

C++資料結構與程式設計

陣列與鏈結串列

NTU CSIE

Outline

結構陣列

鏈結串列

- 單向鏈結串列之資料型態
- 單向鏈結串列之基本運算

作業

結構陣列

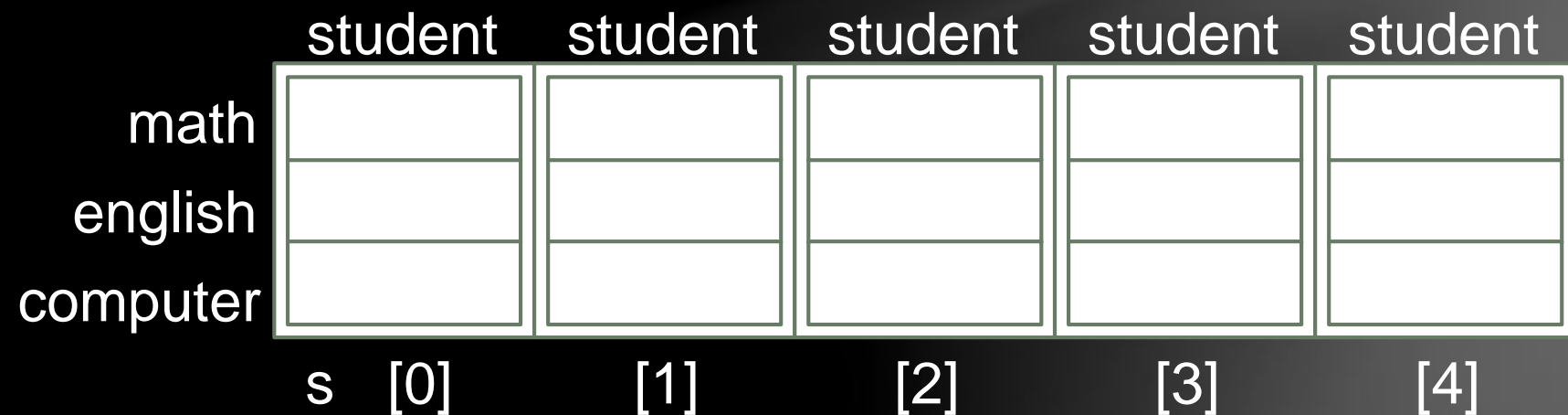
▶ 優點

- ▶ 使用容易

▶ 缺點

- ▶ 刪除與插入造成資料移動頻繁
- ▶ 浪費不必要之記憶體
- ▶ 陣列長度為常數, 可能會不夠用

```
#include<stdio.h>
struct _student
{
    int math;
    int english;
    int computer;
};
typedef struct _student student;
int main()
{
    student s[5];
    return 0;
}
```



Outline

結構陣列

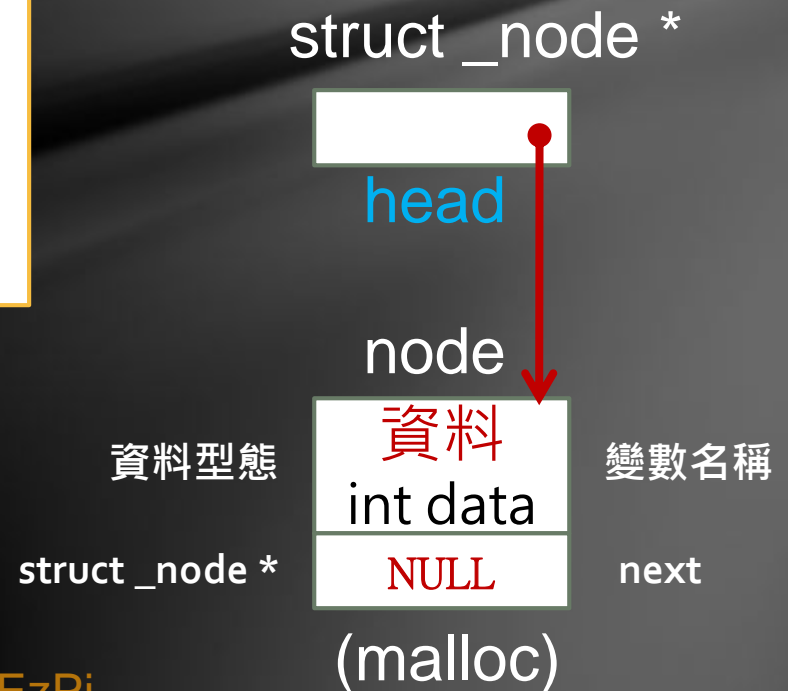
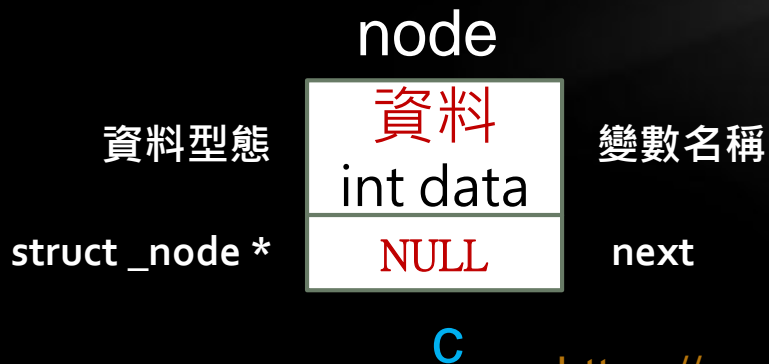
鏈結串列

