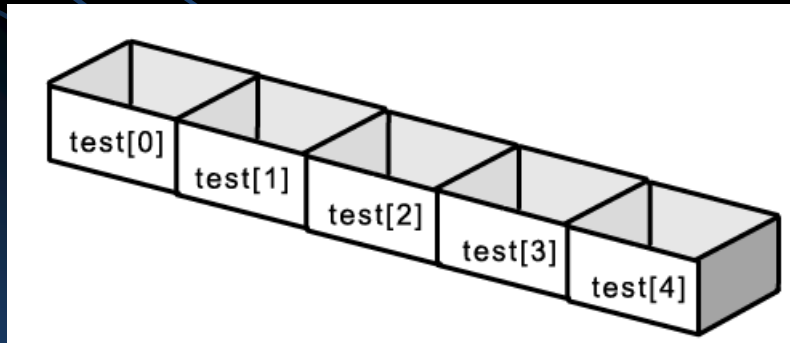
陣列、指標的應用

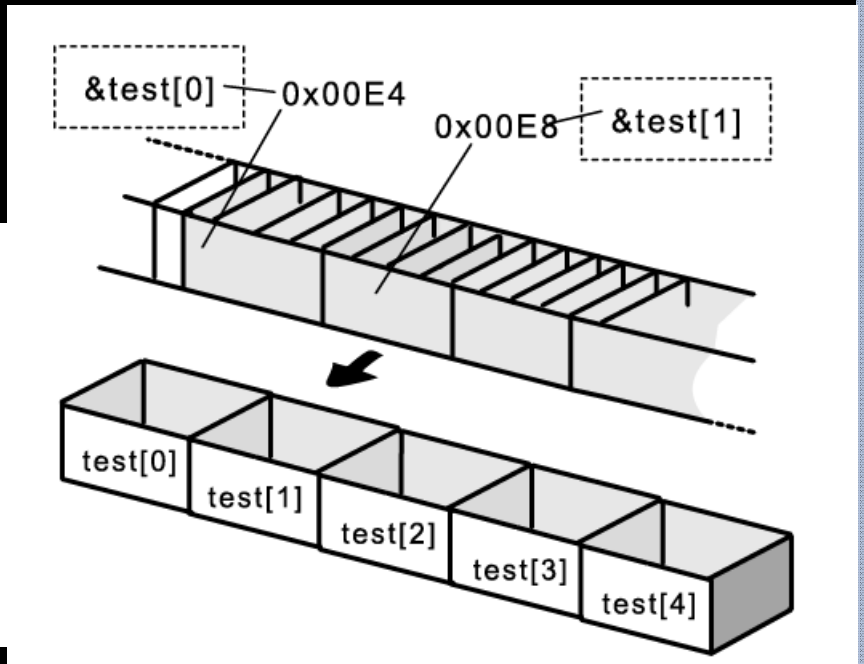
10-1 陣列與指標的關係

- 了解陣列元素的位址
  - 可以使用位址運算子(&)來查詢陣列中各個元素的位址。
  - 範例：

&test[0] ←——— 這行表示陣列最前面元素的位址

&test[1] ←——— 這行表示陣列第二個元素的位址





Sample1.c ▶ 求出陣列元素的位址

```
#include <stdio.h>

int main(void)
{
    int test[5] = {80,60,55,22,75};

    printf("test[0]的值為 %d。\n", test[0]);
    printf("test[0]的位址為 %p。\n", &test[0]);
    printf("test[1]的值為 %d。\n", test[1]);
    printf("test[1]的位址為 %p。\n", &test[1]);

    return 0;
}
```

