

Chapter 15
Linked List
(鏈結串列)

Memory Allocation

- Static Allocation
  - Compile-time
  - Fixed size of memory size
  - Example: int arr[9][9]; // allocate 9x9 2D array
- Dynamic Allocation
  - Run-time
  - Efficiently utilize memory

Dynamic Allocation

- Use the standard function malloc()

```
pointer-variable = (data-type *) malloc(int memory-size-in-byte);
```

- Example:
  - Allocate an 1D integer array with size 3

```
int *ptr;
ptr = (int *)malloc(12);
```

Bad coding style

```
int *ptr;
ptr = (int *)malloc(3 * sizeof(int));
```

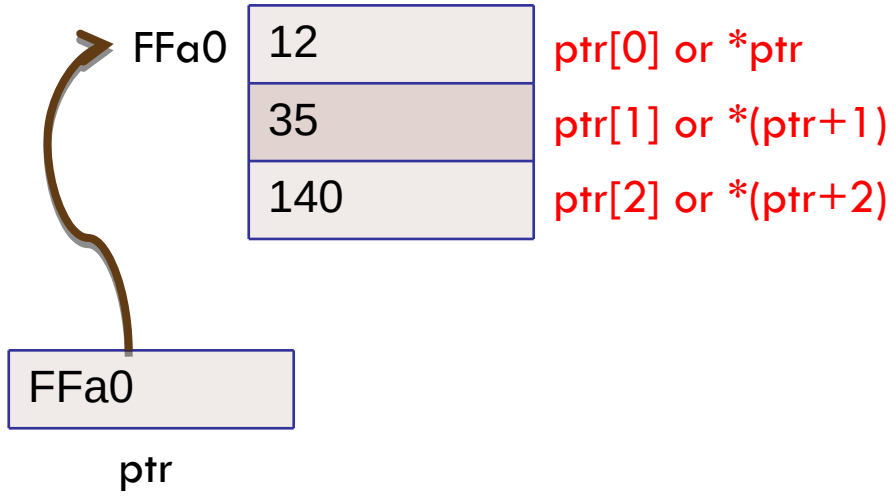
Good coding style

Access Memory

- Access k-th element by *(ptr+k-1) or ptr[k-1]

```
int *ptr;
ptr = (int *)malloc(3 * sizeof(int));
*ptr=12;
*(ptr+1) = 35;
*(ptr+2) = 140;
```

```
int *ptr;
ptr = (int *)malloc(3 * sizeof(int));
ptr[0]=12;
ptr[1] = 35;
ptr[2] = 140;
```



Initialize Memory

- Use the standard function memset()

```
memset(pointer-variable, int value, int memory-size-in-byte);
```

- Initialize all elements to 10

```
int *ptr;
ptr = (int *)malloc(3 * sizeof(int));
memset(ptr, 10, 3 * sizeof(int));
```



Free Memory

- Use the standard function free
- Can not access the pointer after free()

```
free(pointer-variable);
```

```
int *ptr;
ptr = (int *)malloc(3 * sizeof(int));
*ptr=12;
*(ptr+1) = 35;
*(ptr+2) = -15;
free(ptr);
```



Dynamic Allocation Example: Array

```
01  /*  動態記憶體配置的範例 */
02  #include<stdio.h>
03  #include<stdlib.h>
04  int main(void)
05  {
06     int *ptr,i;
07     ptr=(int *) malloc(3*sizeof(int));   /* 配置 3 個存放整數的空間 */
08
09     *ptr=12;              /* 把配置之記憶空間的第 1 個位置設值為 12 */
10     *(ptr+1)=35;          /* 把第 2 個位置設值為 35 */
11     *(ptr+2)=140;         /* 把第 3 個位置設值為 140 */
12
13     for(i=0;i<3;i++)
14        printf("*ptr+%d=%d\n",i,*(ptr+i));   /* 印出存放的值 */
15
16     free(ptr);            /* 釋放由 ptr 所指向的記憶空間 */
17     system("pause");
18     return 0;
19  }
```

```
/* OUTPUT---

*ptr+0=12
*ptr+1=35
*ptr+2=140
----------------------*/
```