- 單向鏈結串列之資料型態
- 單向鏈結串列之基本運算

作業

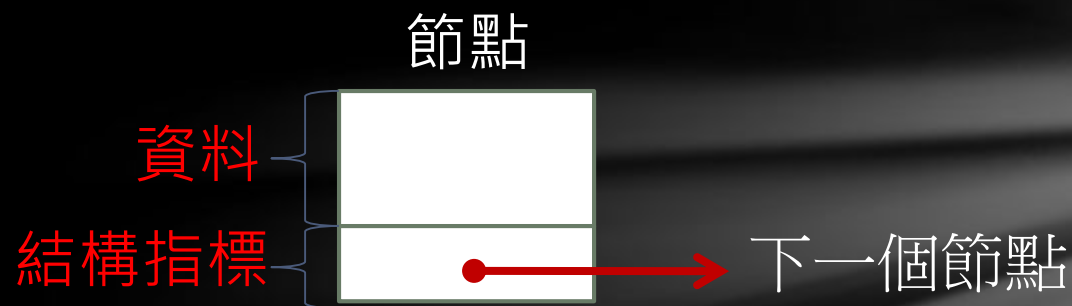
靜態與動態結構

```
struct _node
{
    資料型態 變數名稱;
    int data;
    struct _node *next;
};
typedef struct _node node;
node c;
node *head;
c.next = NULL;
head = (node *)malloc(sizeof(node));
head->next = NULL;
```



鏈結串列節點

節點：鏈結串列中最基本的單位



節點 = 資料 + 結構指標

定義鏈結串列節點結構

鏈結串列透過儲存元素在記憶體之位址為**指標(Pointer)**或**鏈結(Link)**取得**下一個節點**。

定義節點結構

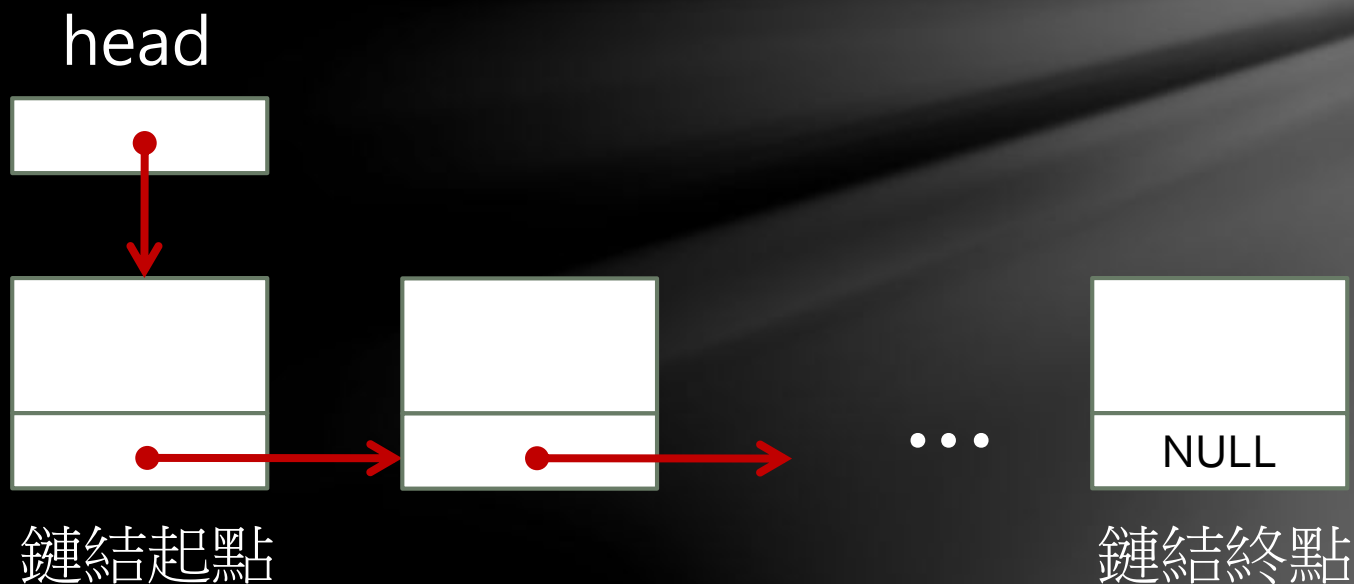
```
struct _node
{
    資料型態 變數名稱;
    int data;
    struct _node *next;
};
typedef struct _node node;
```



單向鏈結串列

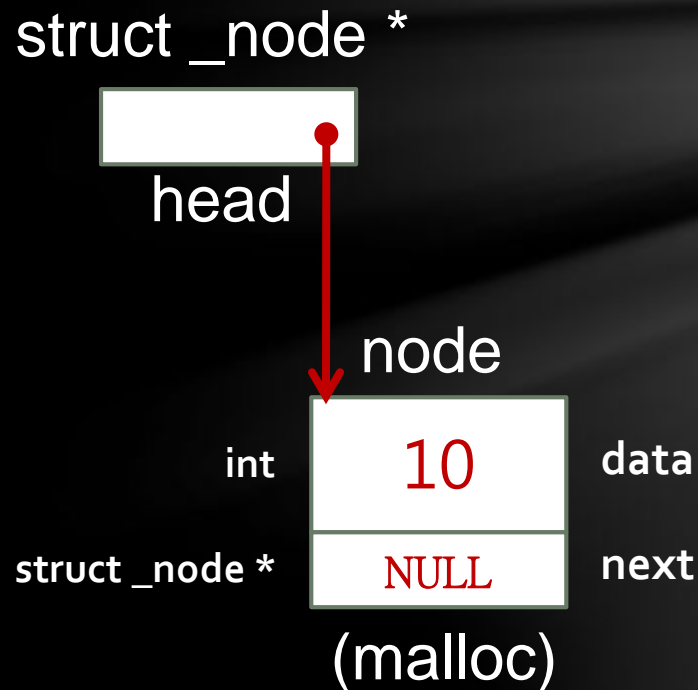
單向鏈結串列之結構如下圖所示

- head：指向串列前端之指標



小練習（建立鏈結串列節點）_(ex01 build-1.c)

