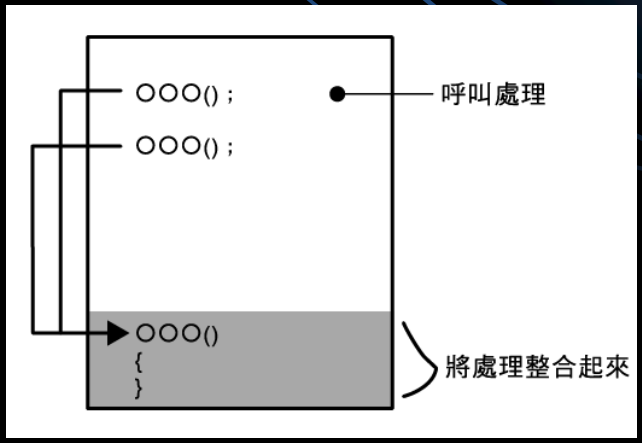
函數

8-1 函數

- 函數（function）
  - 可以整合某些特定的處理。
  - 整合好的處理可以隨時呼叫使用。
  - C語言的程式本身也是一個函數，也就是main()函數。
  - 使用函數可簡化程式。



- 提款的處理
  1. 將提款卡插入自動提款機當中
  2. 輸入個人密碼
  3. 指定提款金額
  4. 領取款項
  5. 確認款項與提款卡

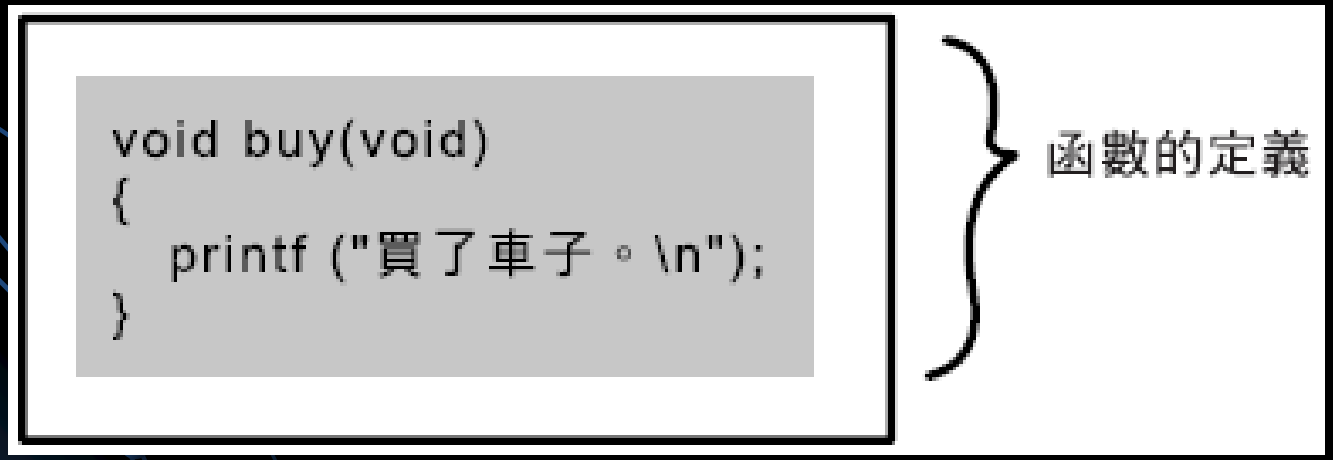
提款處理

8-2 函數的定義與呼叫

- 定義函數的語法：

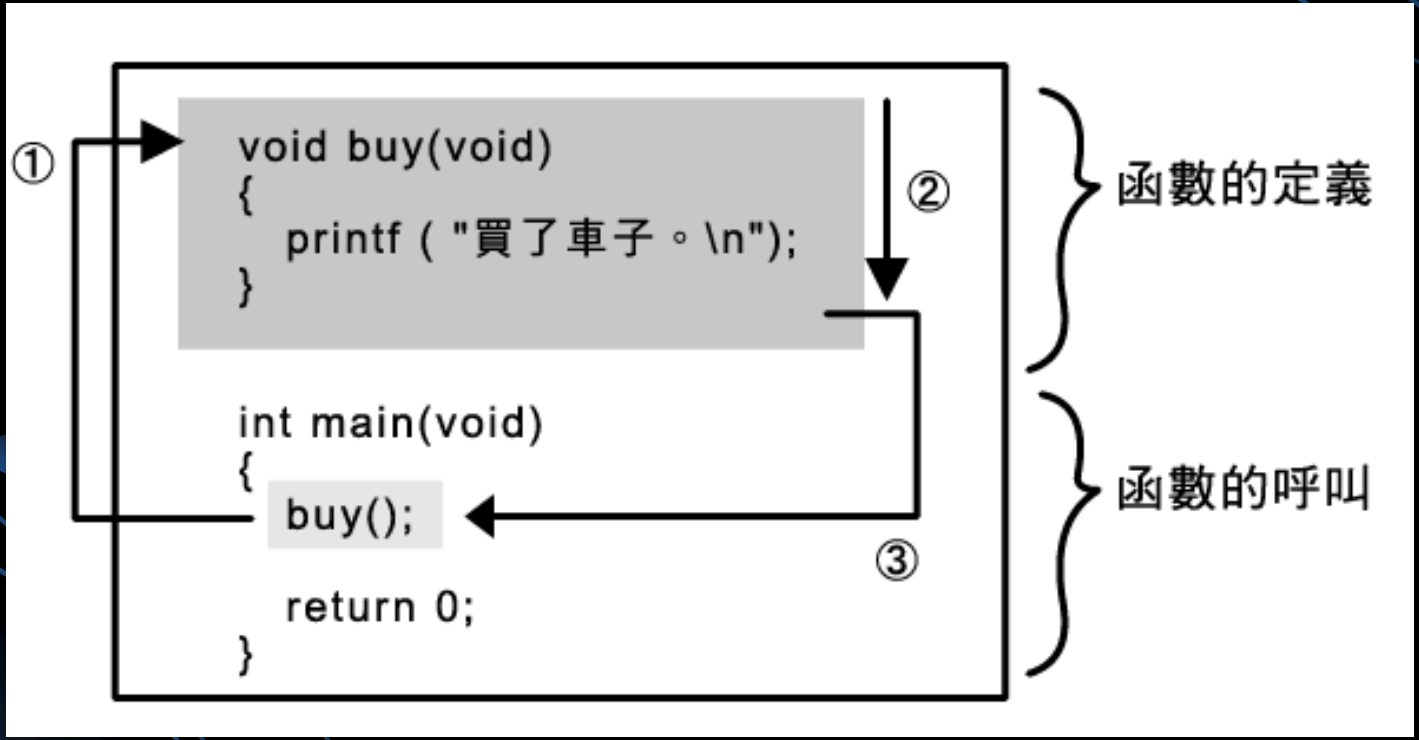
```
傳回值的型態 函數名稱（引數列表）
{
  敘述;
  …
return 運算式;
}
```

- 範例：



- 呼叫函數的語法：
  函數名稱(引數列表);

- 範例：



Sample1.c ▶ 建立基本的函數

```
#include <stdio.h>

/* buy函數的定義 */
void buy(void)
{
    printf("買了車子。\n");
}

/* buy函數的呼叫 */
int main(void)
{
    buy();

    return 0;
}
```