Dynamic Allocation Example: Structure

```
01  /* 配置記憶空間給結構變數 */
02  #include<stdio.h>
03  #include<stdlib.h>
04  int main(void)
05  {
06     int num,i;
07     struct student           /* 定義結構 student */
08     {
09        char name[10];
10        int score;
11     } *ptr;                  /* 宣告指向結構 student 的指標 ptr */
12
13     printf("Number of student: ");
14     scanf("%d",&num);
15
16     ptr=(struct student *) malloc(num*sizeof(struct student));
17
```

Dynamic Allocation Example: Structure (Cont.)

```
18      for(i=0;i<num;i++)
19      {
20         fflush(stdin);                    /* 清空緩衝區的內容 */
21         printf("name for student %d: ",i+1);
22         gets((ptr+i)->name);              /* 將鍵入的字串寫入 name 成員 */
23         printf("score for student %d: ",i+1);
24         scanf("%d",&(ptr+i)->score);      /* 將鍵入的整數寫入 score 成員 */
25      }
26      for(i=0;i<num;i++)
27         printf("%s: score=%d\n",(ptr+i)->name,(ptr+i)->score);
28
29      free(ptr);                           /* 釋放記憶空間 */
30
31      system("pause");
32      return 0;
33  }
```

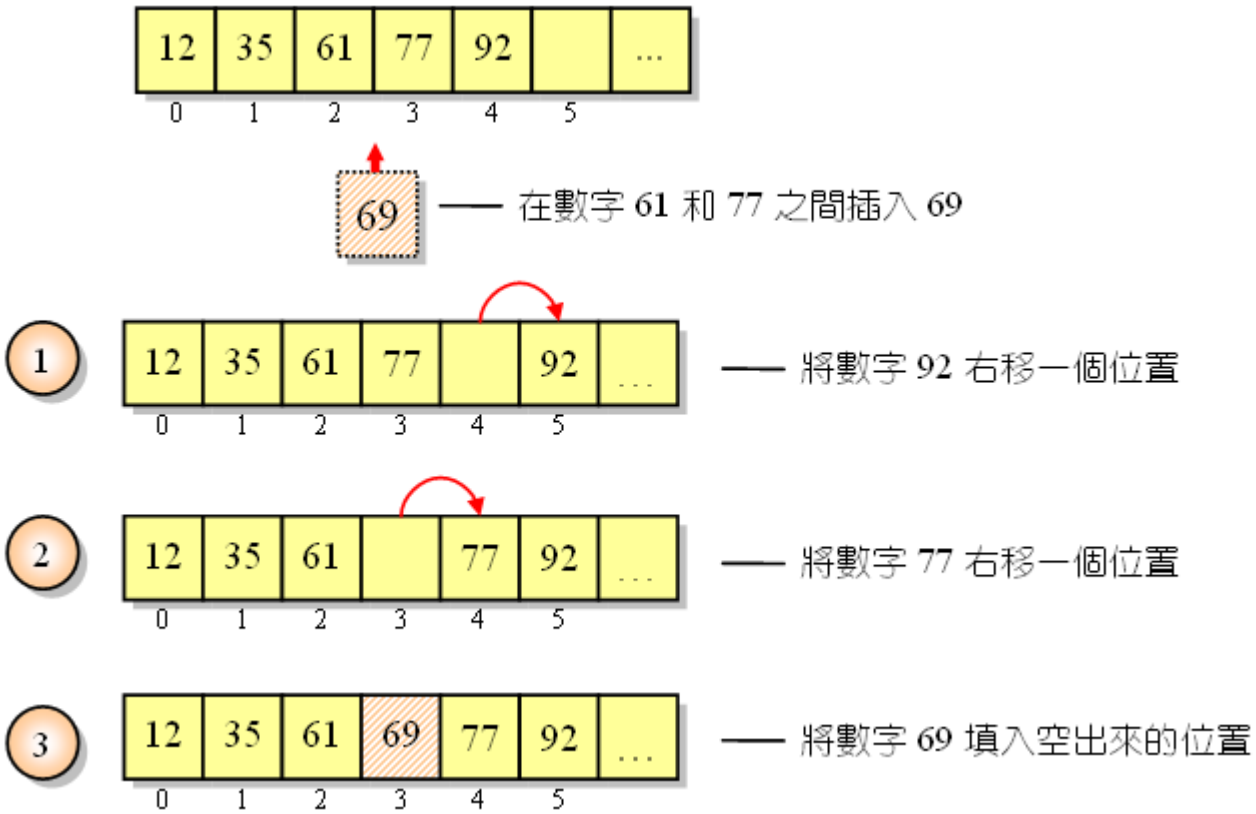
```
/* OUTPUT---------
Number of student: 2
name for student 1: Jenny
score for student 1: 65
name for student 2: Teresa
score for student 2: 88
Jenny: score=65
Teresa: score=88
-----------------------------*/
```