定義一個鏈結串列節點結構如下圖所示，並使用 head 指標指向動態配置之節點



連續鏈結串列

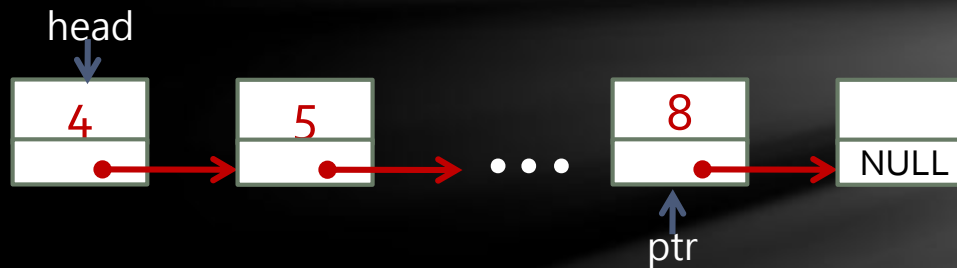
```
node *head, *ptr;  
head = (node *)malloc(sizeof(node));  
ptr = head;  
int value;  
  
scanf("%d", &value);  
ptr->data = value;  
ptr->next = (node *)malloc(sizeof(node));  
ptr = ptr->next;
```



小練習 (建立鏈結串列)

定義一個鏈結串列，將順序輸入的資料存在動態資料結構上(ex:輸入5筆)，然後將此串列資料順序印出來

(ex04 build-3 add list.c)



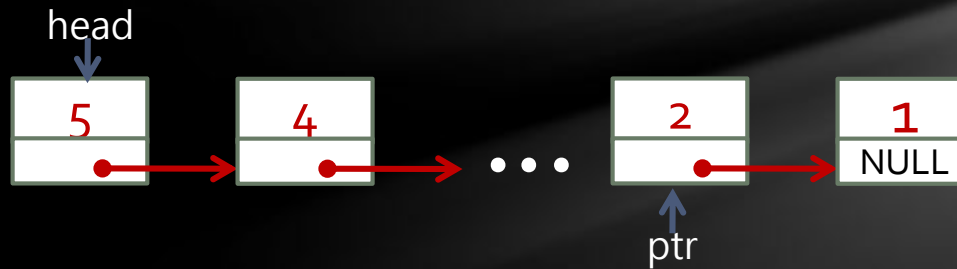
問題

- 這樣加入資料仍要指定是加幾筆
因為最後加入節點後會新增一空白節點
如何將每次新增節點都當作是最後一次呢？
(請修改組成節點的順序) <https://goo.gl/2CpZAB>
- <https://jgirl.ddns.net/problem/0/2017>
- 如何簡化程式碼？(非必要)
- 如何釋放記憶體？ <https://goo.gl/77R3BS>
- 若想要反序列印呢？ <https://goo.gl/k5fEuP>
 - 遞迴反序列印
 - 雙向鏈結串列
 - 反轉鏈結串列
 - 往前新增資料

小練習 (建立鏈結串列)

定義一個鏈結串列，將順序輸入的資料存在動態資料結構上(ex:輸入5筆)，然後將此串列資料反序列印出來

(ex06 build-4 reverse.c)



<https://jgirl.ddns.net/problem/0/2018>

白版題？

- 程式設計師的薪水
- 白板題
- https://www.ptt.cc/bbs/Tech_Job/M.1449039102.A.956.html
- <https://btrend.amassly.com/post/YduuqPVeJ1/>
- <https://btrend.amassly.com/post/9EcKy8gmQ0y3TUepT1eR0g%3D%3D/>
- https://www.ptt.cc/man/Tech_Job/DB04/D730/DC84/M.1384846171.A.BAF.html

Outline

結構陣列

鏈結串列

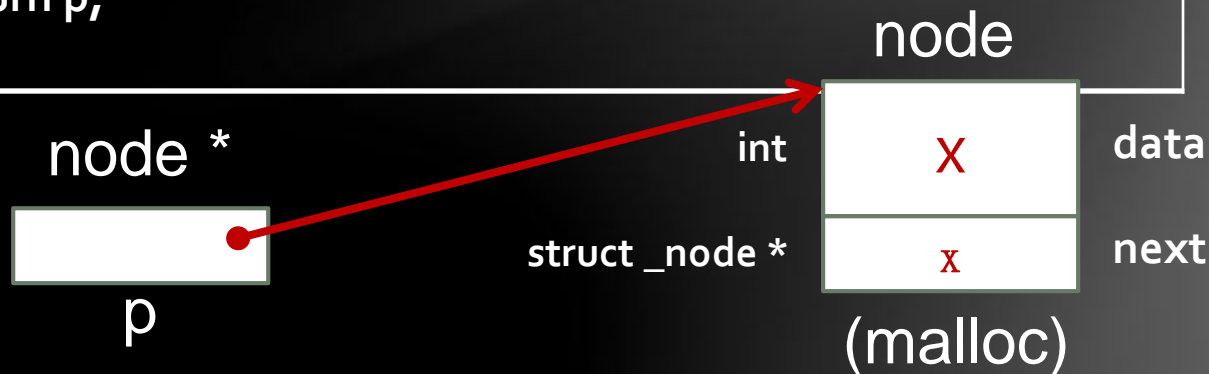
- 單向鏈結串列之資料型態
- 單向鏈結串列之基本運算-函式整合版

作業

新增節點

動態配置一節點之記憶體

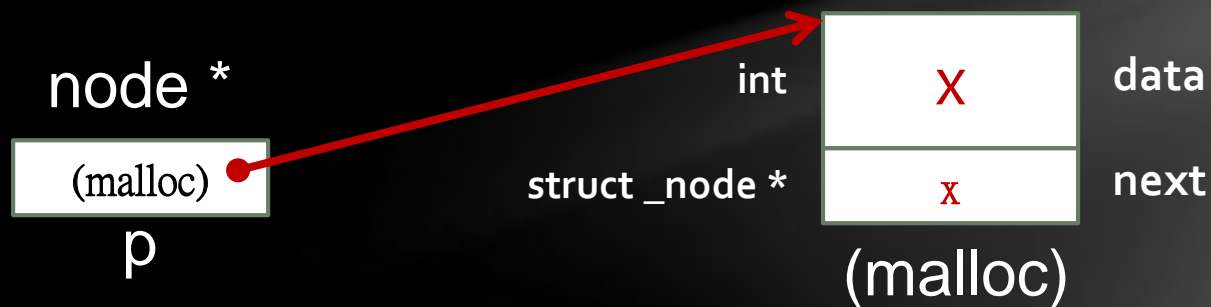
```
node *getnode () /* 此函數產生一個新節點 */
{
    node *p;
    p = (node *) malloc(sizeof(node));
    /* malloc 會動態地配置大小為sizeof 的記憶體*/
    /* sizeof 會傳回一個型態為node之值*/
    if ( p == NULL)
    {
        printf ("記憶體不足");
        exit(1);
    }
    return p;
}
```