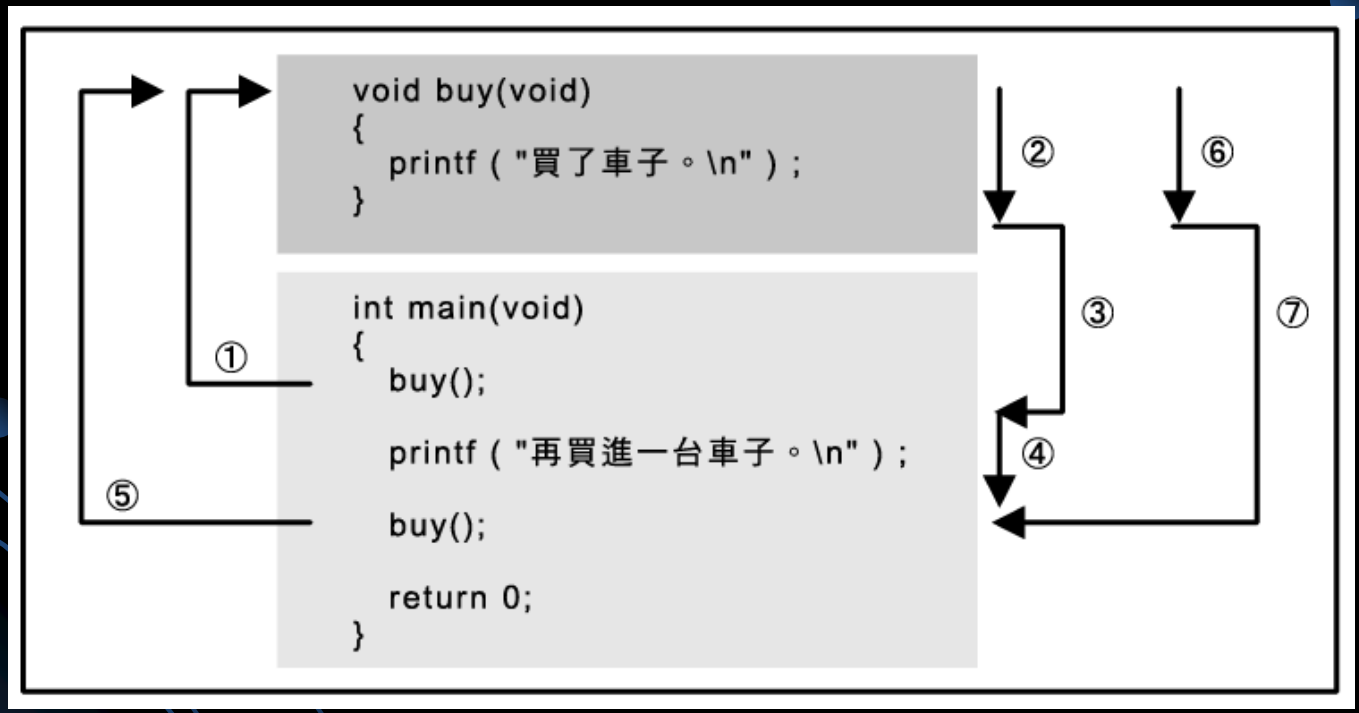
這是 buy()函數的處理內容

在這裡執行 buy()函數的處理

- 使用函數的處理流程爲：

  1.呼叫函數；2.執行函數中的處理；3.回到呼叫處。

- 也可以多次呼叫函數，範例如下：



Sample2.c ▶ 不斷呼叫函數

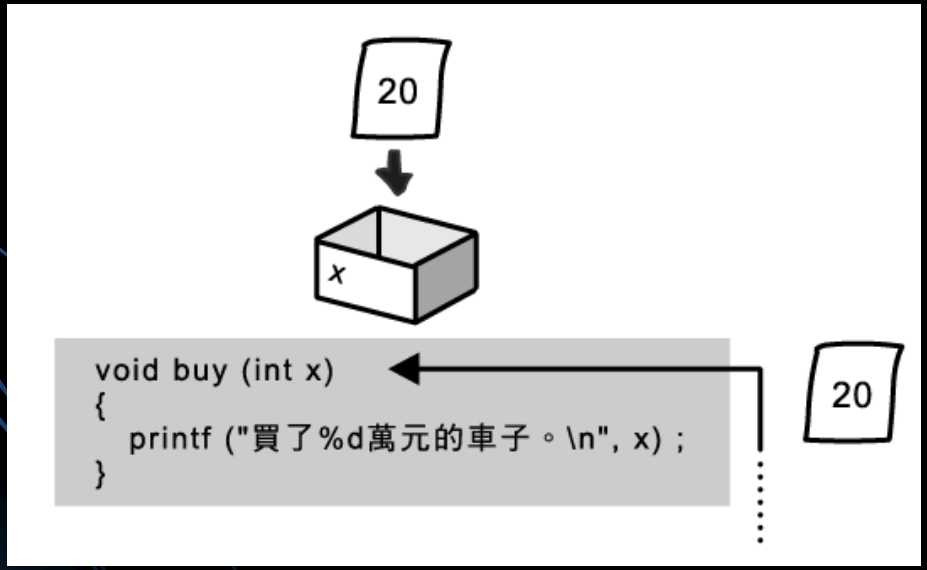
```
#include <stdio.h>

/* buy 函數的定義 */
void buy(void)
{
    printf("買了車子。\n");
}

/* buy 函數的呼叫 */
int main(void)
{
    buy();

    printf("再買進一台車子。\n");

    buy();

    return 0;
}
```

呼叫出 buy()函數

再呼叫一次 buy()函數

8-3 引數

- 使用引數傳遞資料
  - 當要呼叫出函數時，可以進行以下的處理：從呼叫處將某項資料（值）傳遞到函數內、依照這項值來執行相對應之處理。
  - 要傳遞給函數的資料稱爲**引數**（argument）。



- 傳遞引數給函數的範例如下：

```
#include <stdio.h>

/* buy函數的定義 */
void buy(int x)
{
   printf("買了%d萬元的車子。\n", x);
}

/* buy函數的呼叫 */
int main(void)
{
   buy(20);
   buy(50);

   return 0;
}
```

買了20萬元的車子。
買了50萬元的車子。

在這個main()函數當中，會執行以下處理：
第一次呼叫出buy()函數時，傳遞「20」這個值之後再呼叫
第二次呼叫出buy()函數時，傳遞「50」這個值之後再呼叫

Sample3.c ▶ 使用帶有引數的函數

```
#include <stdio.h>

/* buy 函數的定義 */
void buy(int x)
{
    printf("買了 %d 萬元的車子。\n", x);
}

/* buy 函數的呼叫 */
int main(void)
{
    buy(20);
    buy(50);

    return 0;
}
```

