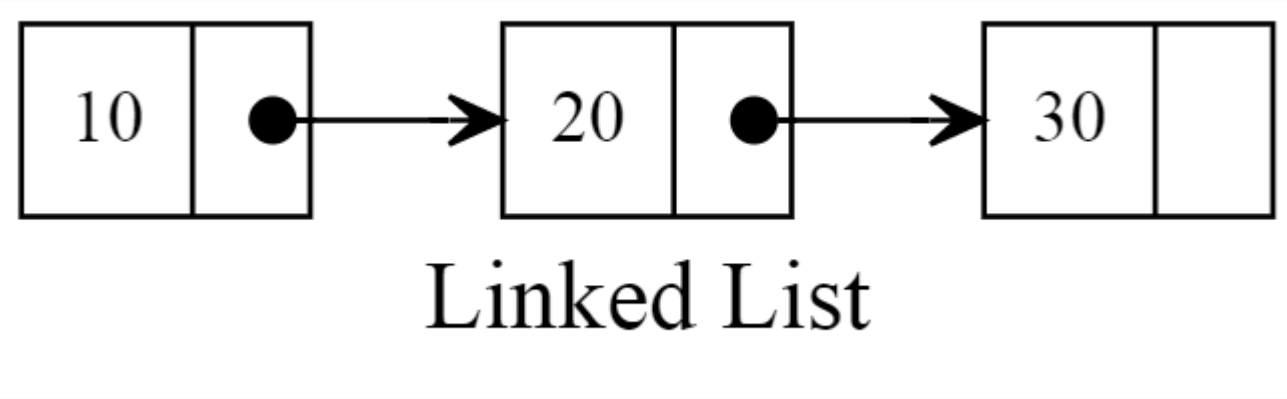
鏈結串列

資訊二甲



https://dotblogs.com.tw/ibllcwhchiu/series/1?qq=%255B%2520C%E8%AA%9E%E8%A8%80%E7%94%9F%E6%B4%BB%E8%A8%98%E4%BA%8B%2520%255D%2520%2520%E8%B3%87%E6%96%99%E7%B5%90%E6%A7%8B%2520Linked%2520List

Step 1
建立節點

利用 typedef ，之後宣告就直接打命名完過後的名稱即可 ~

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

Step 2
新增節點

會有三種情況：

1. 插在前方
2. 插在尾巴
3. 插在中間

```
NODE* InsertNode(NODE* pHead,
NODE* ptr, int data) {
// 先宣告一個新增的節點
NODE* NewNode =
(NODE*)malloc(sizeof(NODE));
NewNode->data = data;
NewNode->next = NULL;
```

Step 2-1 插在前方

會有三種情況：

1. 插在前方
2. 插在尾巴
3. 插在中間

```
//插在Head之前，ptr為NULL
if( ptr == NULL ) {
    NewNode->next = pHead;
    pHead = NewNode;
}
```

Step 2-2 插在尾巴

會有三種情況：

1. 插在前方
2. 插在尾巴
3. 插在中間

```
//插在尾巴之後，ptr的下個節點為NULL
else if( ptr->next == NULL) {
    ptr->next = NewNode;
}
```

Step 2-3 插在中間

會有三種情況：

1. 插在前方
2. 插在尾巴
3. 插在中間

```
//插在中間，讓前面指向New，讓New指向後面
else {
    NewNode->next = ptr->next;
    ptr->next = NewNode;
}
return pHead;
//因為頭指標可能會變，回傳頭指標
}
```

Step 3
刪除節點

也會有三種情況：

1. 刪除前方
2. 刪除尾巴
3. 刪除中間

```
NODE* DeleteNode(NODE* pHead,
NODE* ptr) {

NODE* PreNode = pHead;

//宣告一個pre指標指向欲刪除節
點的前一節點
```

Step 3-1 刪除前方

也會有三種情況：

1. 刪除前方
2. 刪除尾巴
3. 刪除中間

```
//刪除頭指標
if( ptr == pHead ) {
    pHead = pHead->next;
}
```

Step 3-2 刪除尾巴

也會有三種情況：

1. 刪除前方
2. 刪除尾巴
3. 刪除中間

```
else {
    //將PreNode移動至ptr的前一個節點
    while(PreNode->next != ptr)
        PreNode = PreNode->next;
    //刪除尾巴節點，將PreNode的下個節點設為NULL
    if( ptr->next = NULL )
        PreNode->next = NULL;
```

Step 3-2 刪除中間

也會有三種情況：

1. 刪除前方
2. 刪除尾巴
3. 刪除中間

```
//刪除中間節點
    else
        PreNode->next = ptr->next;
    }
FreeNode(ptr);
return pHead;
//因為頭指標可能會變，回傳頭指標
}
```

Step 4
釋放記憶體

使用 free() 方法

```
NODE *ptr;
free(ptr);
```

Step 5
尋找節點

```
NODE* FindNode(NODE* ptr, int data) {

//為了不動到原本指標，宣告一個新指標做移動

    NODE* pTempNode = ptr;

    while( pTempNode->next != NULL && pTempNode != data ){

        pTempNode = pTempNode->next;

    }

return pTempNode;

}
```