釋放節點

歸還一個節點之記憶體

```
void freenode (node *p) /* 此函數將節點還給記憶體 */  
{  
    free(p);  
}
```

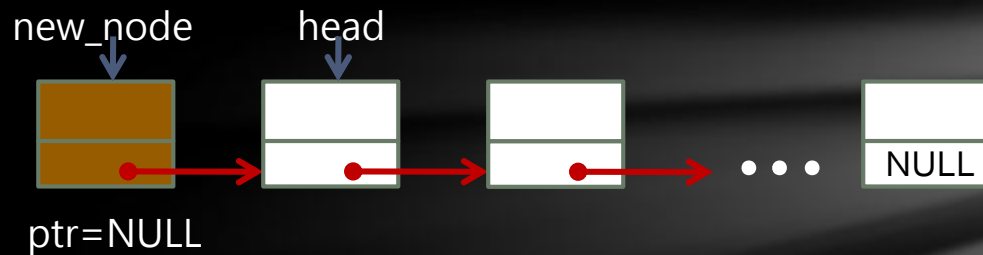


插入節點

由鏈結串列加入一個節點

一個節點之插入有三種情況：

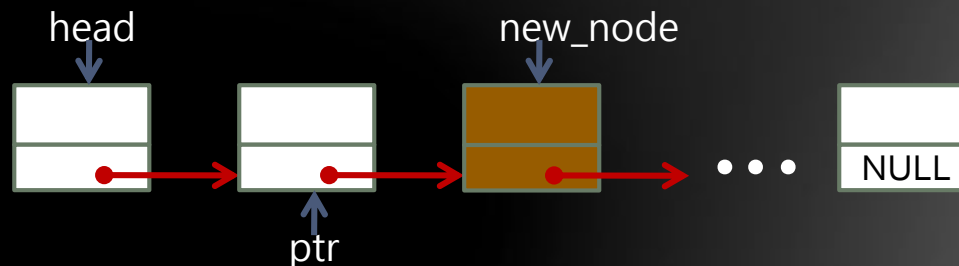
- 節點加於第一個節點之前



- 節點加於最後一個節點之後

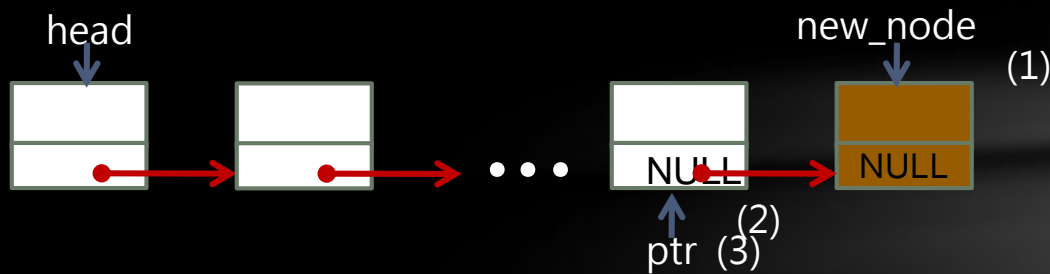


- 加於節點中間任何一個位置



插入節點

節點加於最後一個節點之後



Ex1:

```
node *new_node; //(1)
new_node = getnode();
ptr->next = new_node //(2)
/* 指向新節點 */
new_node -> next = NULL;
```

Ex2:

```
ptr->next = (node *)malloc(sizeof(node)); //(1&2)
ptr = ptr->next; //(3)
```

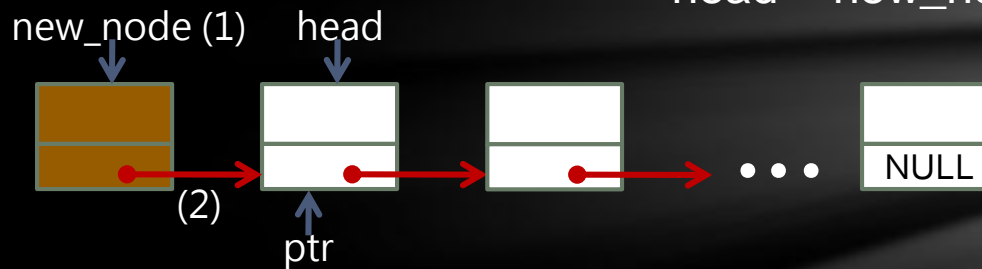


插入節點

節點加於第一個節點之前

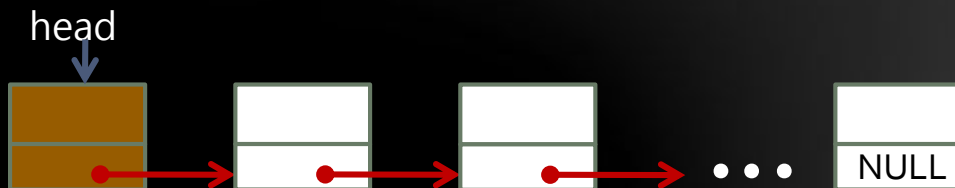
Ex1:

```
node *new_node  
new_node = getnode(); //(1)  
new_node->next = head; //(2)  
head = new_node;
```