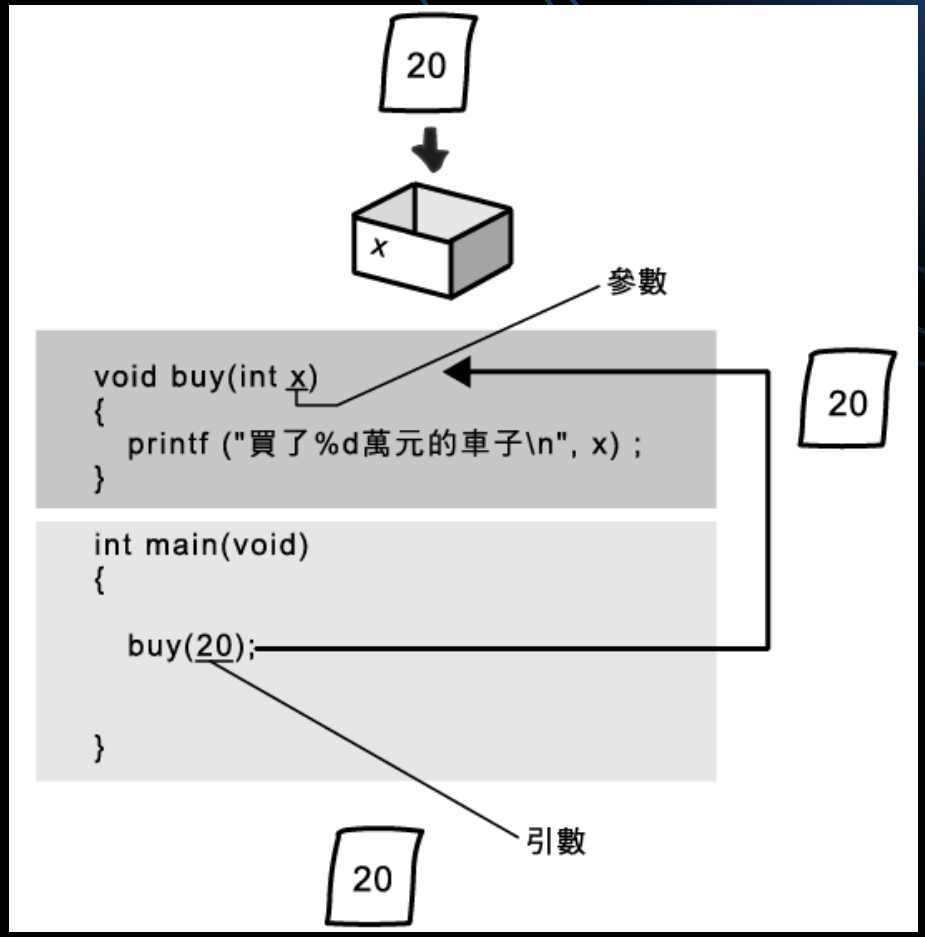
接受值的參數

將所接受的值輸出

傳遞 20 做為引數後進行呼叫

傳遞 50 做為引數後進行呼叫

- 函數本身所定義的引數（變數）稱爲**參數**（parameter）。另一方面，從函數的呼叫來源所傳遞的引數（值）則稱爲**引數**（argument）。



- 所以之前例子的變數x是參數，「20」和「50」則是引數。

- 將從鍵盤輸入的值傳到函數中：

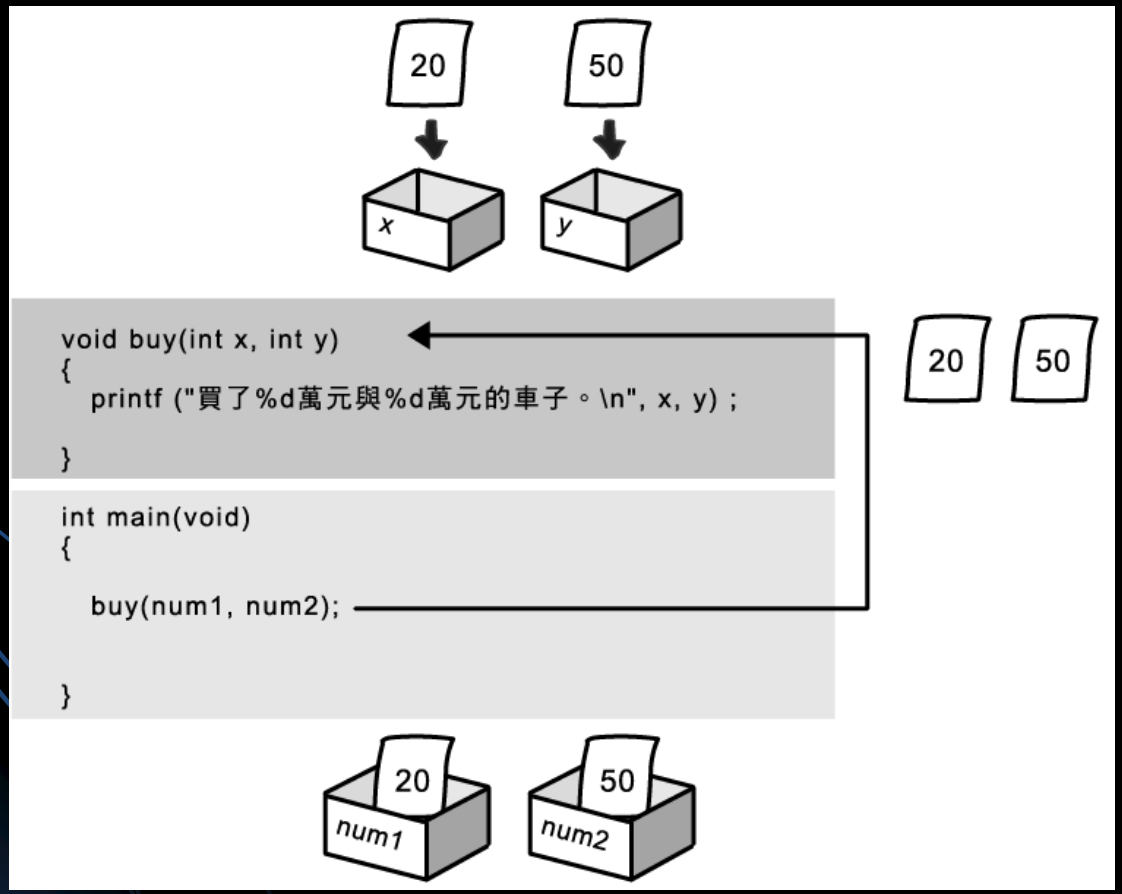
```
#include <stdio.h>
/* buy函數的定義 */
void buy(int x)
{
  printf("買了%d萬元的車子。\n", x);
}

/* buy函數的呼叫 */
int main(void)
{
  int num;
  printf("第1台要買多少錢的車子？ \n");
  scanf("%d", &num);
  buy(num);

  printf("第2台要買多少錢的車子？ \n");
  scanf("%d", &num);
  buy(num);
  return 0;
}
```

在C語言中，傳遞到函數的並不是引數本身，而是儲存在其中的「值」。這種引數的傳遞方法稱爲**傳值**（pass by value）。

這裡使用main()函數當中所準備的變數num（的值），作爲由呼叫來源傳遞到函數的引數。所以會將由鍵盤讀取到num的值傳遞到函數中。

- 函數可以帶有多個引數，但在呼叫時要用逗號(,)來區隔所指定的多個引數，這些引數又叫做「引數列表」，它們會依指定的順序傳遞給參數。

- 範例：



Sample5.c ▶ 使用帶有多個引數的函數

```
#include <stdio.h>

/* buy 函數的定義 */
void buy(int x, int y)
{
    printf("買了 %d 萬元和 %d 萬元的車子。\n", x, y);

}

/* buy 函數的呼叫 */
int main(void)
{
    int num1, num2;

    printf("要買多少錢的車子？\n");
    scanf("%d", &num1);

    printf("要買多少錢的車子？\n");
    scanf("%d", &num2);

    buy(num1, num2);

    return 0;
}
```