List

- Ordered data could construct a list.
- Two types of lists:
  - Sequential list: Continuous memory address to store the list
    - Advantage: Easy to access
    - Disadvantage:
      - Large overheads on insertion and deletion
      - Memory space shortage or waste problems
  - Linked list: Pointers that link elements of the list together
    - Advantage: Flexible on memory usage and memory allcation
    - Disadvantage: Large overheads on searching elements in the list

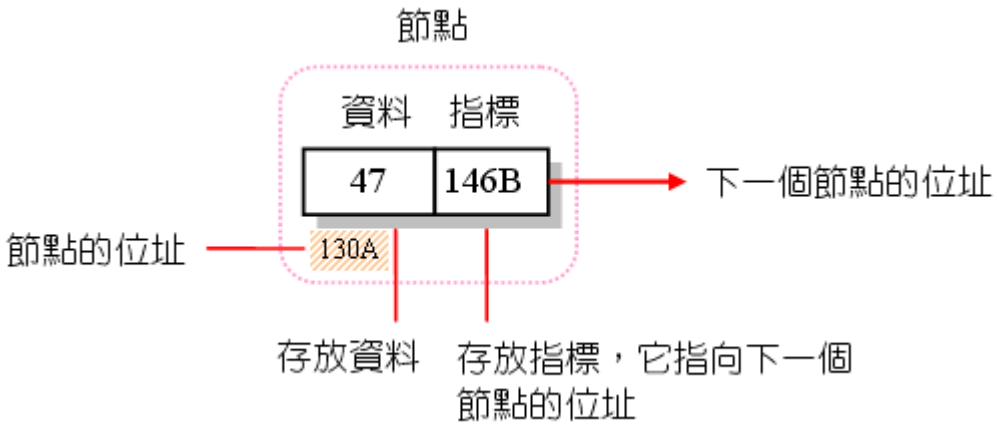
List with Array





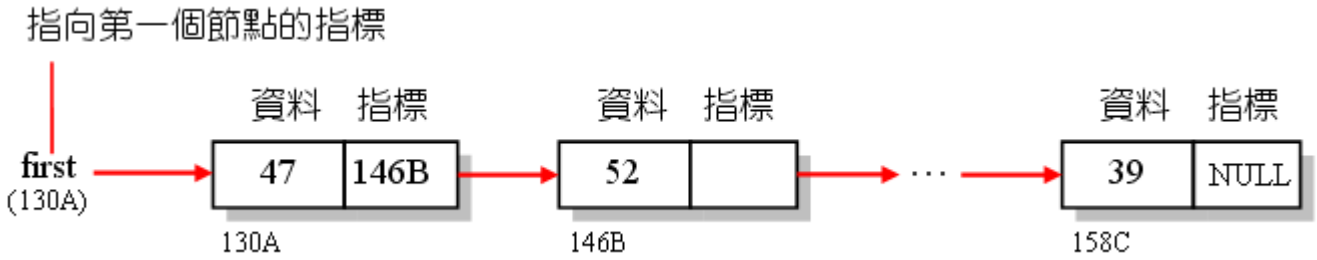
Linked List

- A node of a linked list consists of at least two fields.
- ***For example***:
  - The first field is to store data (資料).
  - The second field is a pointer (指標) to store the address of the next element.



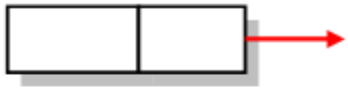
Linked List (Cont.)

- A linked list is composed of multiple nodes (節點).
  - Each node points to the next node.