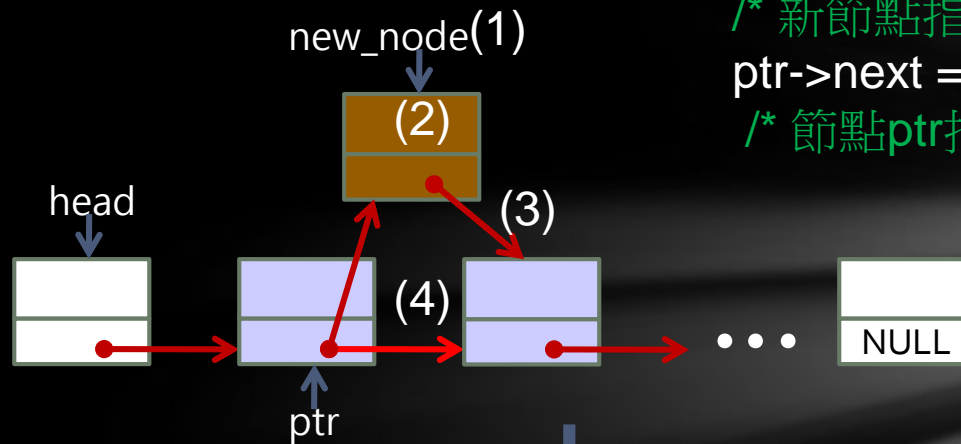
Ex2:

```
head = (node *)malloc(sizeof(node)); //(1)  
head -> next = ptr; //(2)  
ptr = head; //(3)
```

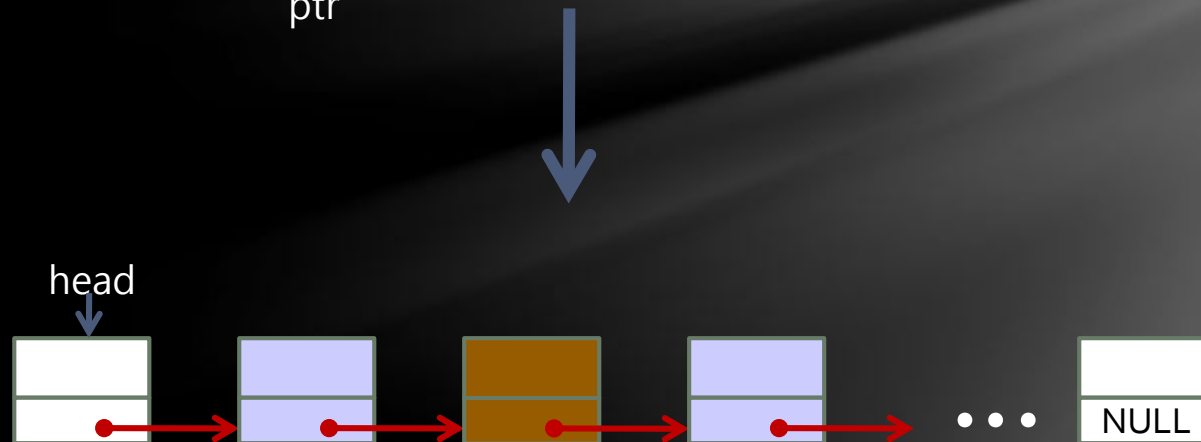


插入節點

加於節點中間任何一個位置



```
node *new_node; //(1)
new_node = getnode(); //(2)
new_node->next = ptr->next; //(3)
/* 新節點指向下一節點*/
ptr->next = new_node; //(4)
/* 節點ptr指向新節點*/
```



插入節點

```
node *insert_node (node *head, node *ptr, node input)
{
    node *new_node;    /* 新節點指標變數 */
    new_node = getnode(); /* 建立新節點,取得一個可用節點 */
    *new_node = input;  /* 建立節點內容 */ https://goo.gl/niwAFr
    new_node->next = NULL; /* 設定指標初值 */

    if ( ptr == NULL ) /* 指標ptr是否是NULL */
    {
        /* 第一種情況: 插入第一個節點 */
        new_node->next = head; /* 新節點成為串列開始 */
        head = new_node;
    }
    else
    {
        if ( ptr->next == NULL ) /* 是否是串列結束 */
        {
            /* 第二種情況: 插入最後一個節點 */
            ptr->next = new_node; /* 最後指向新節點 */
        }
        else
        {
            /* 第三種情況: 插入成為中間節點 */
            new_node->next = ptr->next; /* (3) 新節點指向下一節點 (3) */
            ptr->next = new_node; /* 節點ptr指向新節點 (4) */
        }
    }
    return head;
}
```

<https://goo.gl/XCy9Xw>

小練習 (插入鏈結串列節點) (ex07 interface i.c)

延續上一小練習

https://www.tutorialspoint.com/online_c_formatter.htm

<https://codebeautify.org/c-formatter-beautifier>

寫一個使用者介面，輸入 **i**，接著輸入一個數字 **value**，可插入一筆資料節點中之 **data** 為 **value** 於串列最後。

建立一個鏈結串列如下圖所示：

輸入 **1**，可將全部內容列印出來。

輸入 **q**，離開程式。

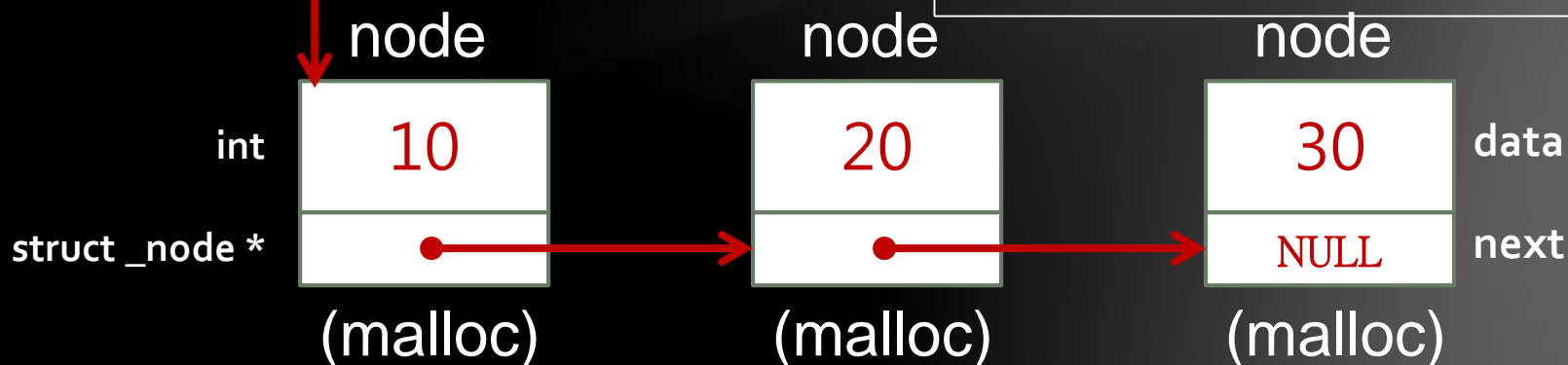
struct _node *



head

```
node *head, *ptr;  
node n;  
char key;  
int value;  
head = NULL;
```

```
scanf("%d",&value);  
n.data = value;  
ptr = head;  
if(head==NULL){  
    head=insert_node(head, NULL, n);  
}  
else {  
    while(ptr->next != NULL)  
        ptr = ptr->next;  
    head=insert_node(head, ptr, n);  
}  
printf("Insert ok\n");
```