帶有 2 個引數的函數

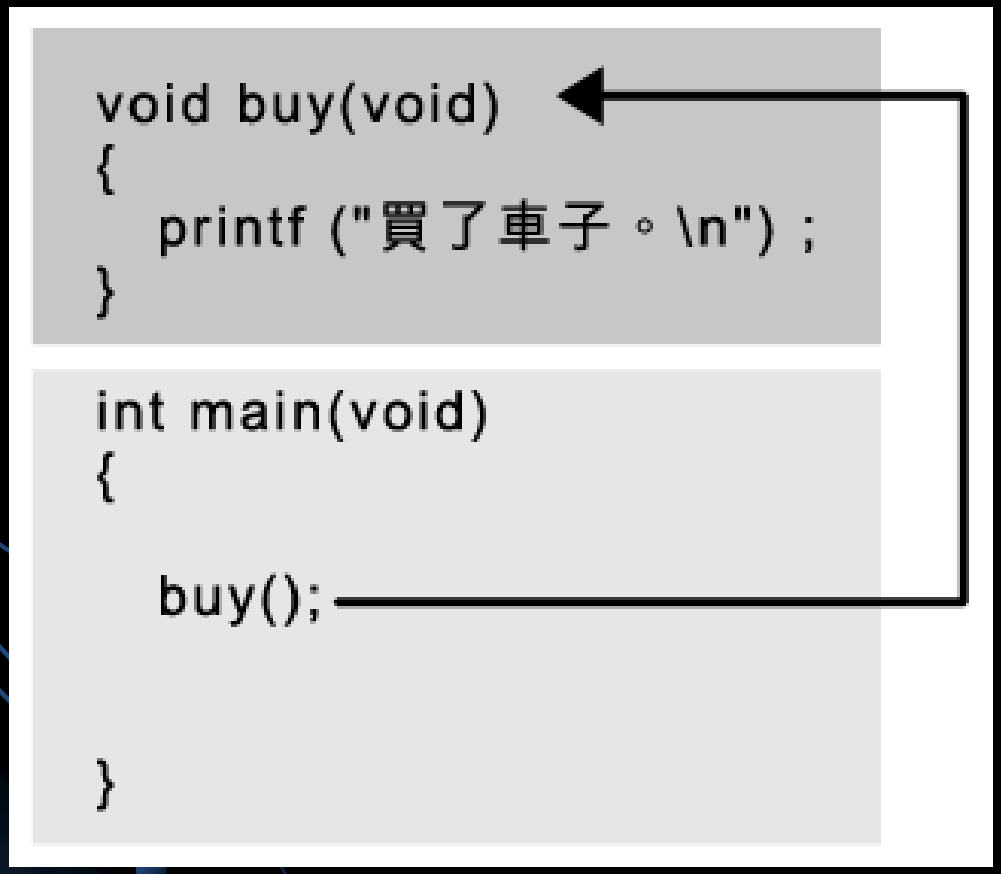
輸出第 1 個引數

輸出第 2 個引數

傳遞 2 個引數

- 定義沒有引數的函數時，可以省略引數型態的指定，或指定爲void型態。

- 範例：

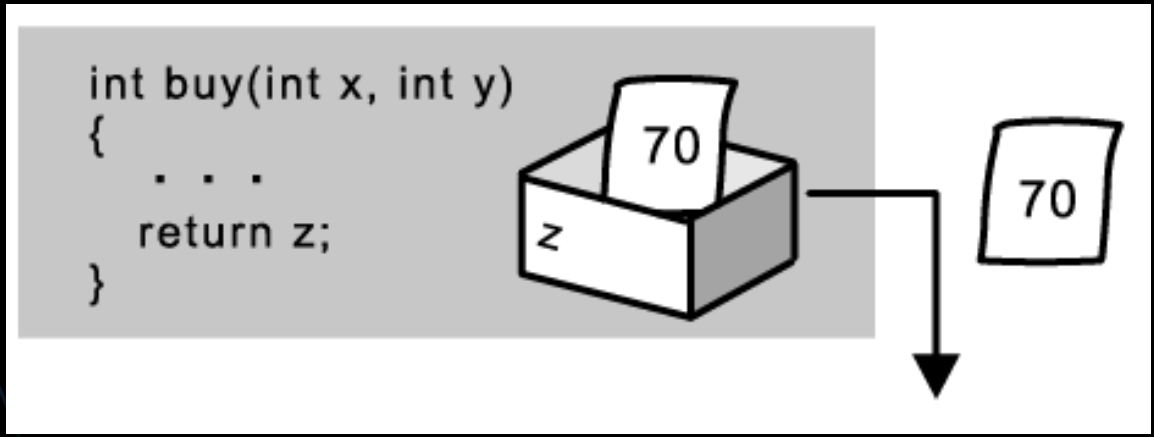


8-4 傳回值

- 理解傳回值的機制
  - 從函數本身傳回特定資訊到函數的呼叫來源處
  - 從函數所傳回的資訊稱爲**傳回值**（return value）。它與可以多重指派的引數不同，傳回值只能有一個。

- 定義函數的語法：

```
傳回值的型態 函數名稱（引數列表）
{
    敘述;
    ...

    return 運算式;
}
```



Sample6.c ▶ 傳回值的函數

```
#include <stdio.h>

/*buy 函數的定義 */
int buy(int x, int y)
{
    int z;

    printf("買了 %d 萬元與 %d 萬元的車子。\n", x, y);

    z = x+y;

    return z;
}

/* buy 函數的呼叫 */
int main(void)
{
    int num1, num2, sum;

    printf("要買多少錢的車子？ \n");
    scanf("%d", &num1);

    printf("要買多少錢的車子？\n");
    scanf("%d", &num2);

    sum = buy(num1, num2);

    printf("合計為 %d 萬元。\n", sum);

    return 0;
}
```

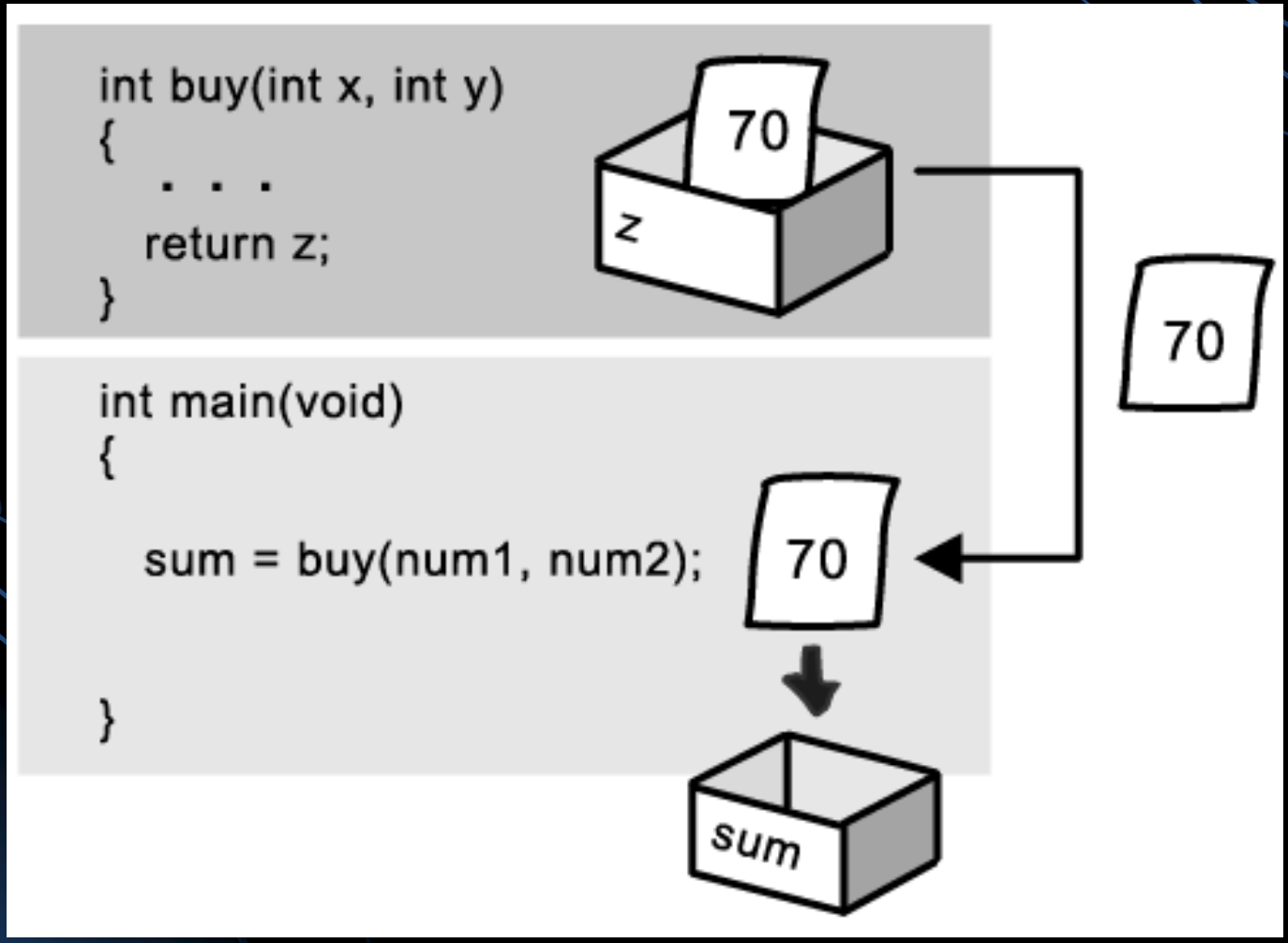
帶有傳回值的函數

將傳回值傳回

呼叫函數，將該傳回值指派給變數 sum

輸出傳回值的值

- 可以在呼叫來源處使用傳回值來進行處理。
- 範例：



- 沒有傳回值的函數
  - 範例：

```
/* buy函數的定義 */
void buy(void)
{
  printf("買了車子。\n");
}
```

  - 在不帶傳回值的函數中，要先將傳回值的型態設定爲void。
  - 也可利用以下這個單純的return敘述來結束函數：

```
/* buy函數的定義 */
void buy(void)
{
  printf("買了車子。\n");

  return;
}
```

8-5 函數的利用

- 求合計值的函數：

```
#include <stdio.h>
/* sum函數的定義 */
int sum(int x, int y)
{
  return x+y;
}

int main(void)
{
  int num1, num2, ans;
  printf("請輸入第1個整數：\n");
  scanf("%d", &num1);
  printf("請輸入第2個整數：\n");
  scanf("%d", &num2);
  ans = sum(num1, num2);
  printf("合計爲%d。\n", ans);
  return 0;
}
```

接收兩個數值

進行傳回合計值的處理

呼叫函數

輸出傳回值

- 求最大值的函數：

```
#include <stdio.h>
/* max函數的定義 */
int max(int x, int y)
{
  if (x > y)
    return x;
  else
   return y;
}

int main(void)
{
  int num1, num2, ans;
  printf("請輸入第1個整數：\n");
  scanf("%d", &num1);
  printf("請輸入第2個整數：\n");
  scanf("%d", &num2);
  ans = max(num1, num2);
  printf("最大值爲%d。\n",ans);
  return 0;
}
```

接收兩個數值

當**x**大於**y**的時候…

傳回**x**的值

如果不是的時候，則傳回**y**的值

呼叫函數

輸出傳回值

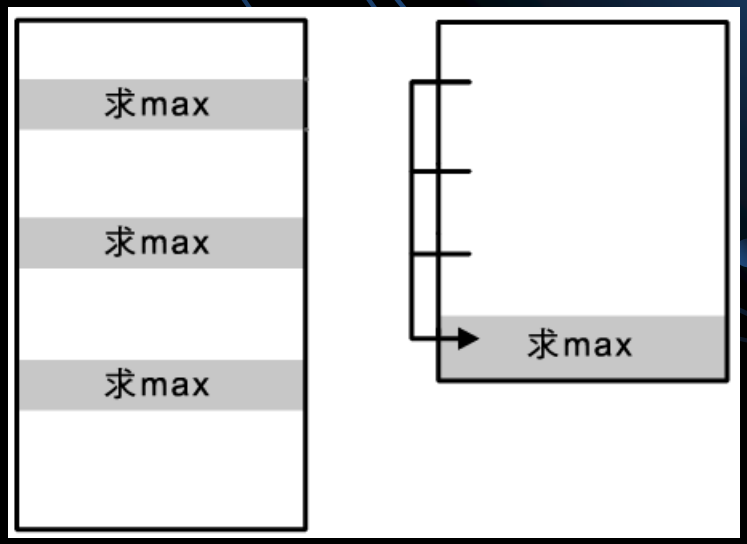
- C語言可以進行自己呼叫自己的處理，這種機制就叫做**遞迴**。

- 理解函數形式巨集的機制
  - 函數形式巨集的定義：
    #define 巨集名稱 (引數) (運算式)
  - 範例：
    #define MAX(x, y) (x, y ? x : y)
  - 函數形式巨集的呼叫：
    巨集名稱(引數);
  - 範例：

    ans = MAX(num1, num2);

    ↓

    ans = (num1 > num2 ? Num1 : num2);
  - 定義函數形式巨集可以用來代替函數。



- 使用函數形式巨集之範例：

```
#include <stdio.h>
#define MAX(x, y) (x > y ? x : y)

int main(void)
{
  int num1, num2, ans;
  printf("請輸入第1個整數。\n");
  scanf("%d", &num1);

  printf("請輸入第2個整數。\n");
  scanf("%d", &num2);

  ans = MAX(num1, num2);
  printf("最大值爲%d。\n", ans);
  return 0;
}
```

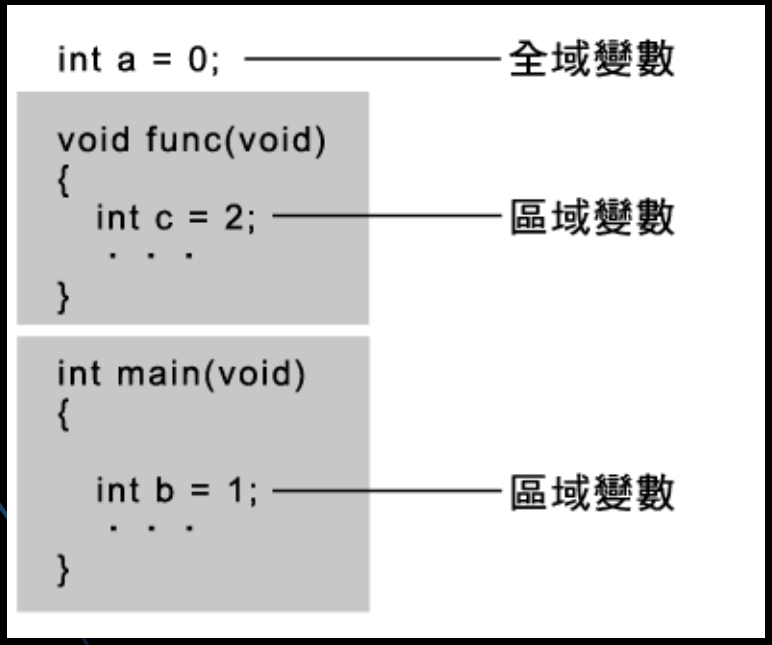
在這裡定義函數形式巨集

這部份的程式碼會藉由前置處理器來進行置換

- 使用函數形式巨集的注意事項：
  - 呼叫函數形式巨集的引數可以是任何型態。
  - 若是要建立較不容易出錯的程式，還是應該使用函數。

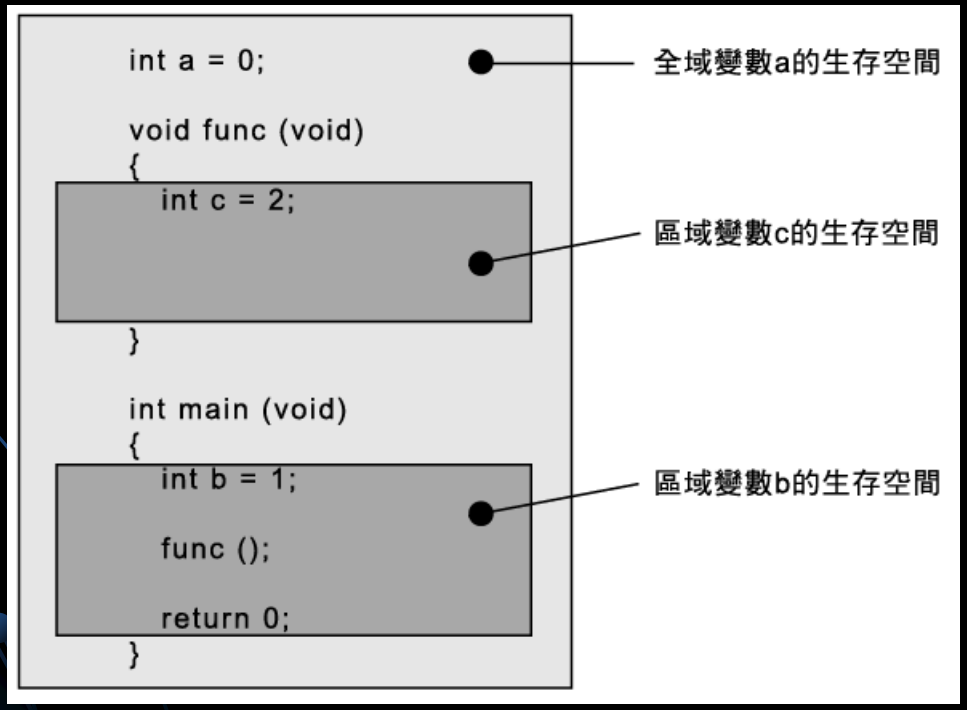
8-6 變數與生存空間

- 理解變數的種類
  - 在函數當中宣告的變數稱爲區域變數（local variable），而在函數外部宣告的變數則稱爲全域變數（global variable）。



- 理解生存空間（scope）
  - 生存空間（scope）是指變數在程式中的有效範圍，也就是它的生存空間，看它是區域或是全域的變數。

	宣告位置	生存空間
區域變數	函數內部	從宣告變數的地方開始到函數結束爲止都可利用
全域變數	函數外部	任何函數都可以利用



- 理解變數的生存空間之範例：

```
#include <stdio.h>
int a = 0;

/* func函數的定義 */
void func(void)
{
  int c = 2;
  printf("在func函數當中可以使用變數a和變數c。\n");
  printf("變數a的值爲%d。\n", a);
  /* printf("變數b的值爲%d。\n", b); */
  printf("變數c的值爲%d。\n", c);
}

/* main函數的定義 */
int main(void)
{
  int b = 1;
  printf("在main函數當中可以使用變數a和變數b。\n");
  printf("變數a的值爲%d。\n", a);
  printf("變數b的值爲%d。\n", b);
  /* printf("變數c的值爲%d。\n", c); */
  func();
return 0;
}
```

全域變數**a**

區域變數**c**

可以使用全域變數

無法使用其他函數內的區域變數

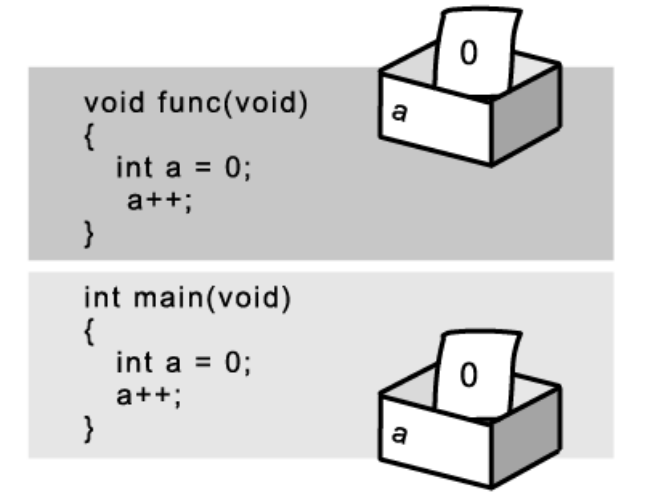
可以使用這個函數中的區域變數

可以使用全域變數

區域變數**b**

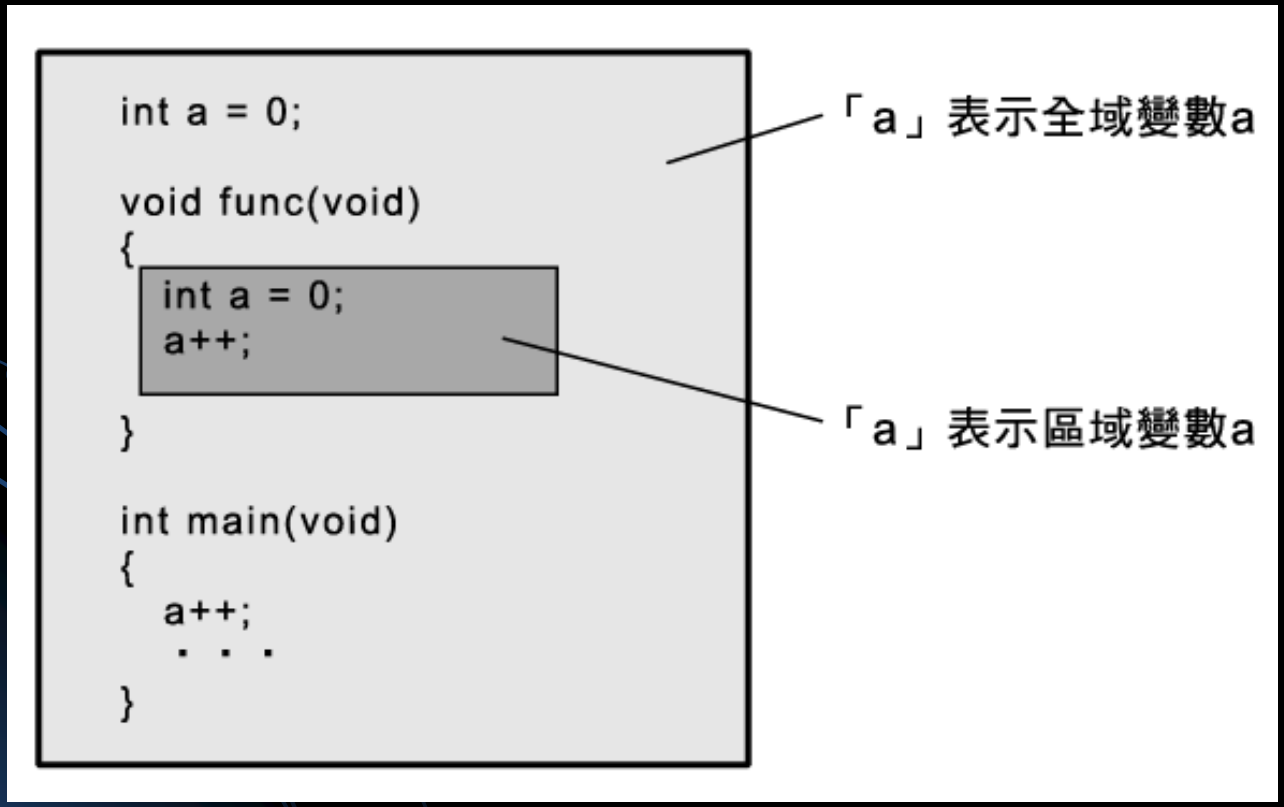
無法使用其他函數內的區域變數

- 區域變數的名稱可以重複使用
  - 位在不同函數內的區域變數，即使名稱相同，也不是同一個變數。



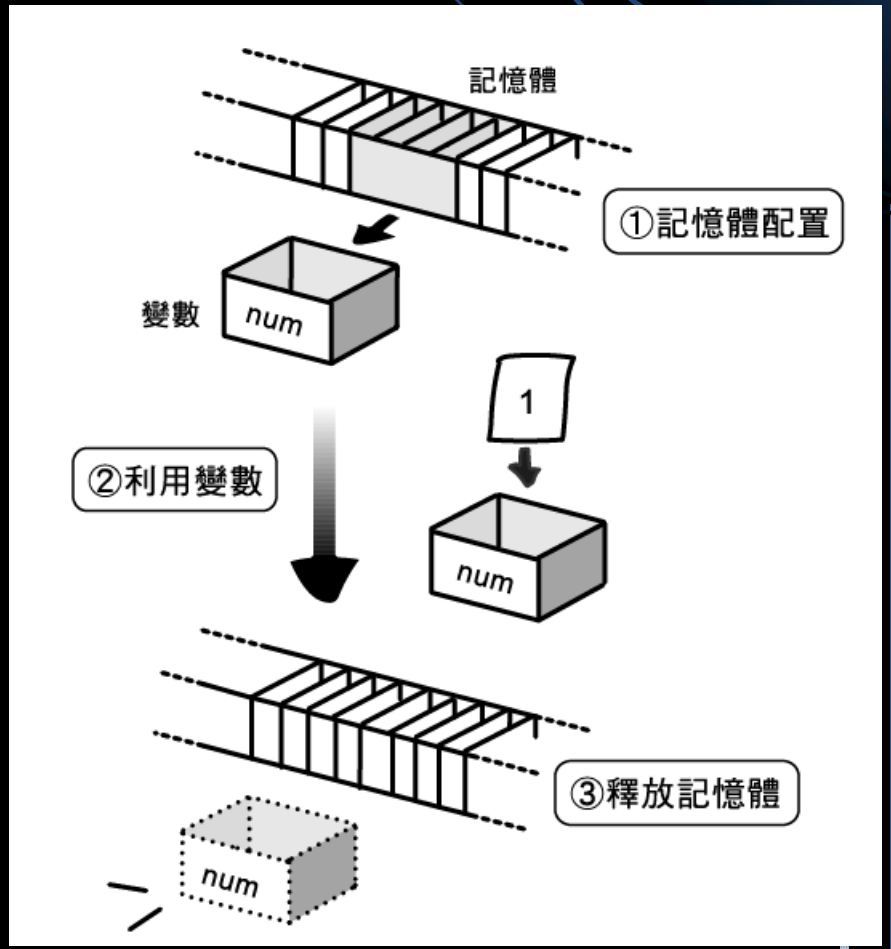
在不同函數內被宣告的區域變數，就代表是2個不同的變數

- 全域變數的名稱可以重複使用
  - 全域變數與區域變數可以使用相同的名稱，此時該函數內的全域變數會被區域變數掩蓋掉。



8-7 變數的生命週期

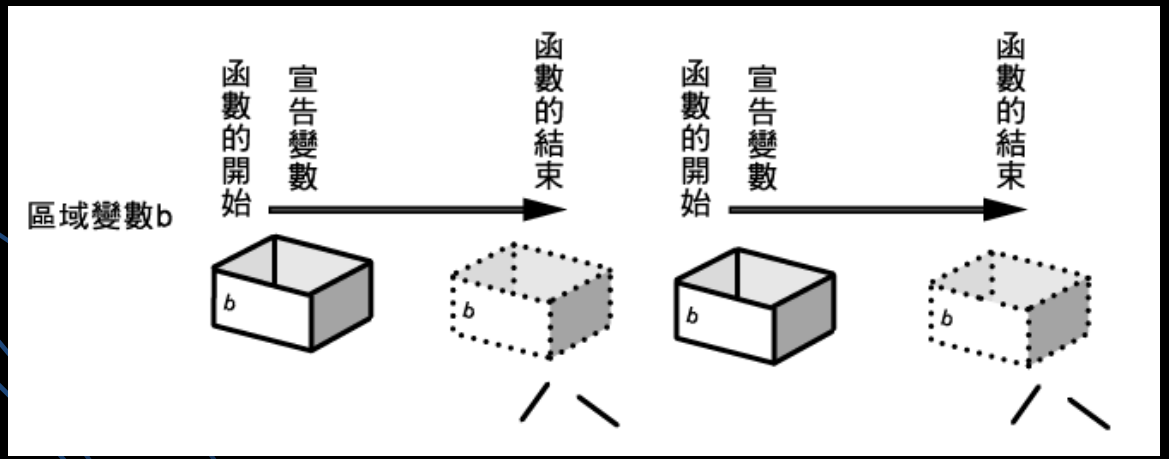
- 變數或是陣列從程式開始到結束爲止，並不是一直保持儲存著值的狀態。我們可以把這段狀態視爲是變數的「一生」。
- 從變數的儲存空間建立，到儲存數值的這段期間，就稱爲變數的**生命週期**。



- 區域變數的生命週期：
  - 當變數在函數內部被宣告時，就會在記憶體內配置其所需的空間。

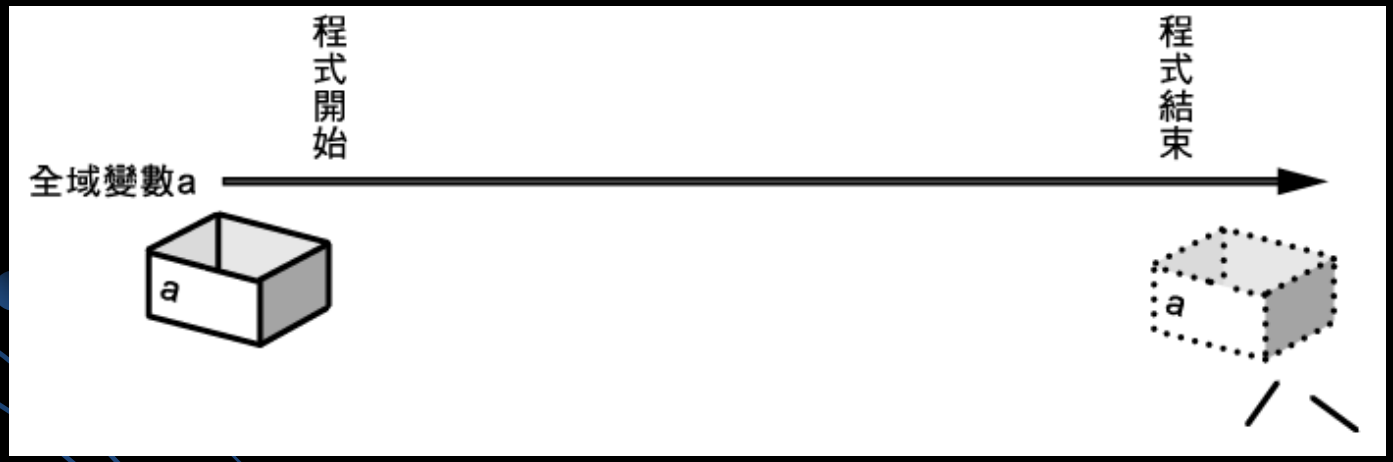
↓

  - 當函數結束時，就丟棄所配置的空間，使記憶體可以用在其他方面。



區域變數的生命週期

- 全域變數的生命週期：
  - 在程式本身的處理開始之前，先暫時配置記憶體。
  - ↓
  - 程式結束的時候釋放記憶體。



全域變數的生命週期

Sample11.c ▶ 確認變數的生命週期

```
#include <stdio.h>

int a = 0;

/* func函數的定義 */
void func(void)
{
    int b = 0;
    static int c = 0;

    printf("變數a為%d變數b為%d變數c為%d。\n", a, b, c);

    a++;
    b++;
    c++;
}

/* main函數的定義 */
int main(void)
{
    int i;

    for(i=0; i<5; i++)
        func();

  return 0;
}
```

這是全域變數 a

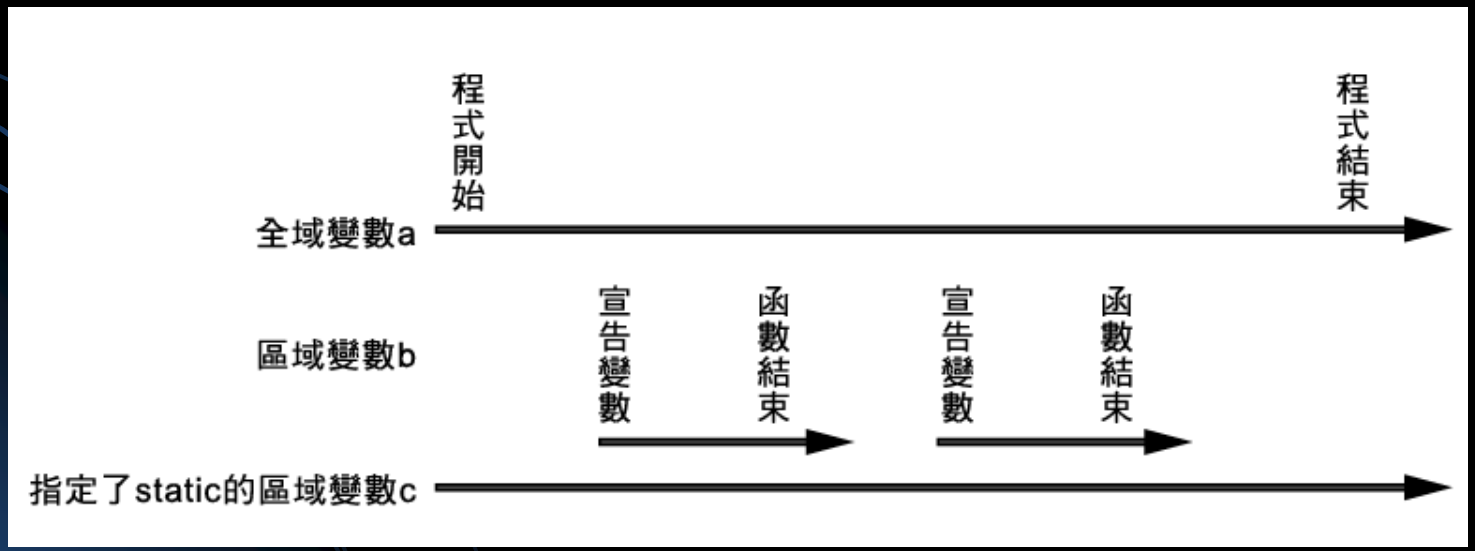
這是區域變數 b

這是指定了 static 的區域變數 c

遞增各變數

- 如果指定爲static的話？
  - 指定了static關鍵字的區域變數，就會擁有和全域變數一樣長的生命週期，這種區域變數就叫做**靜態區域變數**。
  - static又叫做**記憶類別指定字**(storage class identifier)。

	記憶類別	生存空間
區域變數	(自動)	起於宣告，終於函數結束 (自動)
	static	起於程式準備執行時，終於程式結束 (靜態)
全域變數		起於程式準備執行時，終於程式結束 (靜態)



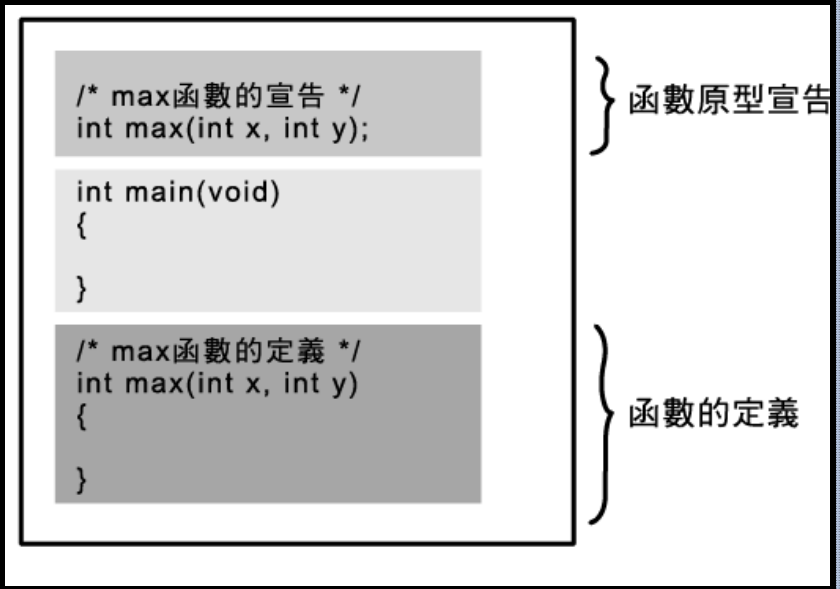
變數的生命週期

8-8 函數的宣告

- 函數原型宣告
  - 語法：

    傳回值的型態　函數名稱（引數列表）；

  - 在C語言可以先告知編譯器「函數的名稱及引數的數目」。
  - 這項功能就稱爲**函數原型宣告**（函數宣告：function declaration）。
  - 函數原型宣告是在呼叫函數之前，先寫下所呼叫函數的名稱、傳回值的型態、引數。



