C/C++基礎程式設計

函式 (Function)

張傑帆
CSIE NTU

我們必須讓小事也令人難忘。
We've got to make the small things unforgettable. -Steve Jobs

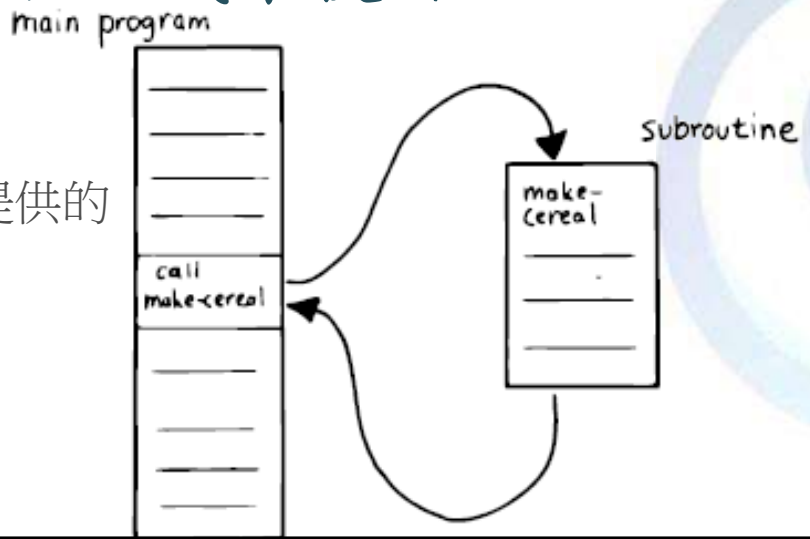
課程大綱

- 函式概論
- 變數類型-全/區域變數
- 函式中以指標當參數
- 傳遞陣列參數
- 把程式拆成多個檔案

函式 (Function)

- 包函許多程式碼的一行程式 (用來代表某種功能)
- 當程式碼太多且會重覆出現時，可以將部份程式碼抽離主程式，寫成一段函式，有需要用到時再去呼叫它
- 之前我們已經使用過C語言提供的函式，現在我們要練習自己寫個函式來使用

內建函式：
main() /printf()/scanf()
也是函式，這些都是系統提供的



函式的宣告

- 宣告一個函式並寫裡面的程式
- 要訣：
  - 取個函式名稱用來代表某個功能 → **函式名稱**
  - 這個功能需要給他什麼資料才能執行 → **參數**
  - 開始寫函式內的程式碼 → **{程式碼}**
  - 執行完後會回傳什麼資料給呼叫函式的程式 → **回傳值**
  - 函式內宣告的**變數是獨立的**，只有在函式內可以使用

回傳值資料型態 (不需回傳可用void)　　傳入函式值資料型態

- **資料型態 函式名稱(資料型態 參數1,資料型態 參數2, …,資料型態 參數n)**

```
{
    程式碼;
    …
    return 回傳值;
}
```

```
int func ( int var1, int var2)
{
    int var3 = var1+var2;
    return var3;
}
```

```
int ans = func(10, 20);
//ans => 30
```

寫好的函式要放在哪裡呢？

- 由於使用者自定函式是無中生有的，因此必須在使用前先定義該函式，此即為該函式的主體。

- 函式主體的位置通常撰寫在 #include和main()主函式的之間，即main()主函式的前面，也允許放在main()主函式的後面，若使用後者就必須在main()主函式前面先宣告函式的原型，以告知編譯器此自定函式在程式中有定義。

```
int func ( int var1, int var2)
{
    int var3 = var1+var2;
    return var3;
}
```

方法一

宣告函式 1 `int func ( int, int );`

主程式-main()

```
int ans = func(10, 20);
//ans => 30
```

函式 1

```
int func ( int var1, int var2)
{
    int var3 = var1+var2;
    return var3;
}
```

方法二

函式 1

```
int func ( int var1, int var2)
{
    int var3 = var1+var2;
    return var3;
}
```

主程式-main()

```
int ans = func(10, 20);
//ans => 30
```

一. 函式原型宣告語法

語法

[傳回值型別] 函式名稱 (資料型別 1, 資料型別 2, …資料型別 n 引數 n);