尋找節點

走訪串列

將找到之節點位置回傳

```
scanf("%d",&value);  
ptr = find_node(head, value);  
if(ptr != NULL)  
    printf("found: %d\n", ptr->data);  
else  
    printf("Not found\n");
```

```
node *find_node(node *head, int num)  
{  
    node *ptr;  
    ptr = head; /* 指向串列起始 */  
    while ( ptr != NULL ) /* 走訪串列 */  
    {  
        if ( ptr->data == num ) /* 找尋data */  
            return ptr; /* 找到 (ptr != NULL) */  
        ptr = ptr->next; /* 指向下一節點 */  
    }  
    return ptr; /* 找不到 (ptr == NULL) */  
}
```

<https://goo.gl/6FJqoL>

小練習（尋找鏈結串列節點）(ex07 interface i.c)

延續上一小練習。

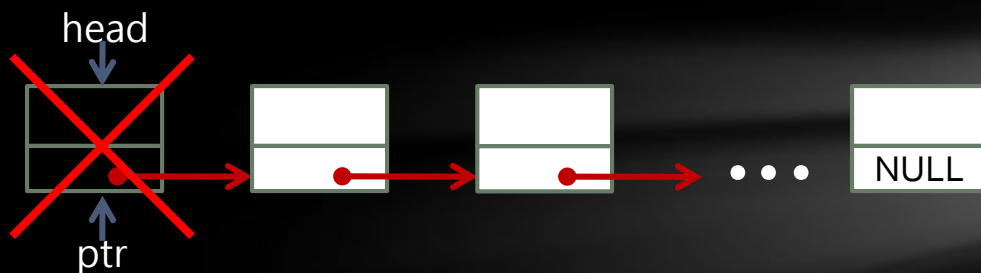
寫一個使用者介面，輸入**f**，接著輸入一個數字value，可將一筆資料節點中之data與value相同者印出資料。

刪除節點

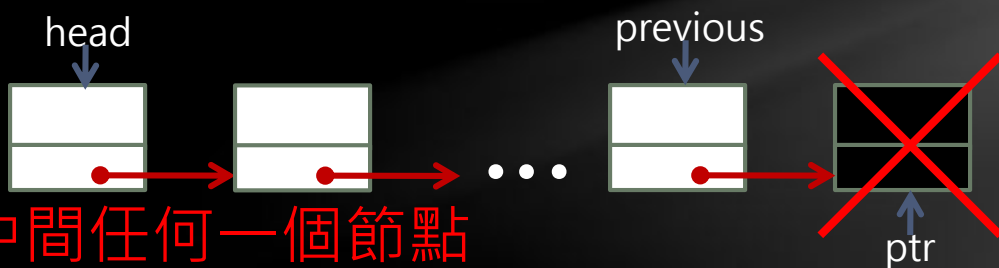
由鏈結串列中刪除一個節點

一個節點之刪除有三種情況：

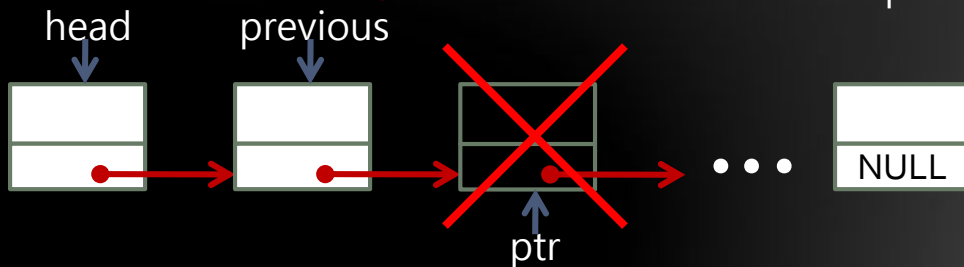
- 刪除第一個節點



- 刪除最後一個節點

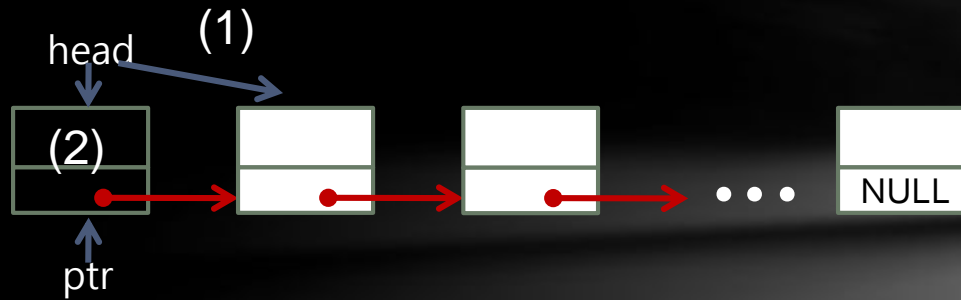


- 刪除中間任何一個節點



刪除節點

刪除第一個節點



```
head = head->next; //(1)  
freenode(ptr); //(2)  
return(head); /* reset 起始節點指標 */
```