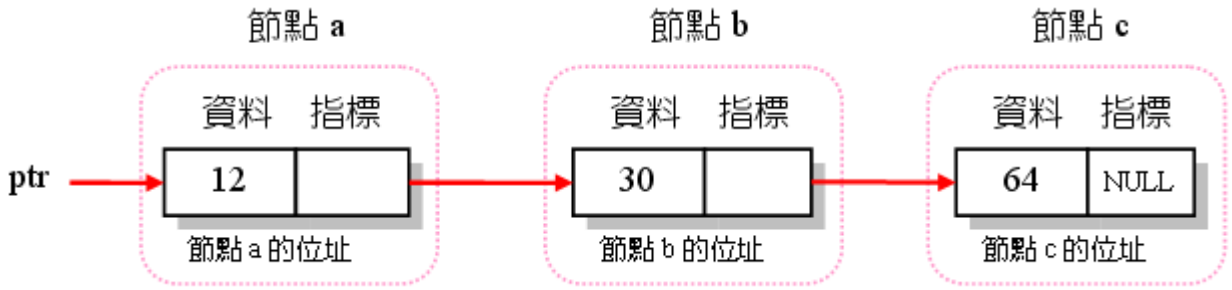
Linked List Consturction




```
struct node
{
   int data;              /* 資料成員 */
   struct node *next;     /* 鏈結成員，指向下一個節點的指標 */
};
```



Linked List Example (1/3)




```
01  /* 建立 3 節點的鏈結串列 */
02  #include<stdio.h>
03  #include<stdlib.h>
04  struct node
05  {
06     int data;                    /* 資料成員  */
07     struct node *next;           /* 鏈結成員，存放指向下一個節點的指標  */
08  };
09  typedef struct node NODE;       /* 將 struct node 定義成 NODE 型態 */
10
```



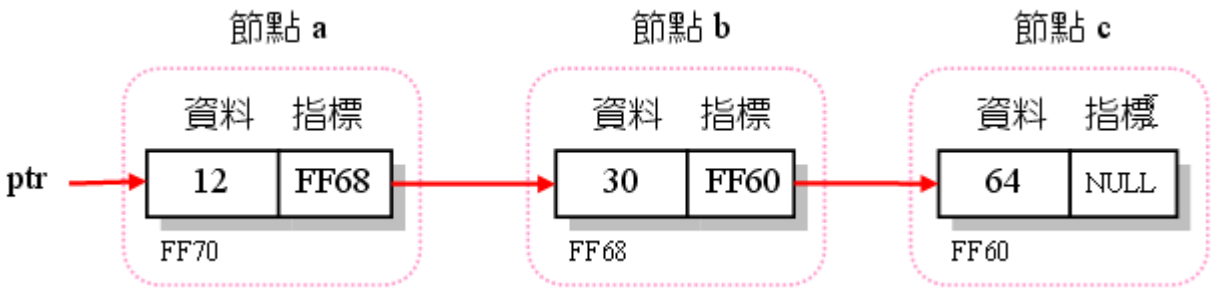
Linked List Example (2/3)

```
11  int main(void)
12  {
13      NODE a,b,c;          /* 宣告 a,b,c 為 NODE 型態的變數 */
14      NODE *ptr=&a;        /* 宣告 ptr,並將它指向節點 a */
15      a.data=12;           /* 設定節點 a 的 data 成員為 12 */
16      a.next=&b;           /* 將節點 a 的 next 成員指向下一個節點，即 b */
17      b.data=30;
18      b.next=&c;
19      c.data=64;
20      c.next=NULL;         /* 將節點 c 的 next 成員設成 NULL */
21
22      while (ptr!=NULL)    /* 當 ptr 不是 NULL 時，則執行下列敘述 */
23      {
24         printf("address=%p, ",ptr);          /* 印出節點的位址 */
25         printf("data=%d, ",ptr->data);       /* 印出節點的 data 成員 */
26         printf("next=%p\n",ptr->next);       /* 印出下一個節點的位址 */
27         ptr=ptr->next;                       /* 將 ptr 指向下一個節點 */
28      }
29      system("pause");
30      return 0;
31  }
```



Linked List Example (3/3)

```
/* OUTPUT----------------------
address=0022FF70, data=12, next=0022FF68
address=0022FF68, data=30, next=0022FF60
address=0022FF60, data=64, next=00000000
---------------------------------------*/
```



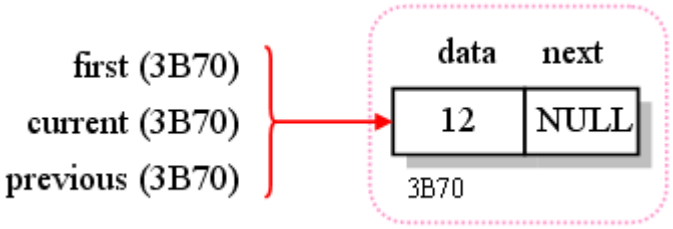


Linked List with Dynamic Allocation (1/3)