說明

```
int func ( int, int );
```

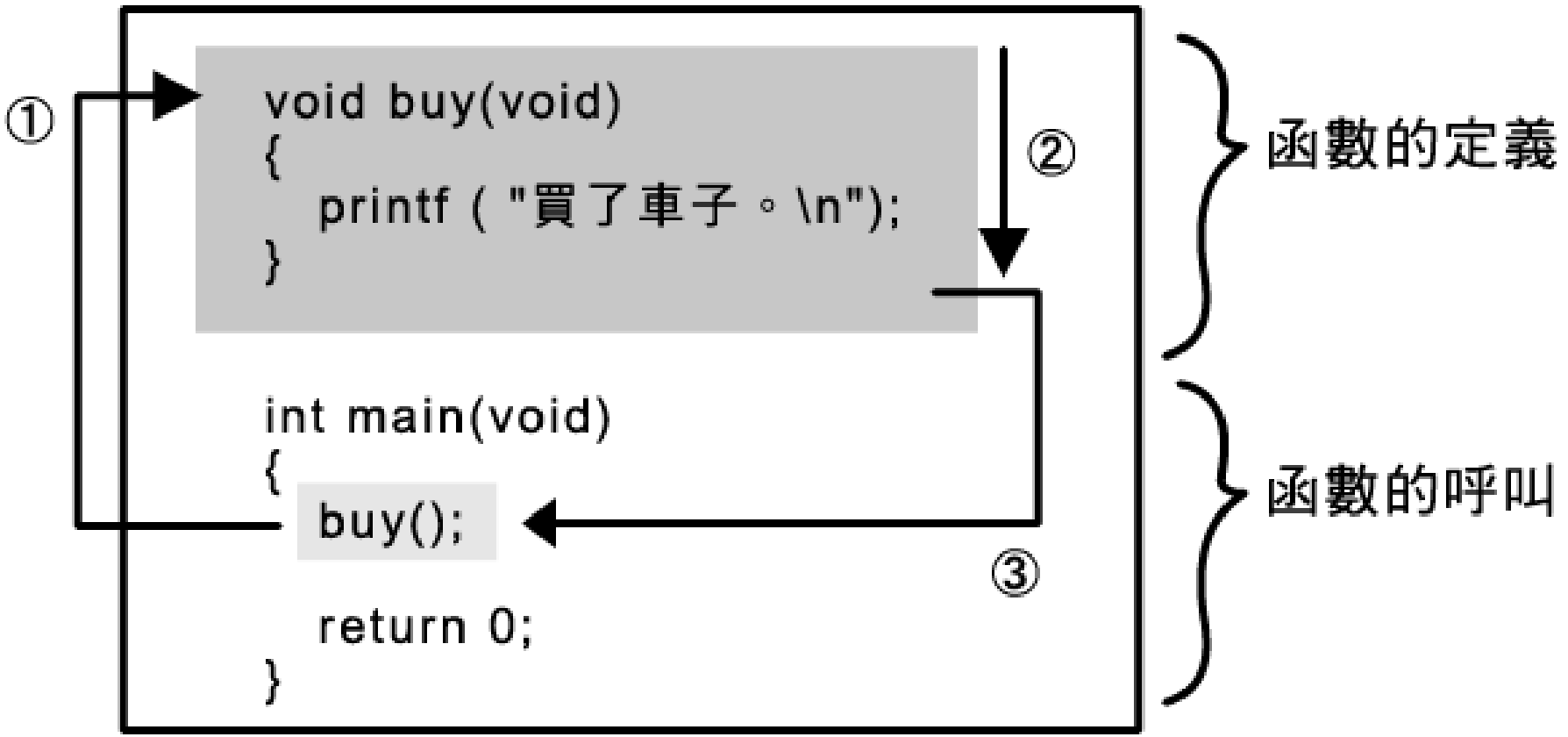
```
int func ( int var1, int var2)
{
    int var3 = var1+var2;
    return var3;
}
```

1. double cal(int);

   宣告 cal 函式，此函式可傳回 double 浮點數資料型別的資料，呼叫時必須傳入一個 int 整數資料型別的資料。

2. int add(int, int);

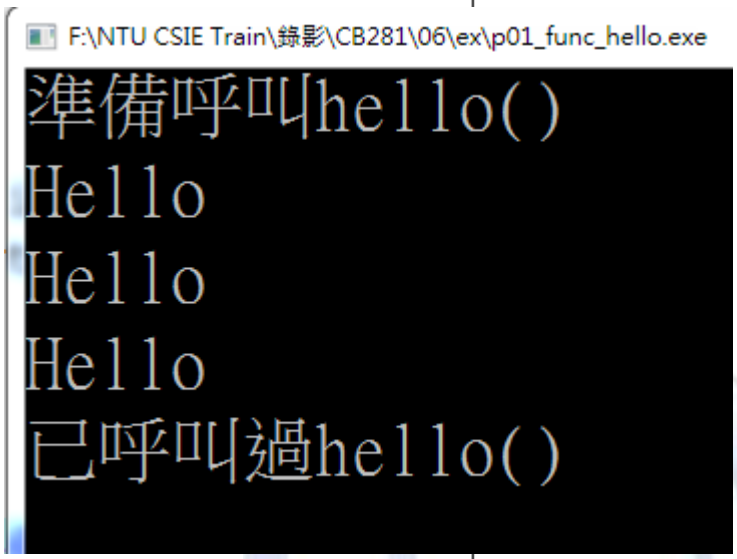
   宣告 add 函式，此函式可傳回 int 整數資料型別的資料，呼叫時必須傳入兩個 int 整數資料型別的資料。


①
void buy(void)
{
printf ("買了車子。\n");
}
②
int main(void)
{
buy();
③
return 0;
}
函數的定義
函數的呼叫