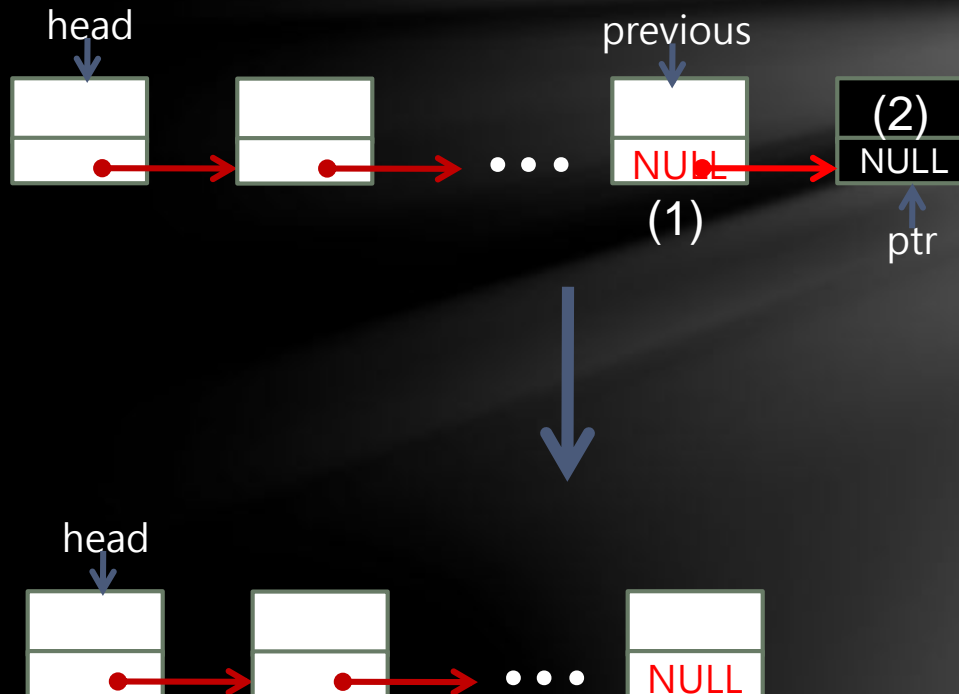


刪除節點

刪除最後一個節點

```
previous = head;  
while ( previous->next != ptr ) /* 找節點ptr的前節點 */  
    previous = previous->next;
```

```
previous->next = NULL; //(1) /* 最後一個節點 */  
freenode(ptr); //(2) /* 此函數將節點歸還給記憶體 */
```



刪除節點

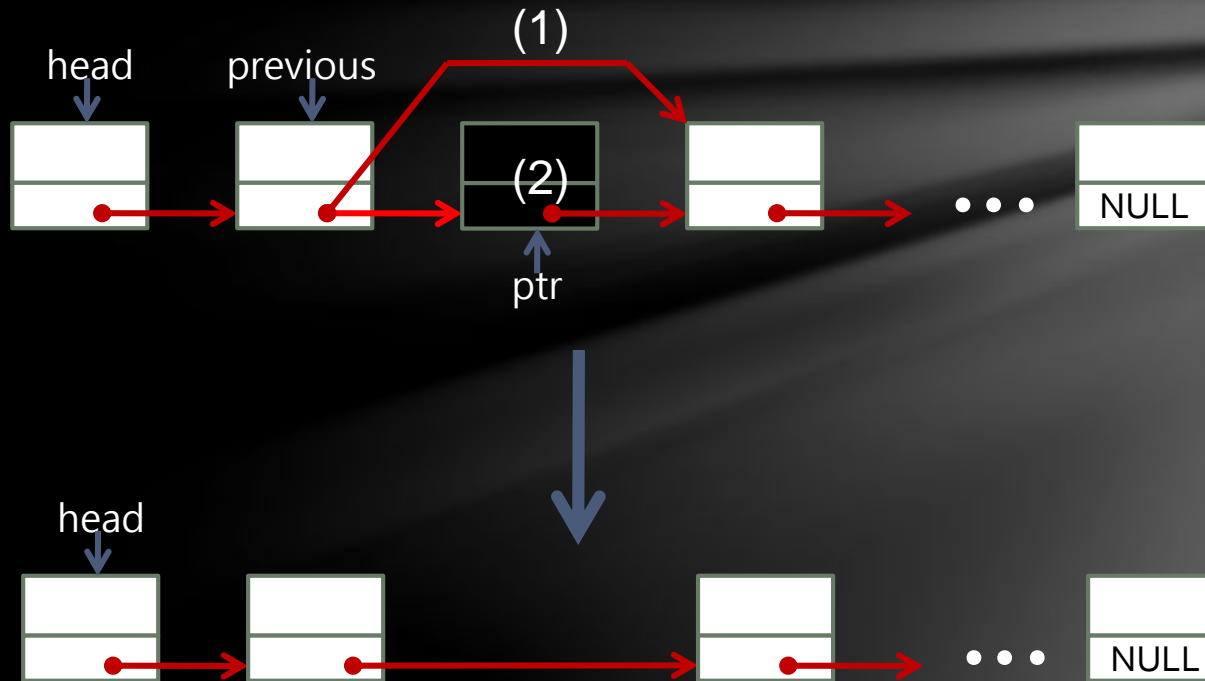
刪除中間任何一個節點

```
previous = head;  
while ( previous->next != ptr ) /* 找節點ptr的前節點 */  
    previous = previous->next;
```

/* 第三種情況: 刪除中間節點 */

```
previous->next = ptr->next; //(1)
```

```
freenode(ptr); //(2) /* 此函數將節點歸還給記憶體 */
```



刪除節點

```
node *delete_node(node *head, node *ptr)
{
    node *previous; /* 指向前一節點 */
    if ( ptr == head ) /* 是否是串列開始 */
        /* 第一種情況: 刪除第一個節點 */
        {
            head = head->next;
        }
    else
    {
        previous = head;
        while ( previous->next != ptr ) /* 找節點ptr的前節點 */
            previous = previous->next;
        if ( ptr->next == NULL ) /* 是否是串列結束 */
            /* 第二種情況: 刪除最後一個節點 */
            previous->next = NULL; /* 最後一個節點 */
        else
            /* 第三種情況: 刪除中間節點 */
            previous->next = ptr->next; /* 圖(3)之步驟(1) */
    }
    freenode(ptr); /* 此函數將節點歸還給記憶體 */
    return(head);
}
```

```
scanf("%d",&value);
ptr = find_node(head, value);
if(ptr != NULL)
{
    head = delete_node(head, ptr);
    printf("Delete ok\n");
}
else
    printf("Can not delete\n");
```