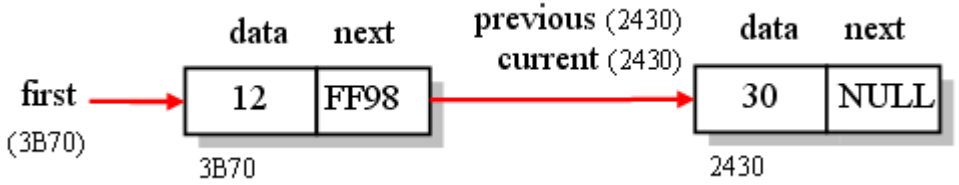
```
01  /* 以動態記憶體配置鏈結串列 */
02  #include<stdio.h>
03  #include<stdlib.h>
04  struct node
05  {
06     int data;                    /* 資料成員  */
07     struct node *next;           /* 鏈結成員，存放指向下一個節點的指標  */
08  };
09  typedef struct node NODE;       /* 將 struct node 定義成 NODE 型態 */
10
11  int main(void)
12  {
13     int i,val,num;
14     NODE *first,*current,*previous;    /* 建立 3 個指向 NODE 的指標 */
15     printf("Number of nodes: ");
16     scanf("%d",&num);                 /* 輸入節點的個數 */
```

Linked List with Dynamic Allocation (2/3)

```
17    for(i=0;i<num;i++)
18    {
19       current=(NODE *) malloc(sizeof(NODE));  /* 建立新的節點 */
20       printf("Data for node %d: ",i+1);
21       scanf("%d",&(current->data));          /* 輸入節點的 data 成員 */
22       if(i==0)                               /* 如果是第一個節點 */
23          first=current;                      /* 把指標 first 指向目前的節點 */
24       else
25          previous->next=current;   /* 把前一個節點的 next 指向目前的節點 */
26       current->next=NULL;          /* 把目前的節點的 next 指向NULL */
27       previous=current;            /* 把前一個節點設成目前的節點 */
28    }
```



After the first iteration



After the second iteration

Linked List with Dynamic Allocation (3/3)

```
29      current=first;              /* 設定 current 為第一個節點 */
30      while (current!=NULL)       /* 如果還沒有到串列末端，則進行走訪的動作 */
31      {
32         printf("address=%p, ",current);     /* 印出節點的位址 */
33         printf("data=%d, ",current->data); /* 印出節點的 data 成員 */
34         printf("next=%p\n",current->next); /* 印出節點的 next 成員 */
35         current=current->next;          /* 設定 current 指向下一個節點 */
36      }
37      system("pause");
38      return 0;
39  }
```

```
/* OUTPUT----------------------

Number of nodes: 3
Data for node 1: 12
Data for node 2: 30
Data for node 3: 64
address=003D3B70, data=12, next=003D2430
address=003D2430, data=30, next=003D2440
address=003D2440, data=64, next=00000000
---------------------------------------*/
```

Basic Operations of Linked List (1/4)

- Header file declaration for the basic operation functions (e.g., linklist.h)

```
01  /* linklist.h, 鏈結串列的標頭檔 */
02  struct node
03  {
04     int data;                    /* 資料成員  */
05     struct node *next;           /* 鏈結成員，存放指向下一個節點的指標  */
06  };
07  typedef struct node NODE;       /* 將 struct node 定義成 NODE 型態 */
08
09  NODE *createList(int *, int);              /* 串列建立函數 */
10  void printList(NODE *);                    /* 串列列印函數 */
11  void freeList(NODE *);                     /* 釋放串列記憶空間函數 */
12  void insertNode(NODE *,int );              /* 插入節點函數 */
13  NODE *searchNode(NODE *, int );            /* 搜尋節點函數 */
14  NODE *deleteNode(NODE *, NODE *);          /* 刪除節點函數 */
```

Basic Operations of Linked List (2/4)

- Use arr[ ] to create a linked list

```
01  /* createList()，串列建立函數 */
02  NODE *createList(int *arr, int len)
03  {
04     int i;
05     NODE *first,*current,*previous;
06     for(i=0;i<len;i++)
07     {
08        current=(NODE *) malloc(sizeof(NODE));
09        current->data=arr[i];          /* 設定節點的資料成員 */
10        if(i==0)                       /* 判別是否為第一個節點 */
11           first=current;
12        else
13           previous->next=current;    /* 把前一個節點的 next 指向目前節點 */
14        current->next=NULL;
15        previous=current;
16     }
17     return first;
18  }
```