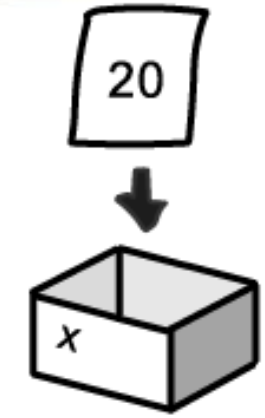
函式的使用練習

- 宣告完的函式即可在其他函式或主程式main中呼叫並使用
- 下面hello為一個不需參數也不會回傳資料的函式：

```
#include <stdio.h>
void hello();  // hello函式的宣告

int main()
{
    printf("準備呼叫hello()\n");
    hello(); // 呼叫hello函式
    printf("已呼叫過hello()\n");
    return 0;
}

void hello() // hello函式的程式碼 (void代表不回傳資料)
{
    printf("Hello\n");
    printf("Hello\n");
    printf("Hello\n");
}
```



example



參數（parameter）

```
void buy(int x)
{
   printf ("買了%d萬元的車子\n", x) ;
}

int main(void)
{

   buy(20);


}
```

20

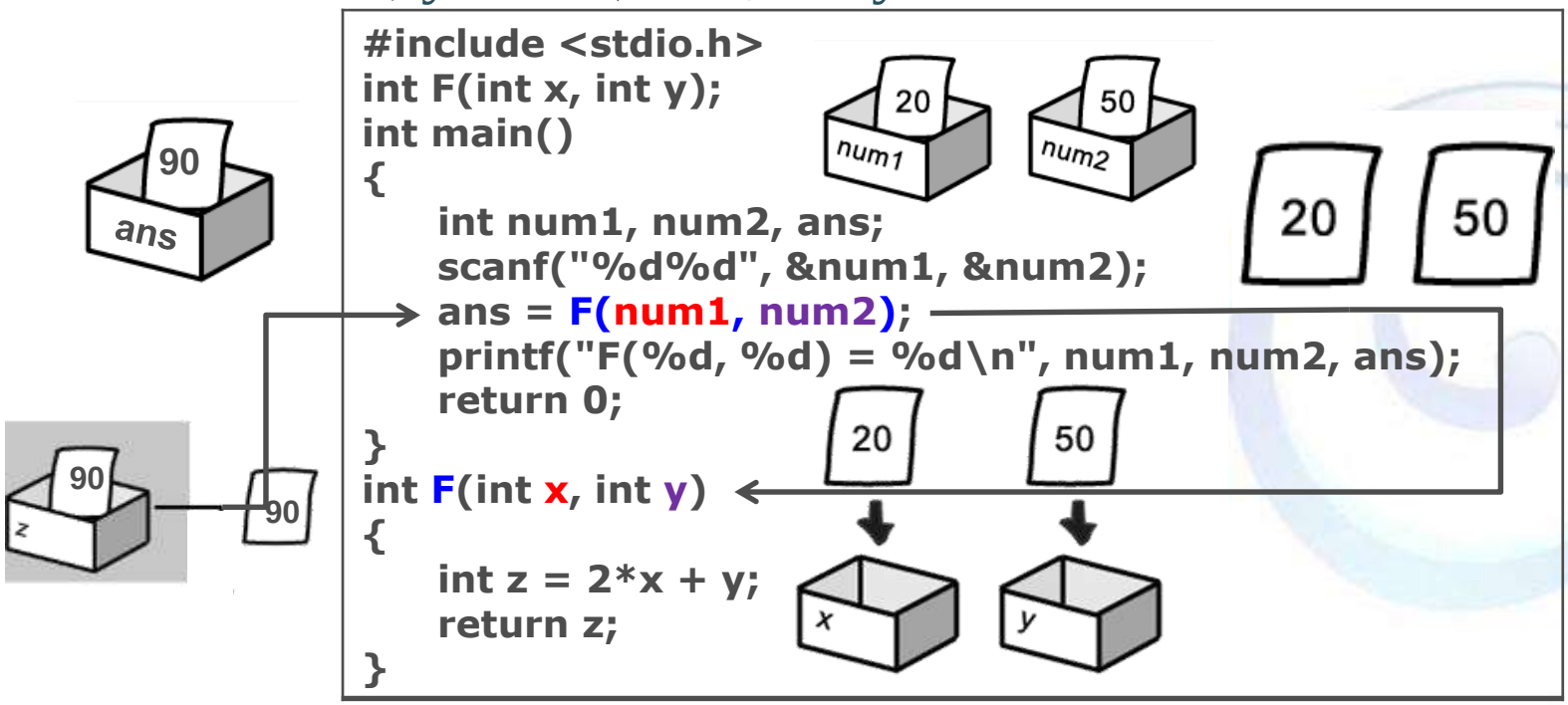
20

引數（argument）

example

函式的使用: 參數與回傳值

- 要訣：資料傳遞需要先儲存！
  - 參數：函式宣告儲存單元將傳入的值儲存
  - 回傳值：函式將結果回傳給呼叫它的地方儲存
- 範例：使用函式計算數學公式 F(x, y) = 2x + y
  - F: 給定x, y兩值即可得2x+y之結果



函式的參數-傳值呼叫(Call by value)

- 呼叫用的引數內容會被copy到函式用來接收參數的變數中，也就是說，呼叫時要傳入的引數，和函式中接收用的參數，事實上是兩個不同的變數
- 所以函式中改變參數數值時，原來呼叫處的引數值並不會改變
- 可以想像說，函式中的參數會宣告成一個新的變數，而它的初值會在呼叫時被設定成傳入的數值
- 執行程式後可以發現，每個變數的記憶體位址都不一樣，所以當然每個變數都是各自獨立的

函式的參數: 易混淆的例子

- 範例

```
#include <stdio.h>
void func(int i);

int main()
{
    // 這裏的變數跟上面都沒有關係
    int a=3;
    int b=2;
    int c=4;
    func(c);
    printf("%d %d %d\n",a,b,c);
    // 有沒有發現, 執行func的c=5後, 這裏的c仍然為4
    return 0;
}
void func(int c)
{
    int a=2;
    int b=3;
    c=5;
    printf("%d %d %d\n",a,b,c);
}
```

https://goo.gl/69qUtu

在函式中結束程式: exit(0);

- 在函式中如果要強制結束程式可以使用exit函式

```
#include <stdio.h>
#include <stdlib.h>
void func();

int main()
{
    func();
    printf("這行不會印出\n");
    return 0;
}
void func()
{
    printf("呼叫exit函式!\n");
    exit(0);
}
```

練習

- 寫一個計算1+2+3…+n的函式
  - int sum1(int n);
    https://jgirl.ddns.net/problem/0/1061
  - int fact(int n);
    https://jgirl.ddns.net/problem/0/1064

課程大綱

- 函式概論
- 變數類型 - 全/區域變數
- 函式中以指標當參數
- 傳遞陣列參數
- 把程式拆成多個檔案

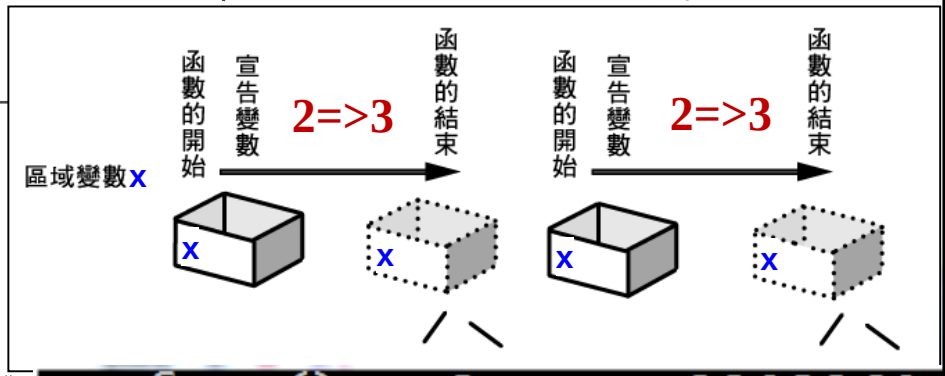
全域變數與區域變數

- C語言將函式內所宣告的變數稱為區域變數(Local Variable)，此類變數的有效範圍僅在該函式內，離開該函式便由記憶體中釋放掉，下次呼叫該函式時再重新配置記憶體給該函式使用。

- C語言另外提供一種變數可供多個函式共同使用，變數的有效時間一直到程式結束為止，我們將此類的變數稱為「全域變數」(Global variable)。

局部/區域變數 (Local Variable)

- 自動變數只在它所定義的**區塊內有效**。只要在變數所屬的區塊結構內執行，該變數的資料是有效而正確的。
- 當程式執行**離開了該區塊**，所有於區塊內定義的自動變數就**不存在了**。




```
#include <stdio.h>
void func();
int main()
{
    int x=1;
    func();
    func();
    printf("%d\n",x);
    return 0;
}
```

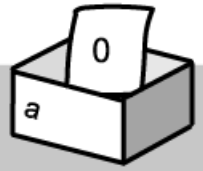
```
void func()
{
    int x=2;
    x = x + 1;
    printf("%d\n",x);
}
```

```
In func() x=3, mem=2293260
In func() x=3, mem=2293260
In main() x=1, mem=2293324
```

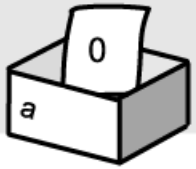
https://goo.gl/oq7JPh

區域變數的名稱可以重複使用

```
void func(void)
{
   int a = 0;
    a++;
}
```




```
int main(void)
{
   int a = 0;
   a++;
}
```



在不同函數內被宣告的區域變數
就代表是2個不同的變數

```
int a = 0;

void func (void)
{
   int c = 2;

}

int main (void)
{
   int b = 1;

   func ();

   return 0;
}
```

全域變數a的生存空間

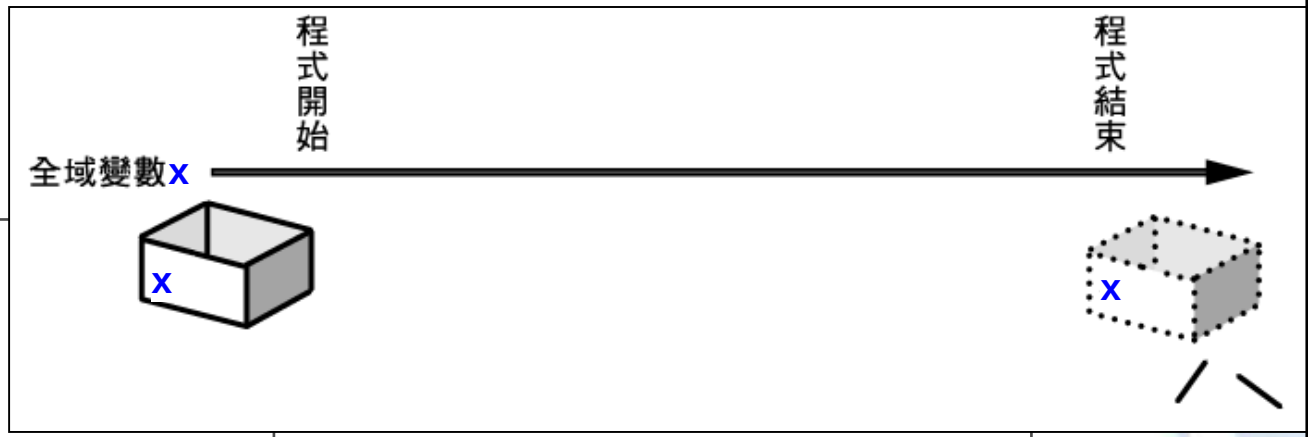
區域變數c的生存空間

區域變數b的生存空間

```
int a = 0;                    全域變數

void func(void)
{
   int c = 2;                 區域變數
    . . .
}

int main(void)
{

   int b = 1;                 區域變數
    . . .
}
```

全域變數 (Global Variable)

- 全域變數的**有效範圍**不是區域性，而**是整體性**
- 變數定義在任何函數的外面，表示可以被其他函數所共用 。

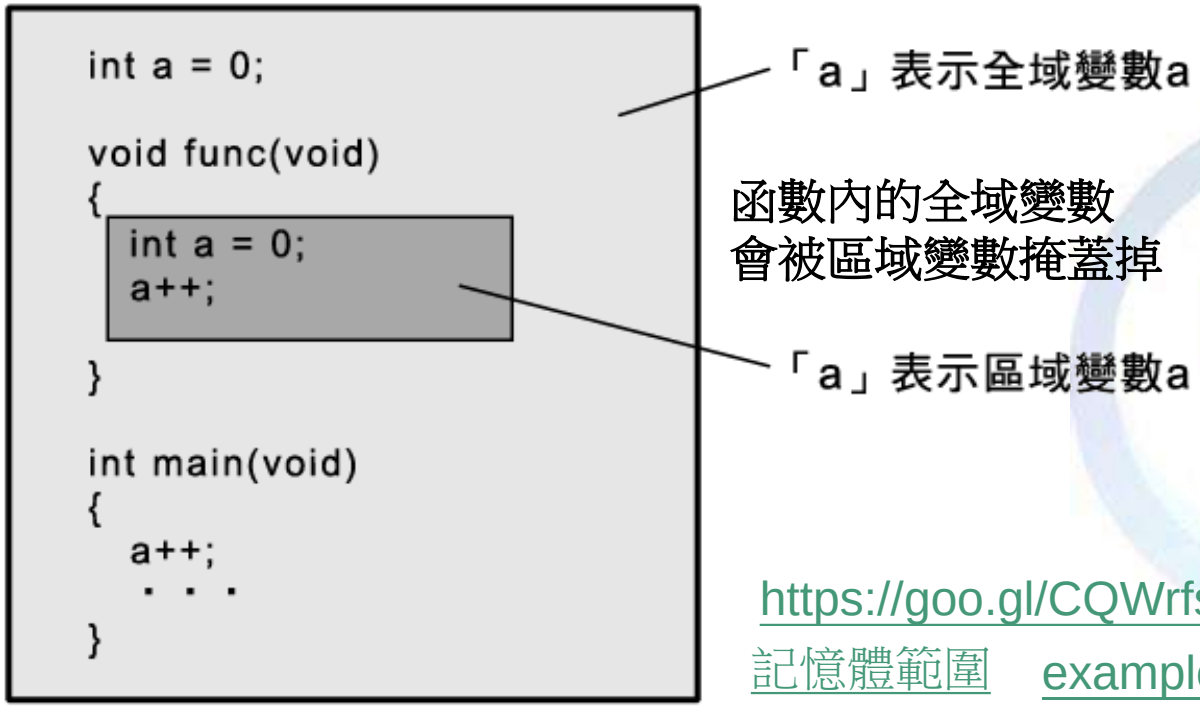



```
#include <stdio.h>
int x = 5;
void func();
int main()
{
    func();
    func();
    printf("%d\n",x);
    return 0;
}
```

```
void func()
{
    x = x + 1;
    printf("%d\n",x);
}
```

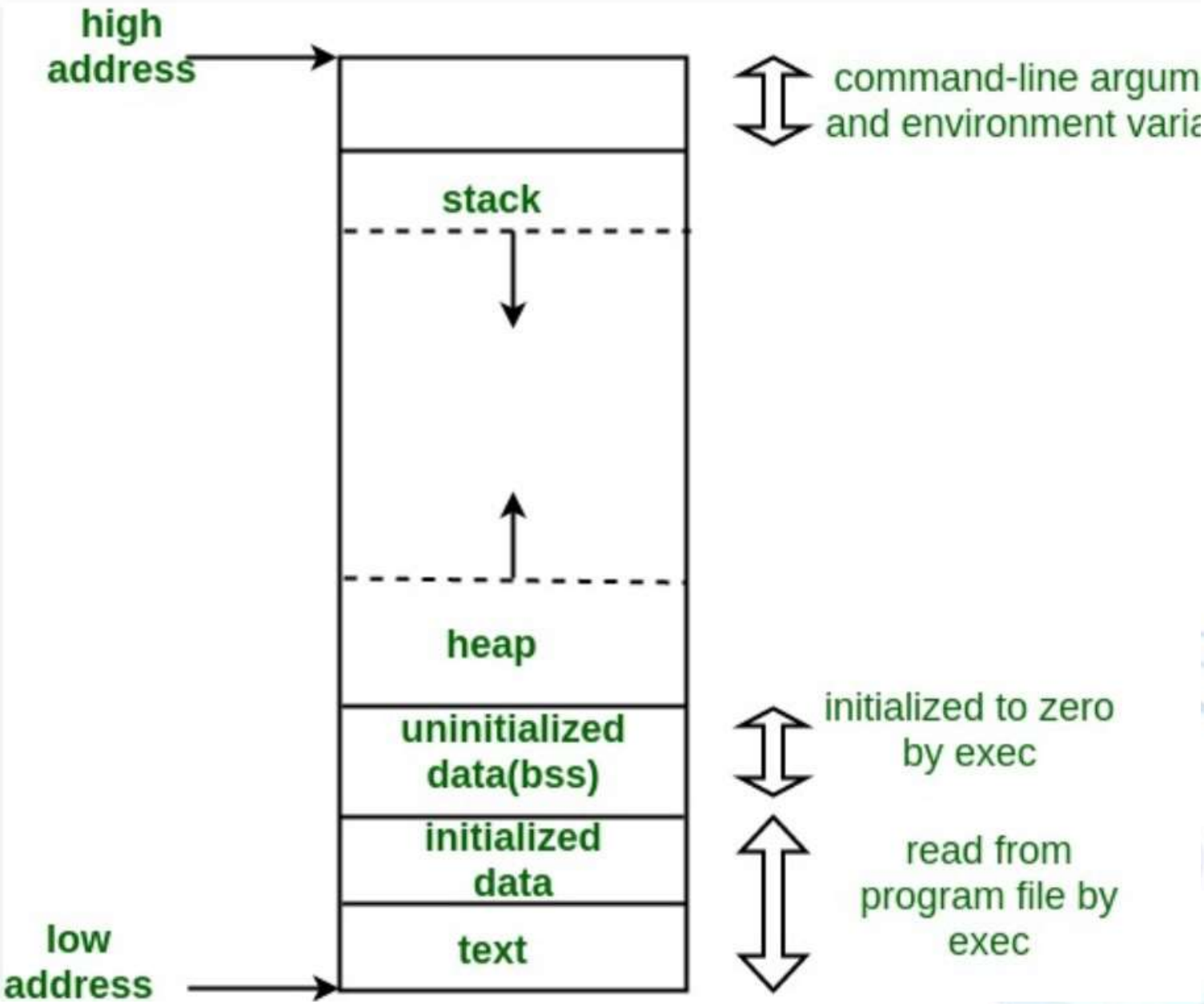
https://goo.gl/eWidFQ
千萬不要這樣子寫程式

- 程式中區域變數與全域變數的**名稱相同**，當存取函式內的變數時會以**區域變數為優先**，使用時最好注意，建議全域變數**最好不要**和區域變數的**名稱重複**，以免參用時造成混淆。



https://goo.gl/CQWrfs

記憶體範圍　example



https://medium.com/@nickteixeira/stack-vs-heap-whats-the-difference-and-why-should-i-care-5abc78da1a88

小練習

- 排列組合 https://goo.gl/ycKFXb
- 輸入參數的合理性

$$P_n^m = nPm = \frac{n!}{(n-m)!} = n(n-1)(n-2)\dots(n-m+1)$$

- 延申閱讀

$$C_n^m = nCm = \binom{n}{m} = \frac{nPm}{m!} = \frac{n!}{m!(n-m)!}$$

  - 公式解
  - 遞迴解
    - 增進遞迴效率
  - 內建函式 math.h

課程大綱

- 函式概論
- 變數類別
- 函式中以指標當參數
- 傳遞陣列參數
- 把程式拆成多個檔案

傳址呼叫 call by address

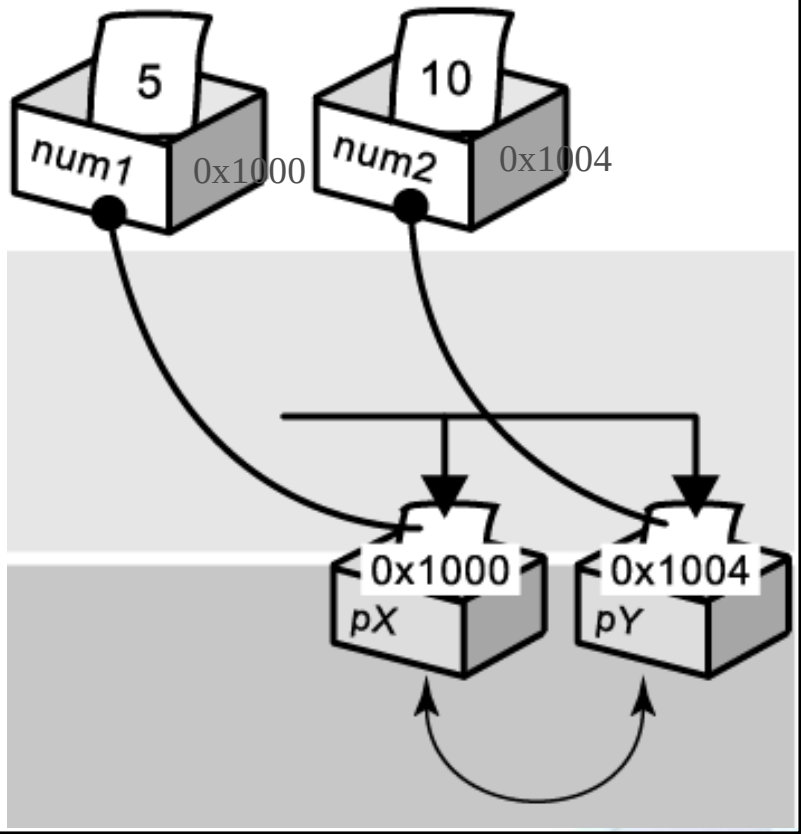
- 所謂的「傳址呼叫」就是函式在做引數傳遞時，呼叫函式中的實引數是將自己本身的記憶體位址傳給被呼叫函式內的參數。
- 希望將回傳一個以上的結果時使用。
- 傳址呼叫的設定方式是：
  - 定義的函式小括號內的虛引數(參數Parameter)必須宣告為指標變數(即在變數前面加上*)
  - 呼叫函式內的實引數(引數Argument)必須傳送變數的記憶體位址(即變數前面加上&)
  - 若傳遞的引數本身即為指標或記憶體位置(陣列)，就不須另外加&

函式中以指標當參數-傳址呼叫

- 要是函式傳參數時，傳的是**變數的位址**，就可以在函式中去改變主程式中變數的內容了
- 範例: swap，兩變數資料交換

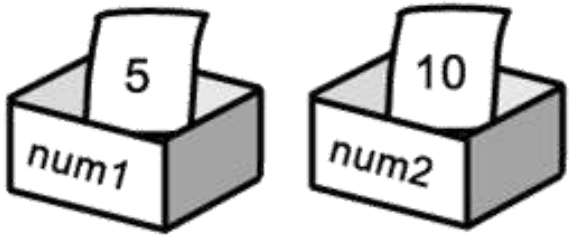
```
#include <stdio.h>
void swap(int *, int *);
int main()
{
    int num1 = 5;
    int num2 = 10;
    swap(&num1, &num2);
    printf("num1=%d\n", num1);
    printf("num2=%d\n", num2);
    return 0;
}
void swap(int *pX, int *pY)
{
    int temp;

    temp = *pX;
    *pX = *pY;
    *pY = temp;
}
```

https://goo.gl/wuxvBk



傳值呼叫 call by value

- 若是想要定義交換引數x與y的swap函數，就不能使用傳值的方式來傳遞引數，因為傳值只是傳遞引數中的值而已，並不會對引數本身產生改變，所以只會交換參數。



pass by reference

cup =

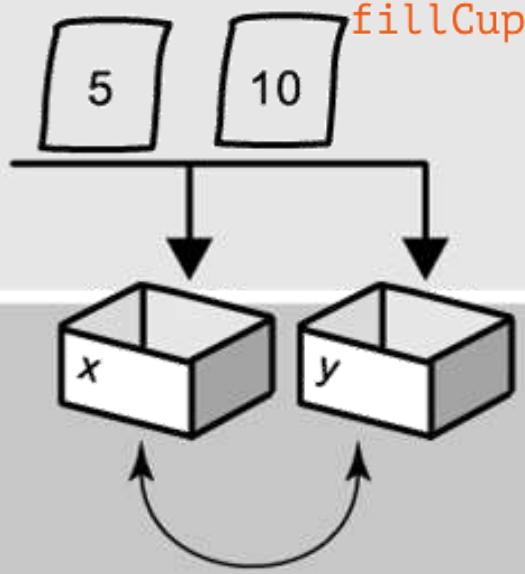
fillCup( )

pass by value

cup =

fillCup( )

```
int main(void)
{
   swap(num1, num2);
}
```




```
void swap(int x, int y)
{
   int tmp;

   tmp = x;
   x = y;
   y = tmp;
}
```

https://goo.gl/gCZcBu

www.penjee.com

swap()函數只有將參數x與y進行交換而已，與實際引數num1及num2無關

小練習

- 請試寫一函式可加總1到n和計算n!
- 並利用指標同時將兩個結果送給原呼叫函式的引數

```
void return2num(int n, int *factRst, int *sumRst);
```

https://jgirl.ddns.net/problem/0/3073

課程大綱

- 函式概論
- 變數類別
- 函式中以指標當參數
- 傳遞陣列參數
- 把程式拆成多個檔案

傳遞陣列參數

- 把陣列當參數來傳遞時，要從記憶體的角度來看，如果傳遞的只是陣列某欄位中的一個變數，那和傳普通變數沒有什麼分別。

  Ex:

  ```
  int ary[5] = {1,2,3,4,5};
  func(ary[0]);
  ```

- 如果傳的是整個陣列，傳遞的東西會是陣列的位址。

  Ex:

  ```
  int ary[5] = {1,2,3,4,5};
  func(ary);
  ```

傳遞陣列參數

- 範例

```
#include <stdio.h>
#include <stdlib.h>
void output(int n, int *p);

int main()
{
    int a[5] = {1,2,3,4,5};
    int b[7] = {1,2,3,4,5,6,7};

    output(5, a);
    output(7, b);
    return 0;
}
```

```
void output(int n, int *p)
{
    int i;
    for(i=0; i<n; i++)
    {
        printf("%d ",p[i]);
    }
    printf("\n");
}
```

https://goo.gl/4L5Qeo

https://goo.gl/3W9NzN

void printAry(int n, int *P);

傳遞陣列參數:平均值計算

- 寫一個回傳同學成績平均分數之函式

```c
#include <stdio.h>
#include <string.h>
double average(int student, int *score);

double average(int student, int *score)
{
    int  i;
    double sum=0;
    double aver;
    for(i=0; i<student; i++)
    {
        sum+=score[i];
    }
    aver = sum/student;
    return aver;
}
```

https://goo.gl/U9wEhY

```c
int main()
{
    int student;
    int *score;
    double aver;
    int i;

    printf("請輸入學生人數: ");
    scanf("%d", &student);
    score = (int *)malloc(sizeof(int) * student);

    for(i=0; i<student; i++)
    {
        printf("學生%d: ", i+1);
        scanf("%d", &score[i]);
    }

    aver = average(student, score);
    printf("平均: %.2lf 分\n", aver);

    free(score);

    return 0;
}
```

練習

- 繼續上一範例，寫一個回傳最高分成績位置之函式
  - void inputAry(int n, int *P);
  - void printAry(int n, int *P);
  - double Avg(int n, int *P);
  - int MaxScore(int n, int *P);

https://goo.gl/xS5qAR

傳遞陣列參數: 字串處理

- 使用upper函式，將字串中小寫英文轉大寫英文

```
#include <stdio.h>
void upper(char *a);

int main()
{
    char str[80];

    gets(str);
    upper(str);

    printf("%s \n", str);
    return 0;
}
```

```
void upper(char *a)
{
    int  i=0;
    while(1)
    {
        if(a[i]>='a' && a[i]<='z')
            a[i]-=32;
        else if(a[i] == '\0')
            break;
        i++;
    }
}
```

https://goo.gl/B99BKE

課程大綱

- 函式概論
- 變數類別
- 函式中以指標當參數
- 傳遞陣列參數
- 把程式拆成多個檔案

把程式拆成多個檔案

- 要訣：
  - 標頭檔.h ← 用來宣告函式名稱 (用來被include)
  - 程式檔.c ← 用來寫函式程式碼 (放到專案一起編譯與連結)

main.c

```c
#include <stdio.h>

#include "hello.h"
// hello()的宣告在這裏面

int main()
{
    printf("準備呼叫hello()\n");
    hello(); // 呼叫hello函式
    printf("已呼叫過hello()\n");
    return 0;
}
```

hello.h

```c
#ifndef HELLO_H
#define HELLO_H

void hello();  // hello函式的宣告

#endif
```

hello.c

```c
#include "hello.h"
#include <stdio.h>

void hello() // hello函式的程式碼
{
    printf("Hello\n");
    printf("Hello\n");
    printf("Hello\n");
}
```

把程式拆成多個檔案

- 練習：
  將下述幾個函式分成**score.h**與**score.c**，並寫一個程式計算班上同學成績並列出**平均**與**最高分**
  - void inputAry(int n, int *P);
  - void printAry(int n, int *P);
  - double Avg(int n, int *P);
  - int MaxScore(int n, int *P);

https://jgirl.ddns.net/problem/0/1063

回家作業-質數判斷程式

- 請寫一函式令其可以判斷傳入參數是否為質數
  - int IsPrime(int num);

    https://goo.gl/xpiDyz
- 是質數的話回傳 1，不是的話回傳 0
- 並將函式分成prime.h、prime.c與main.c

https://jgirl.ddns.net/problem/0/1062

延申閱讀

- **static 靜態區域變數** https://goo.gl/qhvLJz
- **傳遞一維陣列的別種寫法** https://goo.gl/WixBn6
- **函數指標**
- **遞迴函式**
- **Unit test (單元測試)**
  - 在函式上驗證參數與程序的正確性

常用 gcc 編譯指令

- 單檔
  - gcc main.c (沒寫輸出檔名則得到a.exe)
  - gcc main.c -o main.exe
- 多檔多行
  - gcc -c main.c
  - gcc -c HelloWorld.c
  - gcc main.o HelloWorld.o -o hello.exe
- 多檔一行
  - gcc main.c HelloWorld.c -o hello.exe

https://magicjackting.pixnet.net/blog/post/72822288