小練習（刪除鏈結串列節點）(ex07 interface i.c)

延續上一小練習。

寫一個使用者介面，輸入`d`，接著輸入一個數字`value`，可將一筆資料節點中之`data`與`value`相同者刪除(假設輸入之`value`不會重覆)。

鏈結串列長度

計算鏈結串列head之長度

```
int length (node *head ) /* 此函數計算節點之鏈結長度 */
{
    int num=0;
    node *ptr = head;
    while (ptr != NULL)
    {
        num ++;
        ptr = ptr->next;
    }
    return num;
}
```


小練習 (ex07 interface i.c)

將上述鏈結串列功能整合起來成為同一程式
功能

- 輸入'i'，接著輸入一個數字，可插入節點中之data為value於串列最後
- 輸入'd'，接著輸入一個數字，可將節點中之data與value相同者刪除(假設輸入之value不會重覆)
- 輸入'f'接著輸入一個數字，可將一筆資料節點中之數字相同者印出
- 輸入'l'印出串列所有節點內容並顯示目前資料筆數
- 輸入'q' 離開程式
- ※bonus:可設計插入位置於最前/中間/最後

延申題目

- 設計新增節點插入位置於最前/中間/最後
 - 加在中間1:輸入要加在第n個節點
 - 須注意是否超過現有節點的長度
 - 若是就加在最後面
 - 加在中間2:輸入要加在數值為x的節點後面
 - 改進插入最後的方式，不用每次都從頭移到最後
 - 加入一指標(tail)永遠指向最後一個節點。
<https://jgirl.ddns.net/problem/0/2030>
- 設計新增節點加入串列後即是排序完成的
<https://jgirl.ddns.net/problem/0/2032>
- 如何對沒有排序的鏈結串列進行排序？(兩個節點的交換)
- 如何讓鏈結串列的列印順序可正反序列印？

Outline

結構陣列

鏈結串列

- 單向鏈結串列之資料型態
- 單向鏈結串列之基本運算

作業

回家作業 (ex08 member data.c)

使用鏈結串列製作一個 會員資料表

功能

- 輸入'i' 新增節點在串列最後,可輸入姓名, 電話, Email
- 輸入'd'接著輸入姓名, 可將節點中之姓名相同者刪除(假設輸入之姓名不會重覆)
- 輸入'f'接著輸入一個姓名, 可將節點中之姓名相同者印出資料
- 輸入'l' 印出串列所有節點內容並顯示目前人數
- 輸入'q' 離開程式

提示: 使用strcmp來實作 (**string.h**)

```
char str1[80], str2[80];
```

str1 == str2 這種寫法是不行的, 編譯ok但結果會錯 example

```
strcmp(str1, str2)    if(strcmp(ptr->name, input.name)==0 ){.....}
```

第一個字串大於第二個字串回傳正值, 反之回傳負值。相等則為0

※bonus:可設計插入位置於最前/中間/最後

<https://jgirl.ddns.net/problem/0/2112>

Visual C++ 的問題

```
'strcpy': This function or variable may be unsafe.  
Consider using strcpy_s instead. To disable deprecation,  
use _CRT_SECURE_NO_WARNINGS. See online help for details.
```

在 ProjectProperties -> Configuration Properties -> C/C++ -> Preprocessor -> Preprocessor Definitions

加入這兩行

_CRT_SECURE_NO_DEPRECATED

_CRT_NONSTDC_NO_DEPRECATED

組態(C): 作用中 (Debug)

平台(P): 作用中 (Win32)

組態管理員(O)...

通用屬性

架構和參考

組態屬性

一般

偵錯

VC++ 目錄

C/C++

一般

最佳化

前置處理器

程式碼產生

語言

先行編譯標頭檔

輸出檔

瀏覽資訊

進階

所有選項

命令列

▶ 連結器

▶ 資訊清單工具

▶ XML 文件產生器

▶ 瀏覽資訊

▶ 建置事件

▶ 自訂建置步驟

▶ 程式碼分析

前置處理器定義

WIN32;_DEBUG;_CONSOLE;%(PreprocessorDefinitions)

取消前置處理器的定義

取消所有前置處理器的定義

否

忽略標準的 Include 路徑

否

前置處理器定義

?

✖

```
WIN32
_DEBUG
_CONSOLE
_CRT_SECURE_NO_DEPRECATED
_CRT_NONSTDC_NO_DEPRECATED
```

繼承值:

```
_UNICODE
UNICODE
```

☒ 從父代或專案預設值繼承①

巨集(M) >>

確定

取消

前置處理器定義

為原始程式檔定義前置處理符號。

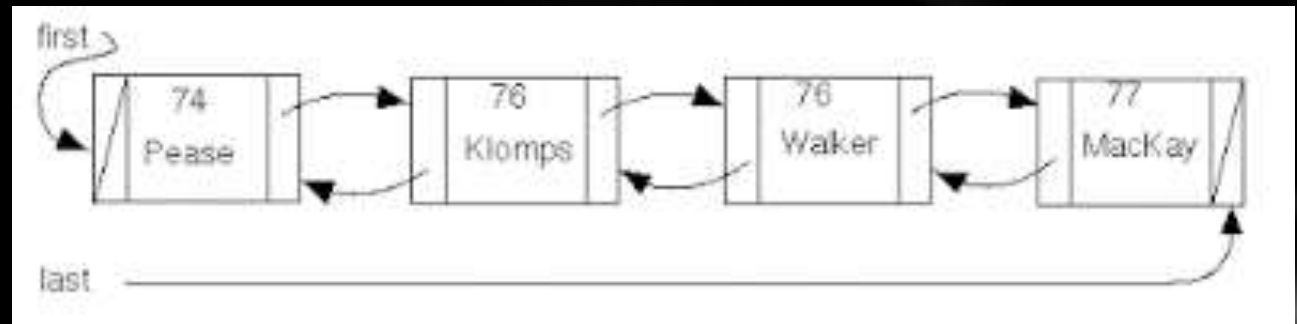
確定

取消

套用(A)

延伸閱讀

- 雙向鏈結串列
- 環狀鏈結串列



Circular, singly linked list:



Circular, doubly linked list:

