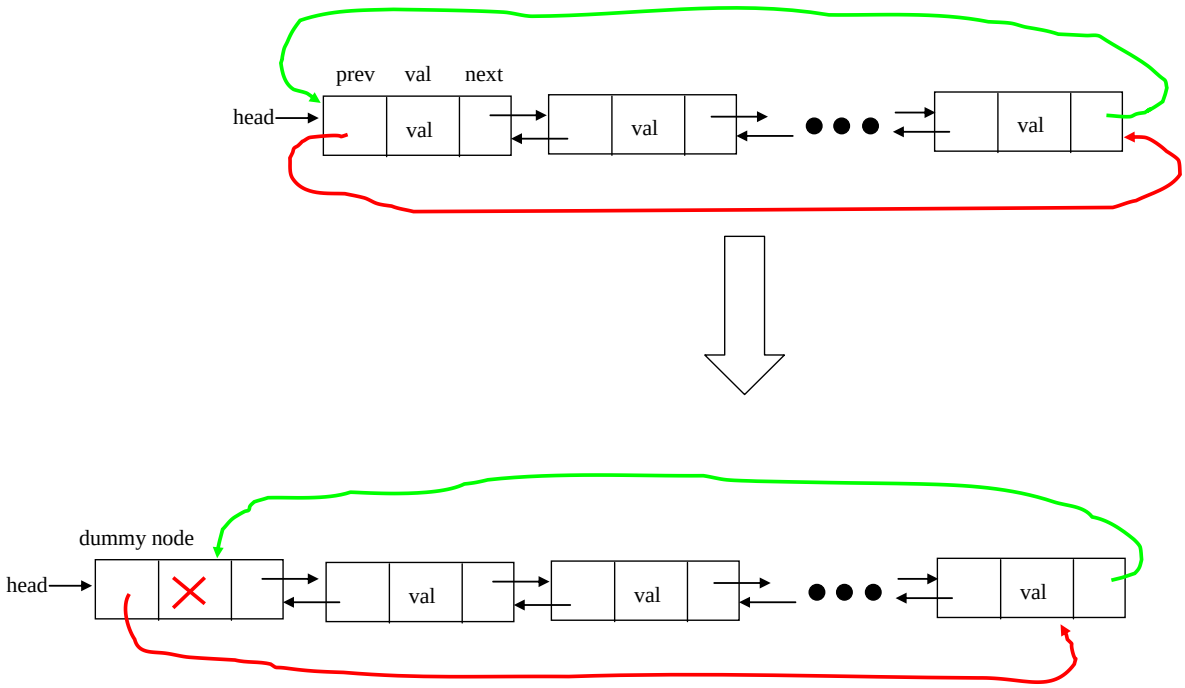

圖 6-1

- **以環狀單向鏈結串列為基礎**，每次以亂數產生一[0,1000]之整數值，若該值>100，則以同方式繼續產生下一亂數值，若該值<=100，則停止此產生值之程序。所有產生之值必須全部記錄在使用 **dummy node 的環狀雙向鏈結串列**中（均在串列的最開頭新增一 node），並可重複列印所有產生的值。

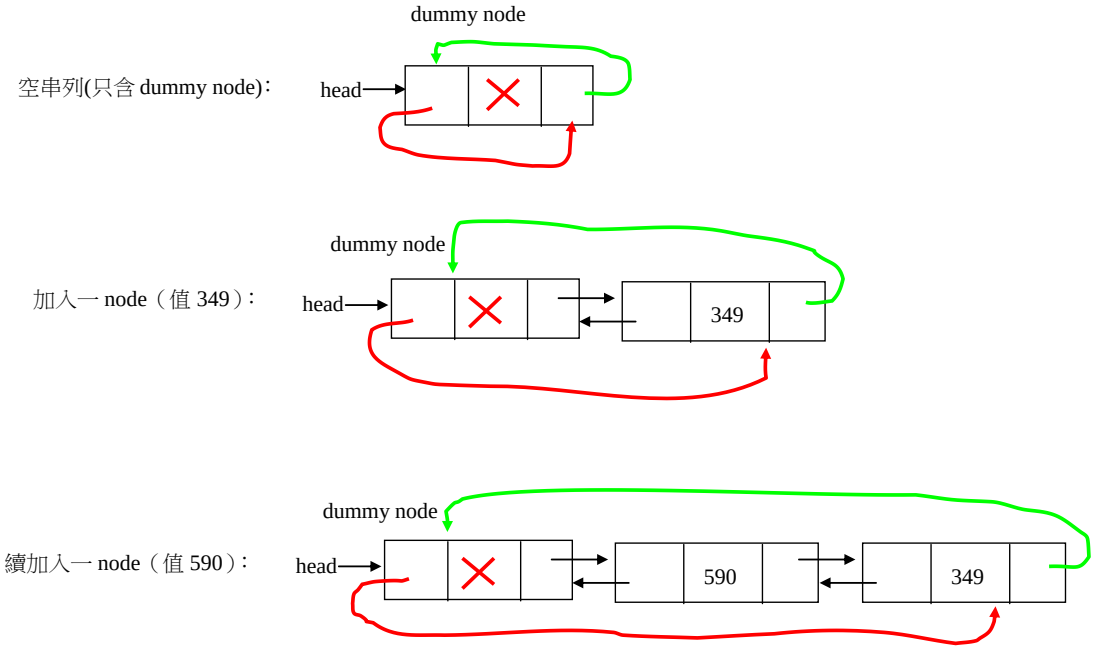

圖 6-2

- 串列前端加入 1 個 node

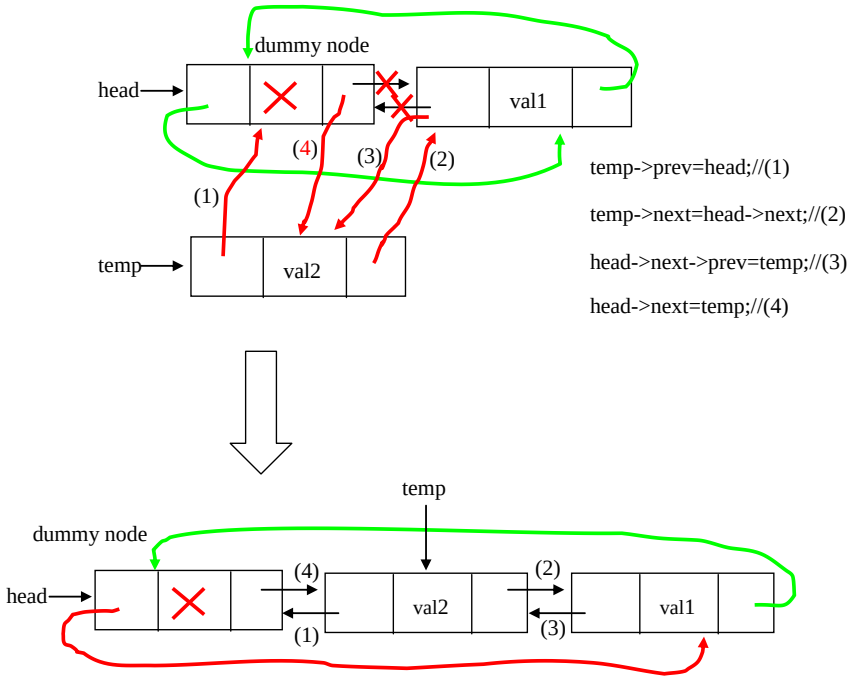

圖 6-3

- 空串列前端加入 1 個 node 亦適用

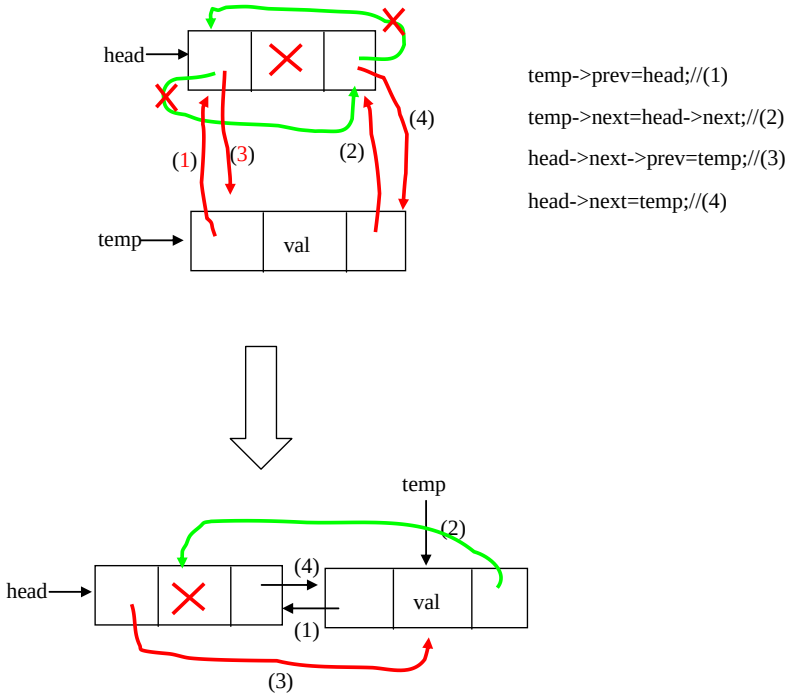

圖 6-4

- 列印環狀雙向鏈結串列（有 dummy node）的內容：順向印

```
void printlistforward(){
struct ANODE * ptr;

   ptr=head->next;
   while (ptr!=head){
         printf("%d\n",ptr->val);
         ptr=ptr->next;
   }
}
```

- 列印環狀雙向鏈結串列（有 dummy node）的內容：反向印

```
void printlistbackward(){
struct ANODE * ptr;

   ptr=head->prev;
   while (ptr!=head){
         printf("%d\n",ptr->val);
         ptr=ptr->prev;
   }
}
```

- 列印環狀雙向鏈結串列（有 dummy node）的內容，dir=1 為順向列印，否則為反向列印

```
void printlist(int dir){
struct ANODE * ptr;

   if (dir==1) ptr=head->next;
   else ptr=head->prev;
   while (ptr!=head){
         printf("%d\n",ptr->val);
         if (dir==1) ptr=ptr->next;
         else ptr=ptr->prev;
   }
}
```

- 每次以亂數產生一[0,1000]之整數值，若該值>100，則以同方式繼續產生下一亂數值，若該值<=100，則停止此產生值之程序。所有產生之值必須全部記錄在使用 **dummy node 的環狀雙向鏈結串列**中(均在串列的最開頭新增一node)，並可重複列印所有產生的值。(s3-18.c)

```
//產生使用 dummy node 的空串列
head=(struct ANODE *)malloc(sizeof(struct ANODE));
head->prev=head;
head->next=head;

do{
   //產生一個新的 node
   temp=generateanode();

   //產生的 node 加入串列前端
   temp->prev=head;//(1)
   temp->next=head->next;//(2)
   head->next->prev=temp;//(3)
   head->next=temp;//(4)


} while (temp->val>100);

printlist(1);
printlist(2);
```

- 將環狀雙向鏈結串列第一個 node 所配置之記憶體釋放(亦適用於 list 中只有一個 node 的刪除)

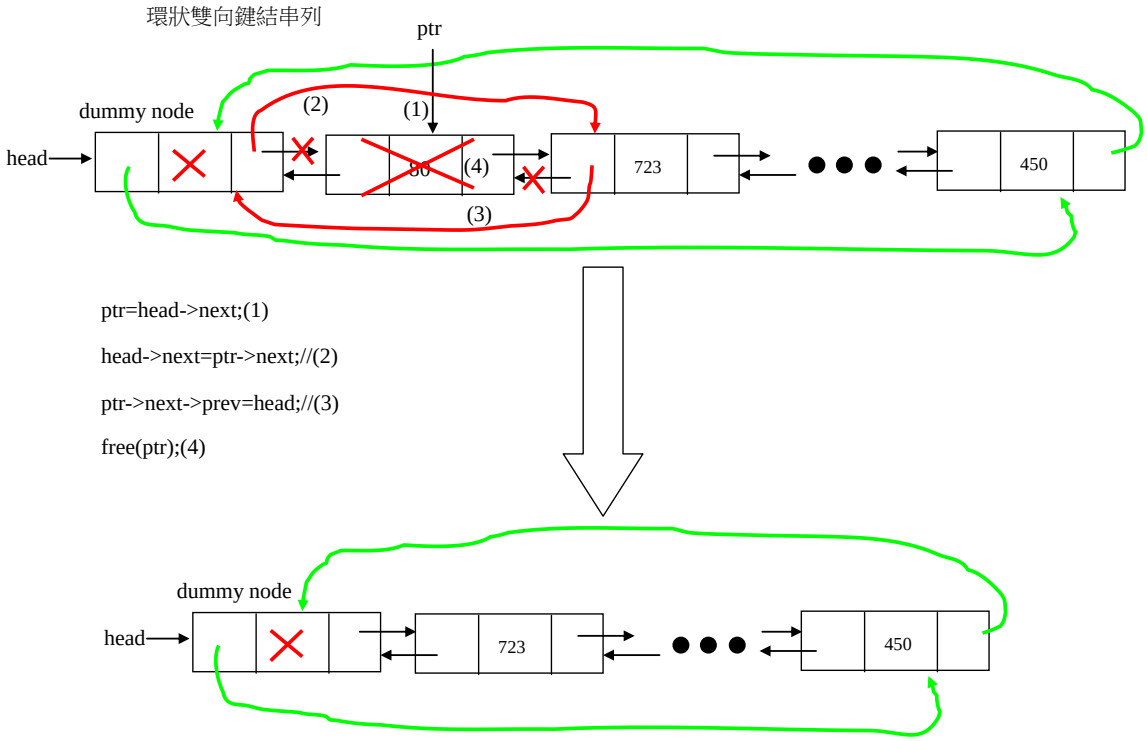

圖 6-5

- 將前所產生的環狀雙向鏈結串列所配置之記憶體完全釋放(s3-19.c)

```
------------------------------------------------------------
while (head->next!=head){//串列尚未為空
      ptr=head->next;//(1)
      head->next=ptr->next;//(2)
      ptr->next->prev=head;//(3)
      free(ptr);//(4)
}
------------------------------------------------------------
```