Lab 07
指標與記憶體管理

授課：ANT 實驗室

簡報製作：Vincent Chi < song374561@chivincent.net >

實驗一：指標（point to）的意義

- 實驗目的
  - 瞭解指標的意義
  - 瞭解指標與陣列的配合與異同
- 實驗內容
  - 定義一函式 void swap (int *a, int *b)
    - 函式能實習 a, b 數值互換

實驗一：指標（point to）的意義

```
int a = 5 , b = 10 ;
```

	a	b
value	5	10
address	0x04	0x08

儲存一個數值的我們稱為 **變數**

儲存某個變數的位址我們稱為 **指標**

```
int *c = &a , *d = &b ;
```

	c	d
value	0x04	0x08
address	0x0C	0x10

實驗一：指標（point to）的意義

```
void swap (int c , int d){
    int temp=c;
    c=d;
    d=temp;
}
```

swap 目的是要交換兩個變數數值

結果：a = 5 ， b = 10

```
int main(){
        int a=5,b=10;
        swap (a,b);
}
```

做完swap後 a = ? 、 b = ?

實驗一：指標（point to）的意義

```
void swap (int* c , int* d){
    int temp= *c;
    *c=*d;
    *d=temp;
}
```

宣告的 * 為 指標變數的意思 ；
內容的 * 為 提取指向位址的值

結果：a = 10 ， b = 5

```
int main(){
        int a=5,b=10;
        swap (&a, &b);
}
```

做完swap後 a = ? 、 b = ?

實驗二：取址（address of）的意義

實驗目的

瞭解 & 不唸 “and”，在取址時應唸做 “address of”

瞭解 & 運算元的意義

瞭解 & 搭配 * 的技巧

實驗二：取址（address of）的意義

```
int a = 10 ;
a   >>  10
&a >>  0x04
*a  >>  error
```

```
int* b = &a ;
b   >>  0x04
&b >>  0x18
*b  >>  10
```

& = 取址 (address of)
* = 取該位址的值 (value of)

```
*b++     >> ?
*(b++)  >> ?
(*b)++  >> ?
*++b     >> ?
*(++b)  >> ?
++*b     >> ?
++(*b)  >> ?
```

實驗二：取址（address of）的意義

```
char s[] = “123456”;
char* p = s;
```

Address	s	*p指向	*p取值
0x00	1	0x00	1
0x04	2		
0x08	3		
0x0C	4		
0x10	5		
0x14	6		

```
*p++    >> ?
*(p++)  >> ?
(*p)++  >> ?
*++p    >> ?
*(++p)  >> ?
++*p    >> ?
++(*p)  >> ?
```

實驗三：「指標的指標」的意義

實驗目的

瞭解 **p 不叫「雙指標」，叫做「指標的指標」

瞭解 Call By Reference 的誤會來自於指標的指標的 Call By Value

如何用指標的指標去操作傳入函式中的值

```
int  a = 10 ;

int  *b  = &a ;

int  **c = &b ;
```

Address	a	*b	**c
0x00	10		
0x04		0x00	
0x08			0x04

實驗四：函式指標（Function Pointer）

實驗目的

瞭解 Function Pointer 的定義方法與實作方法

瞭解 Function Pointer Pointer 的用法

指向 Function Pointer Array 的 Pointer

實驗四：函式指標（Function Pointer）

宣告fp 為一個參數為int，回傳值也是int的函數指標

```
int  (*fp) (int) ;
int   a1 (int){}
int   b2 (float){}
char  c3 (int){}
```

```
fp = a1 ;
(*fp)(999);
相當於
a1(999);
```

```
fp = a1 ;
fp = b2 ; //error
fp = c3 ; //error
```

b2 參數型別不符
c3 回傳值型別不符

實驗五：記憶體管理

實驗目的

瞭解 malloc()、calloc()、realloc() 及 free()

瞭解 realloc() 的陷阱與因應之道

實驗五：記憶體管理

```
int  arr[100] ;
int  *arr  =  malloc(100 * sizeof(int));  //100個Int空間
int  *arr  =  calloc(100 * sizeof(int));   //100個Int空間(初始化)
arr  =  realloc(arr , sizeof(int) * 200 ) ; //申請arr擴充到200空間
```

注意：如果 realloc 失敗的話會回傳 null 回來。 realloc不做初始化

```
free (arr) ;  //釋放空間
```

選讀一：Call By Value 與 Call By Reference

「C 語言沒有 Call By Reference」

「C 語言沒有 Call By Reference」

「C 語言沒有 Call By Reference」

很重要所以講三次

所有的函式傳遞都涉及數值（Value）

參考資料

《你所不知道的 C 語言》 系列講座　作者：jserv　https://hackmd.io/s/HyBPr9WGl

作業一：實作 Linked List

作業內容

引入 linkedlist.h，練習使用自定義結構 List

實現 Push(List *list) 與 Pop(List *list)，於指定 Linked List 上針對最後一個元素進行新增與移除

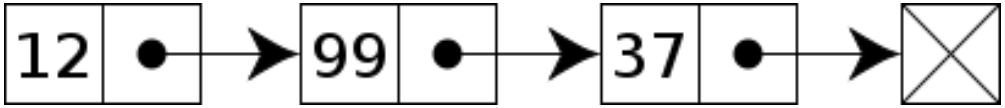
作業繳交內容

.c 檔 、 .pdf 檔

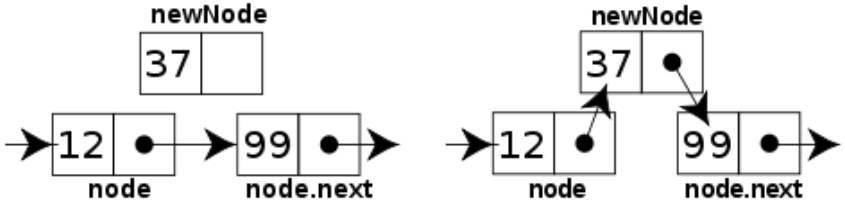
加分項目

實作 Insert(List *list, int n, List item)、Remove(List *list, int n) 兩個函式，可以針對 Linked List 的指定項目進行新增與刪除，被移除的項目記得 free 乾淨，否則酌酌給分

作業一：實作 Linked List



插入新節點



移除節點