Basic Operations of Linked List (3/4)

- The function to print the data in the nodes of the link list

```
01  /* printList()，串列列印函數 */
02  void printList(NODE* first)
03  {
04     NODE* node=first;          /* 將 node 指向第一個節點 */
05     if(first==NULL)            /* 如果串列是空的，則印出 List is empty! */
06        printf("List is empty!\n");
07     else                       /* 否則走訪串列，並印出節點的 data 成員 */
08     {
09        while(node!=NULL)
10        {
11           printf("%3d", node->data);
12           node=node->next;
13        }
14        printf("\n");
15     }
16  }
```

Basic Operations of Linked List (4/4)

- Free the memory space allocated for the linked list

```
01  /* freeList()，釋放記憶空間函數 */
02  void freeList(NODE* first)
03  {
04     NODE *current,*tmp;
05     current=first;                    /* 設定 current 指向第一個節點 */
06     while (current!=NULL)
07     {
08        tmp=current;                   /* 先暫存目前的節點 */
09        current=current->next;         /* 將 current 指向下一個節點 */
10        free(tmp);                     /* 將暫存的節點釋放掉 */
11     }
12  }
```



Linked List Example with Operation Functions

Use array {14,27,32,46} to create a linked list

```
01  /*  鏈結串列的建立、列印與記憶體的釋放 */
02  #include<stdio.h>
03  #include<stdlib.h>
04  #include "linklist.h"                    /* 含括標頭檔 linklist.h */
05
06  int main(void)
07  {
08     NODE *first;
09     int arr[]={14,27,32,46};              /* 建立陣列 arr[] */
10     first=createList(arr,4);              /* 以陣列元素建立鏈結串列 */
11     printList(first);                     /* 印出鏈結串列的內容  */
12     freeList(first);                      /* 釋放記憶空間  */
13     system("pause");
14     return 0;
15  }
16  /* 請將 createList()函數放在此處 */
17  /* 請將 printList()函數放在此處  */
18  /* 請將 freeList()函數放在此處   */
```

```
/*  OUTPUT--
 14  27  32  46
--------------------*/
```



Node Searching

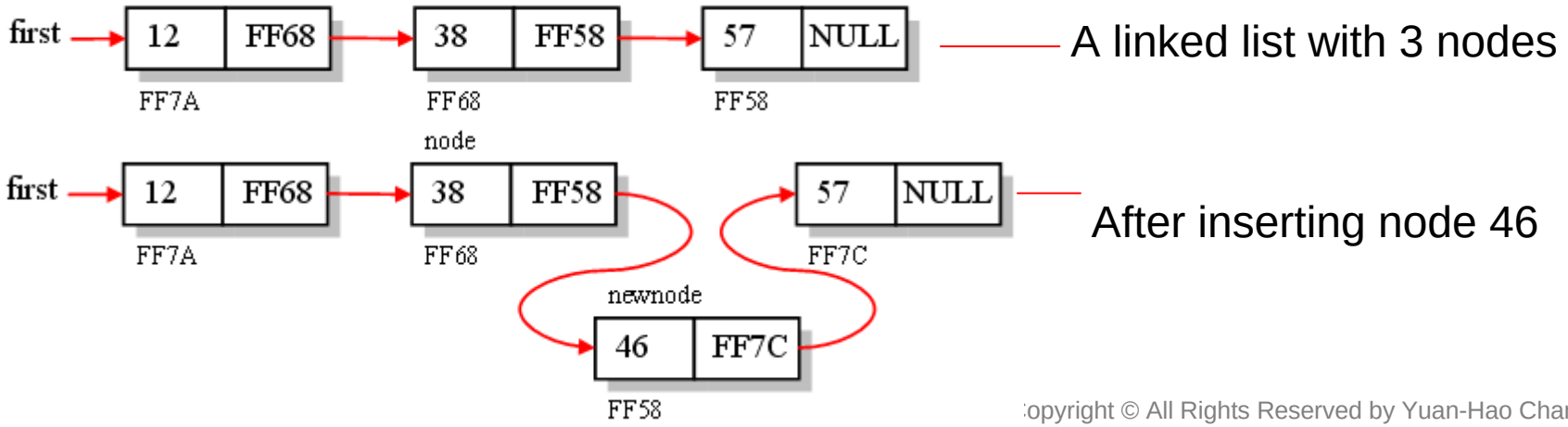
- This function could search where is the node containing “item.”

```
01  /* searchNode()函數，可傳回第一個存放 item 之節點的位址 */
02  NODE* searchNode(NODE* first, int item)
03  {
04     NODE *node=first;
05     while(node!=NULL)
06     {
07        if(node->data==item)       /* 如果 node 的 data 等於 item */
08           return node;            /* 傳回 node，即該節點的位址 */
09        else
10           node=node->next;        /* 否則將指標指向下一個節點 */
11     }
12     return NULL;          /* 如果找不到符合的節點，則傳回 NULL */
13  }
```



Node Insertion

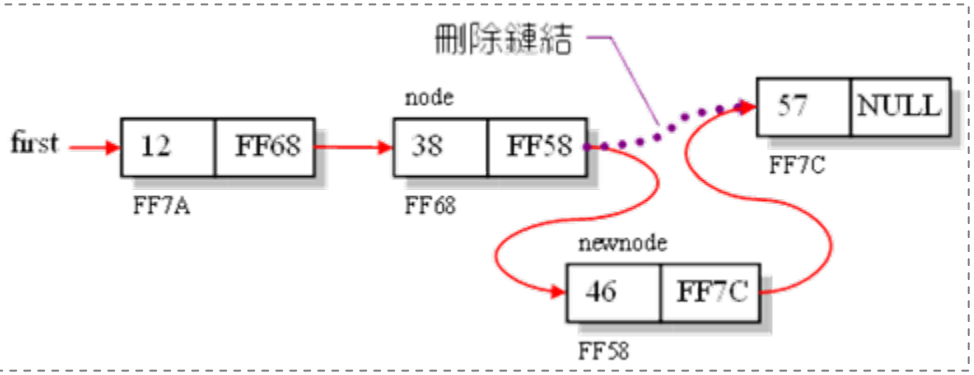
```
01  /* insertNode()，可在 node 之後加入一個新的節點 */
02  void insertNode(NODE *node,int item)
03  {
04     NODE *newnode;
05     newnode=(NODE *) malloc(sizeof(NODE));     /* 取得新節點的位址 */
06     newnode->data=item;                        /* 將新節點的 data 設為 item */
07     newnode->next=node->next;    /* 將新節點的 next 設為原節點的 next */
08     node->next=newnode;                        /* 將原節點的 next 指向新節點 */
09  }
```





Application: Node Searching and Insertion

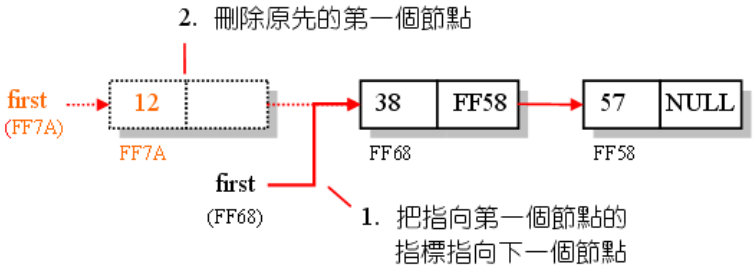
```
01  /* 節點的搜尋與插入 */
02  #include<stdio.h>
03  #include<stdlib.h>
04  #include "linklist.h"
05  int main(void)
06  {
07     NODE *first,*node;
08     int arr[]={12,38,57};
09     first=createList(arr,3);          /* 建立鏈結串列 */
10     printList(first);
11
12     node=searchNode(first,38);        /* 找出節點資料值為 38 的位址 */
13     insertNode(node,46);              /* 將節點 46 鏈結在節點 38 之後 */
14     printList(first);                 /* 印出節點的內容 */
15     freeList(first);
16     system("pause");
17     return 0;
18  }
```



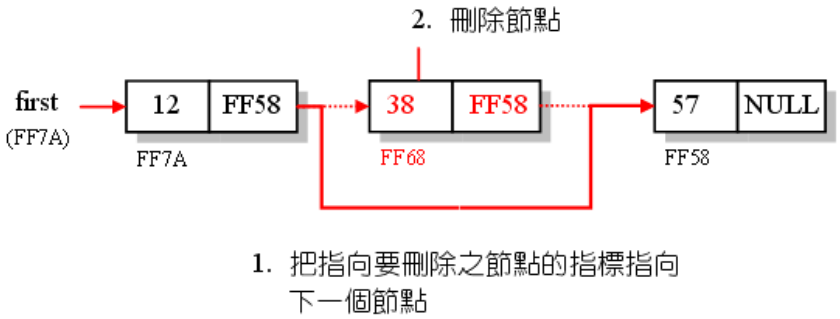


Node Deletion

- Three condition upon node deletion:
  - 1. An empty list: Node operation is performed.
  - 2. The deleted node is the first node in the list:
    - Move “first” to the next node”, and then delete the first node.



  - 3. The deleted node is not the first node in the list:
    - Pont the next node of the next node, and free the space.





Node Deletion Function

```
01  /* 刪掉 node，傳回刪掉 node 之後，串列第一個節點的位址 */
02  NODE* deleteNode(NODE *first, NODE *node)
03  {
04     NODE *ptr=first;
05     if(first==NULL)  /* 如果串列是空的，則印出 Nothing to delete! */
06     {
07        printf("Nothing to delete!\n");
08        return NULL;
09     }
10     if(node==first)           /* 如果刪除的是第一個節點 */
11        first=first->next;     /* 把 first 指向下一個節點 */
12     else                      /* 如果刪除的是第一個節點以外的其它節點 */
13     {
14        while(ptr->next!=node)    /* 找到要刪除之節點的前一個節點 */
15           ptr=ptr->next;
16        ptr->next=node->next;     /* 重新設定 ptr 的 next 成員 */
17     }
18     free(node);
19     return first;
20  }
```



Node Deletion Example (1/2)

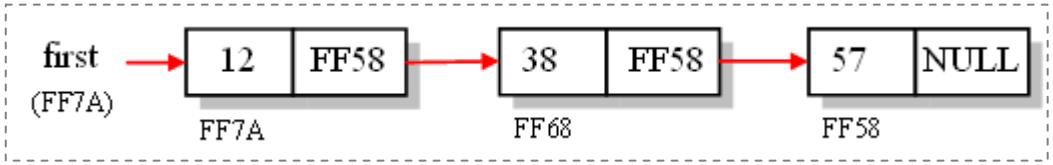
```
/* OUTPUT--------------
 12 38 57   ———— 執行完第 10 行的結果
 12 57      ———— 執行完第 14 行的結果
 57         ———— 執行完第 17 行的結果
List is empty! — 執行完第 20 行的結果
-------------------------------*/
```

```
01  /* 節點刪除的範例 */
02  #include<stdio.h>
03  #include<stdlib.h>
04  #include "linklist.h"
05  int main(void)
06  {
07     NODE *first,*node;
08     int arr[]={12,38,57};
09     first=createList(arr,3);          /* 建立鏈結串列 */
10     printList(first);
11
12     node=searchNode(first,38);        /* 找出節點資料值為 38 的位址 */
13     first=deleteNode(first,node);     /* 將節點 38 刪除掉 */
14     printList(first);                 /* 印出節點的內容 */
15
```



Node Deletion Example (2/2)

```
16      first=deleteNode(first,first);    /* 刪除掉第一個節點*/
17      printList(first);                 /* 印出節點的內容 */
18
19      first=deleteNode(first,first);    /* 刪除掉第一個節點*/
20      printList(first);                 /* 印出節點的內容 */
21
22      freeList(first);
23
24      system("pause");
25      return 0;
26  }
```




```
/* OUTPUT--------------
 12 38 57   ———— 執行完第 10 行的結果
 12 57      ———— 執行完第 14 行的結果
 57         ———— 執行完第 17 行的結果
List is empty! — 執行完第 20 行的結果
--------------------------------*/
```

Lab 15

- 試以malloc() 配置3個可存放double型態的變數（即利用malloc(3*sizeof(double)) 的語法）之記憶空間，然後在for迴圈裡，分別以scanf() 函數輸入三個浮點數，最後再計算它們的總和與平均值。
- 定義下列結構:

```
struct student {
        int num;
        struct student *next;
};
```

試使用上列結構建立可存放 int 型態的變數 linked list。請使用while迴圈，在 while 迴圈以scanf() 函數輸入整數並存到一個新建的node，然後把該node加在linked list的最後，若輸入值為零，則離開迴圈並印出linked list中所有的直並算出list中的node數及平均值。