Sample12.c ▶ 使用函數原型宣告

```
#include <stdio.h>

/* max函數的宣告 */
int max(int x, int y);
int main(void)
{
    int num1, num2, ans;

    printf("請輸入第1個整數。\n");
    scanf("%d", &num1);

    printf("請輸入第2個整數。\n");
    scanf("%d", &num2);

    ans = max(num1, num2);

    printf("最大值為%d。\n", ans);

    return 0;
}

/* max函數的定義 */
int max(int x, int y)
{
    if (x > y)
        return x;
    else
        return y;
}
```

這是函數原型宣告

正在呼叫函數

可以寫在函數的定義之後

- 製作大型程式時會用到許多函數，這些函數不是記敘在與main()函數相同的檔案內，而是分割到其他的檔案。

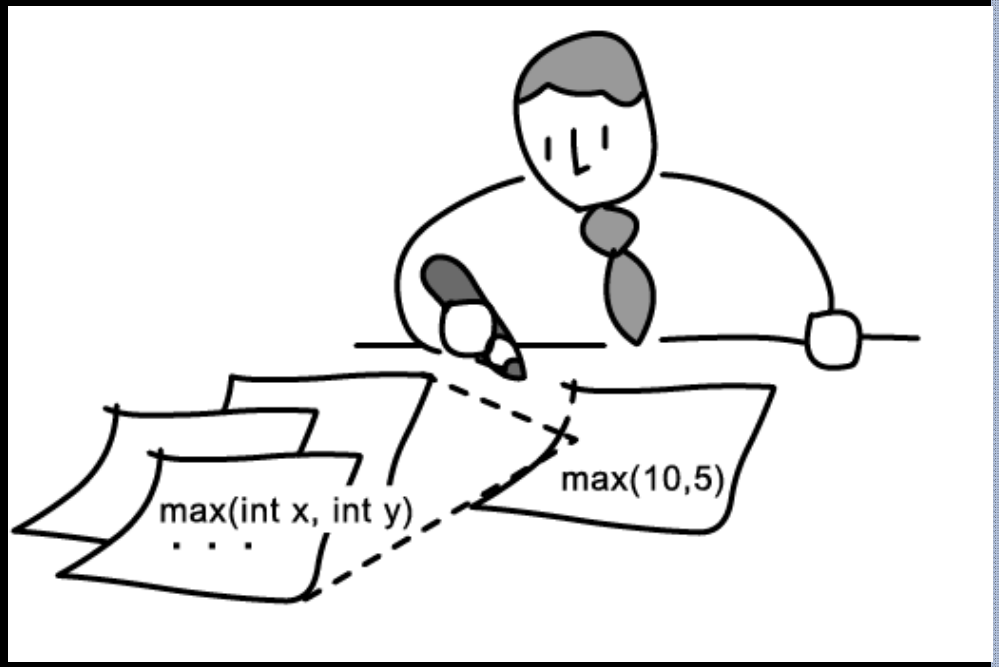
- 分割檔案的範例如下：

myfunc.h

```
/* max函數的宣告 */
int max(int x, int y);
```

myfunc.c

```
/* max函數的定義 */
int max(int x, int y)
{
    if (x > y)
        return x;
    else
        return y;
}
```



Sample13.c

```
#include <stdio.h>
#include "myfunc.h"
int main(void)
{
    int num1, num2, ans;
    printf("請輸入第1個整數。\n");
    scanf("%d",&num1);
    printf("請輸入第2個整數。\n");
    scanf("%d",&num2);
    ans = max(num1, num2);
    printf("最大值爲%d。\n", ans);
    return 0;
}
```

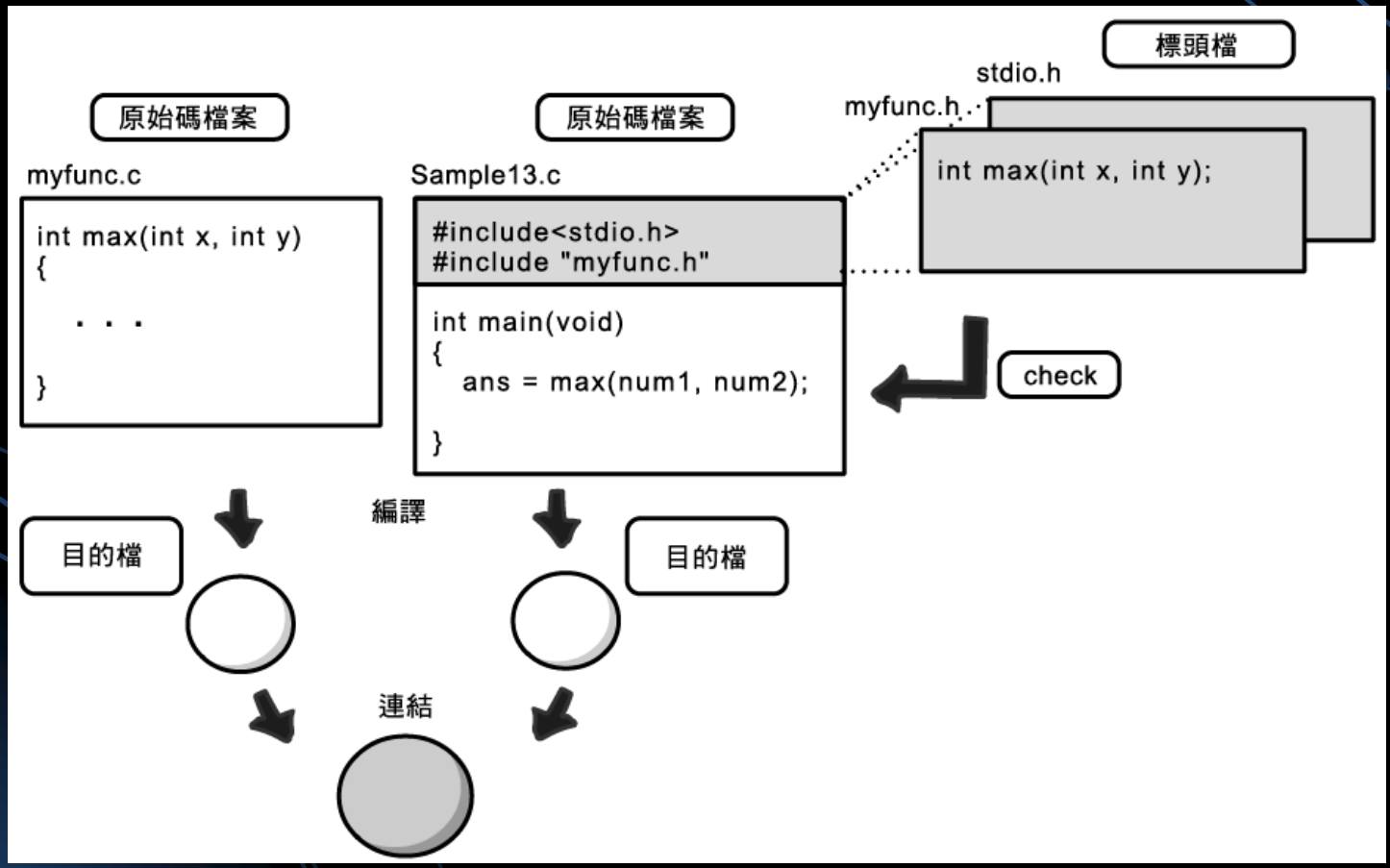
讀入標頭檔

呼叫其他檔案的函數

- myfunc.h … 函數原型宣告
- myfunc.c … 已做好的max()函數的定義
- Sample13.c … main()函數的定義 (程式主體)

- 分別編譯Sample13.c和myfunc.c，將它們做成不同的目的檔，再以連結的方式把這些目的檔串連成一個程式。
- myfunc.h是只負責記述函數原型宣告的**標頭檔**。

檔案的分割



理解標準程式庫函數的機制

- C語言的開發環境預先定義好的函數，就叫做**標準函數庫(standard library)**。
- 使用標準函數庫的標頭檔，要加上「< >」，而使用自己製作的標頭檔，則要加上「” ”」。