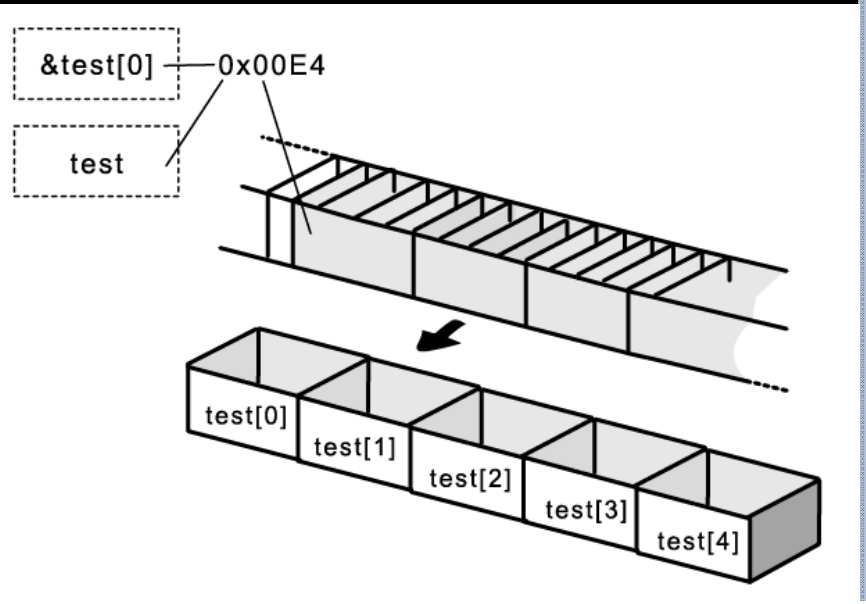
輸出陣列第 1 個元素的位址

輸出陣列第 2 個元素的位址

- 關於陣列名稱的機制
  - 陣列名稱可以表示陣列最前面元素的位址。
  - 範例：

```
#include <stdio.h>
int main(void)
{
    int test[5] = {80,60,55,22,75};
    printf("test[0]的值爲%d。\n", test[0]);
    printf("test[0]的位址爲%p。\n", &test[0]);
    printf("test的值爲%p。\n", test);
    return 0;
}
```

用陣列名稱來表示陣列最前面元素的位址



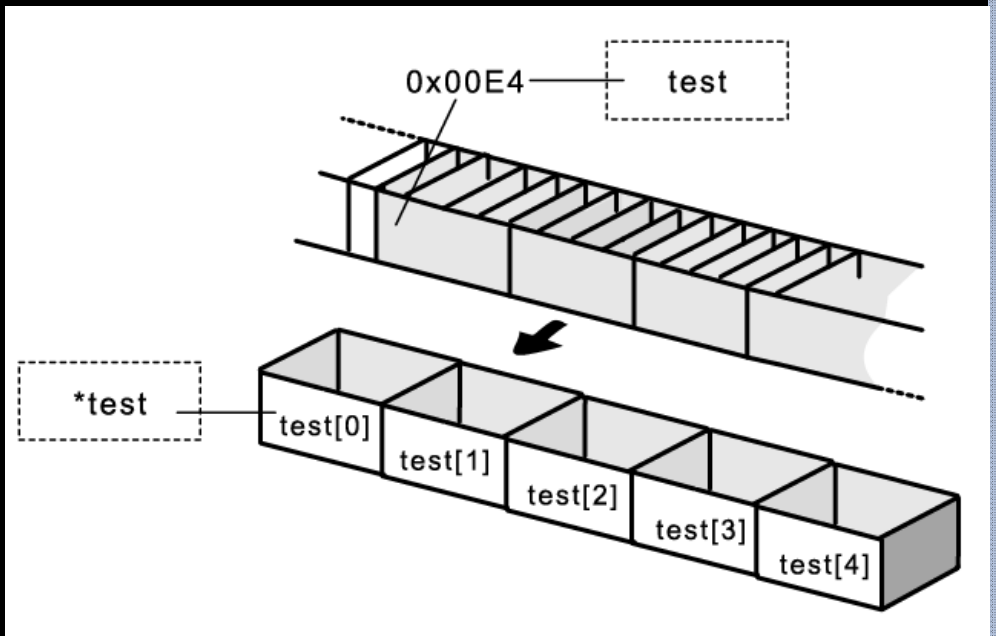
Sample2.c ▶ 使用陣列名稱

```
#include <stdio.h>

int main(void)
{
    int test[5] = {80,60,55,22,75};

    printf("test[0]的值為 %d。\n", test[0]);
    printf("test[0]的位址為 %p。\n", &test[0]);
    printf("test 的值為 %p。\n", test);

    return 0;
}
```

用「陣列名稱」來表示陣列最前面元素的位址

- 由陣列名稱來求出陣列最前面元素的值
  - 陣列名稱和儲存了陣列最前面元素位址的指標擁有相同的功用。
  - 在陣列名稱加上*之後，就可以表示陣列最前面元素的值。

test	&test[0]	陣列最前面元素的位址
*test	test[0]	陣列最前面元素的值



Sample3.c ▶ 由陣列名稱求最前面元素的值

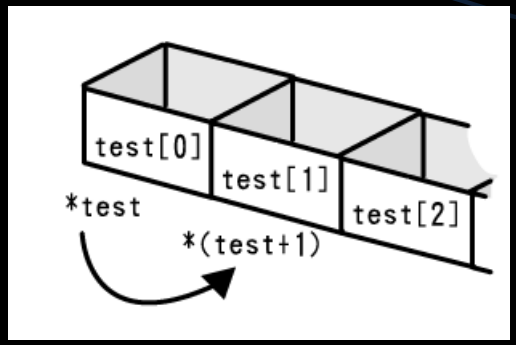
```
#include <stdio.h>

int main(void)
{
    int test[5] = {80,60,55,22,75};

    printf("test[0]的值為 % ◦ \n", test[0]);
    printf("test[0]的位址為 %p ◦ \n", &test[0]);
    printf("test 的值為 %p ◦ \n", test);
    printf(" 也就是說 *test 的值為 %d ◦ \n", *test);

    return 0;
}
```

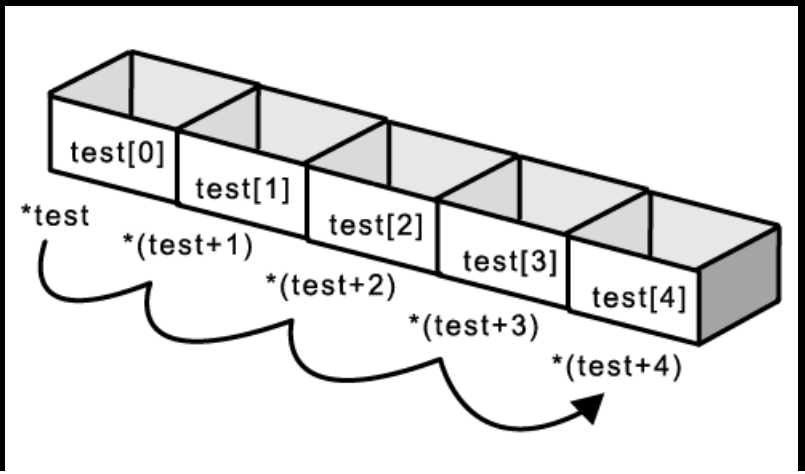
如果在陣列名稱前面加上 * 的話…

- 理解指標運算的機制

運算子	指標運算的例子（但是p、p1、p2爲指標）	
+	p+1	取得p所指向元素的下一個元素位址
-	p-1	取得p所指向元素的前一個元素位址
-	p1-p2	取得p1和p2之間的元素數
++	p++	取得p所指向的下一個元素位址
--	p--	取得p所指向的前一個元素位址

*test	test[0]	陣列最前面元素的值
*(test+1)	test[1]	陣列第二面元素的值
*(test+2)	test[2]	陣列第三個元素的值
…	…	…





Sample4.c ▶ 進行指標的運算

```
#include <stdio.h>

int main(void)
{
    int test[5] = {80,60,55,22,75};

    printf("test[0]的值為 %d。\n", test[0]);
    printf("test[0]的位址為 %p。\n", &test[0]);
    printf("test 的值為 %p。\n", test);
    printf("test+1 的值為 %p。\n", test+1);
    printf("*(test+1)的值為 %d。\n", *(test+1));

    return 0;
}
```

表示最前面元素的「下一個」元素位址

表示最前面元素的「下一個」元素位址

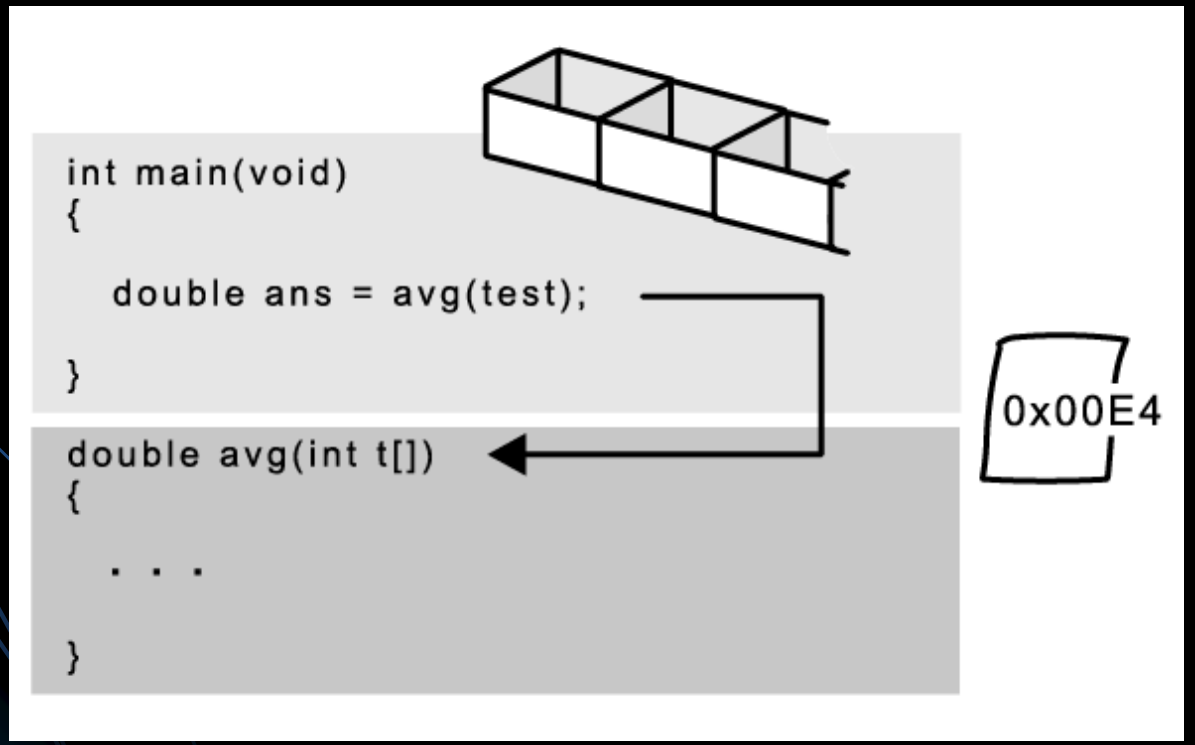
- 使用陣列名稱的注意事項
  - 陣列名稱可以想成是儲存陣列最前面元素之位址的指標。但這裡的指標和一般的指標有些不同。表示陣列名稱的指標不可以指派其他變數的位址。

  - 範例：

```
#include <stdio.h>
int main(void)
{
    int a = 5;
    int test[5] = {80,60,55,22,75};
    /* 錯誤 */
    /* test = &a; */     <-------- 無法指派其他的位址
    return 0;
}
```

10-2 引數與陣列

- 把陣列當作引數使用
  - 當使用陣列作爲引數時，必須傳遞陣列第一個元素的位址作爲引數。
  - 範例：



Sample5.c ▶ 使用陣列當成函數的引數

```
#include <stdio.h>

/* avg 函數的宣告 */
double avg(int t[]);

int main(void)
{
    int test[5];
    int i;
    double ans;

    printf("請輸入 5 個人的測驗成績。\n");

    for(i=0; i<5; i++){
        scanf("%d", &test[i]);
    }

    ans = avg(test);

    printf("5 個人的平均成績為 %lf 分。\n", ans);

    return 0;
}
```

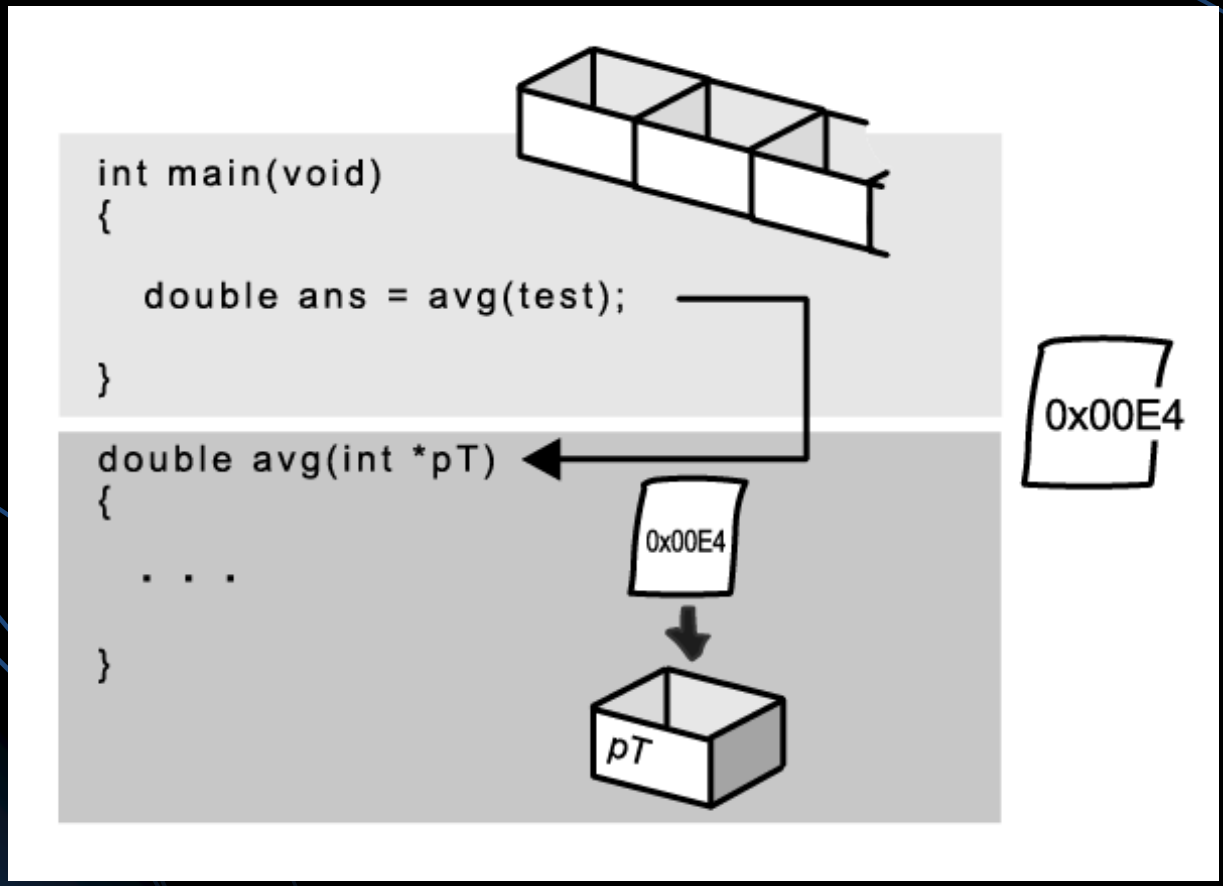
將陣列當作函數的引數使用

將陣列名稱（陣列第一個元素的位址）當作引數傳遞

```
/* avg函數的定義 */
double avg(int t[])
{
    int i;
    double sum;

    sum = 0.0;

    for(i=0; i<5; i++){
        sum += t[i];
    }

    return sum/5;
}
```

利用陣列

- 將指標當作引數使用
  - 當傳遞陣列第一個元素的位址作爲引數時，可以把指標寫成參數。
  - 範例：



Sample6.c ▶ 將指標當作引數使用

```
#include <stdio.h>

/* avg函數的宣告 */
double avg(int *pT);

int main(void)
{
    int test[5];
    int i;
    double ans;

    printf("請輸入5個人的測驗成績。\n");

    for(i=0; i<5; i++){
        scanf("%d", &test[i]);
   }

    ans = avg(test);

    printf("5個人的平均成績為%lf分。\n", ans);

    return 0;
}
```

```
/* avg 函數的定義 */
double avg(int *pT)
{
    int i;
    double sum;

    sum = 0.0;

    for(i=0; i<5; i++){
        sum += *(pT+i);
    }

    return sum/5;
}
```

用指標來表示參數…

利用指標運算來處理

- 對指標使用[ ]運算子
  - [ ]也稱爲**註標運算子**（subscript operator）。
  - 對指標使用[ ]運算子的表示方式之意義爲「從指標pT所指向的元素開始算起，找出前二位的元素」。
  - 當指標指向陣列時，把指標加上[ ]，就可以代表其所指向的元素。這正好與陣列的標示方法相同。
  - 只有在指標和陣列有密切的關係時，才可以在指標加上[ ]運算子。
  - 範例：

```
...
double avg(int *pT)
{
    int i;
    double sum;
    sum = 0.0;
    for(i=0; i<5; i++){
        sum += pT[i];
    }
    return sum/5;
}
```

←——— 可以對指標使用**[ ]**運算子

Sample7.c ▶ 將[]使用於指標

```
#include <stdio.h>

/* avg函數的宣告 */
double avg(int *pT);

int main(void)
{
    int test[5];
    int i;
    double ans;

    printf("請輸入5個人的測驗成績。\n");

    for(i=0; i<5; i++){
        scanf("%d", &test[i]);
    }

    ans = avg(test);

    printf("5個人的平均成績為%lf分。\n", ans);

    return 0;
}

/* avg函數的定義 */
double avg(int *pT)
{
    int i;
    double sum;

    sum = 0.0;

    for(i=0; i<5; i++){
        sum += pT[i];
    }
    return sum/5;
}
```

可以對指標使用[]運算子

- 將字串視爲指標處理
  - 範例1：

    char str[] = "Hello";

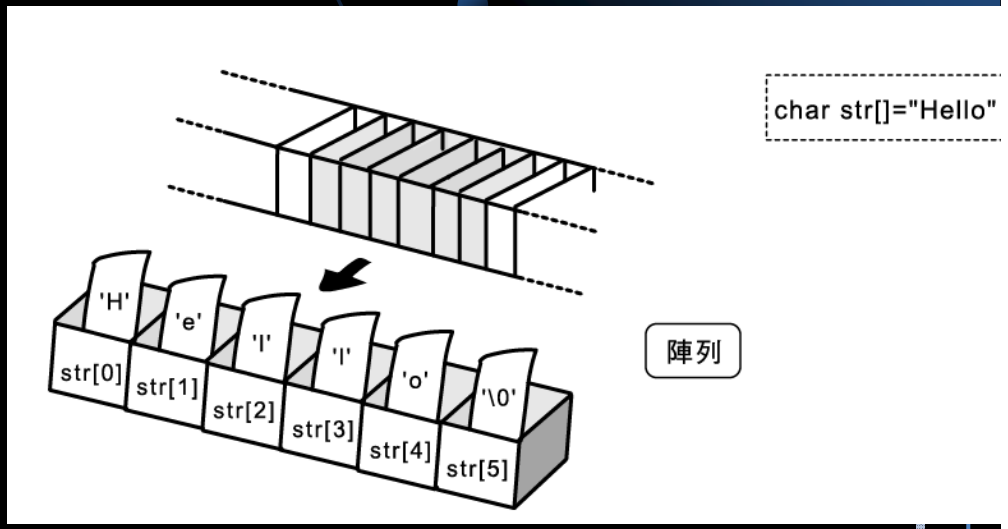
    利用陣列將字串初始化
  - 範例2：

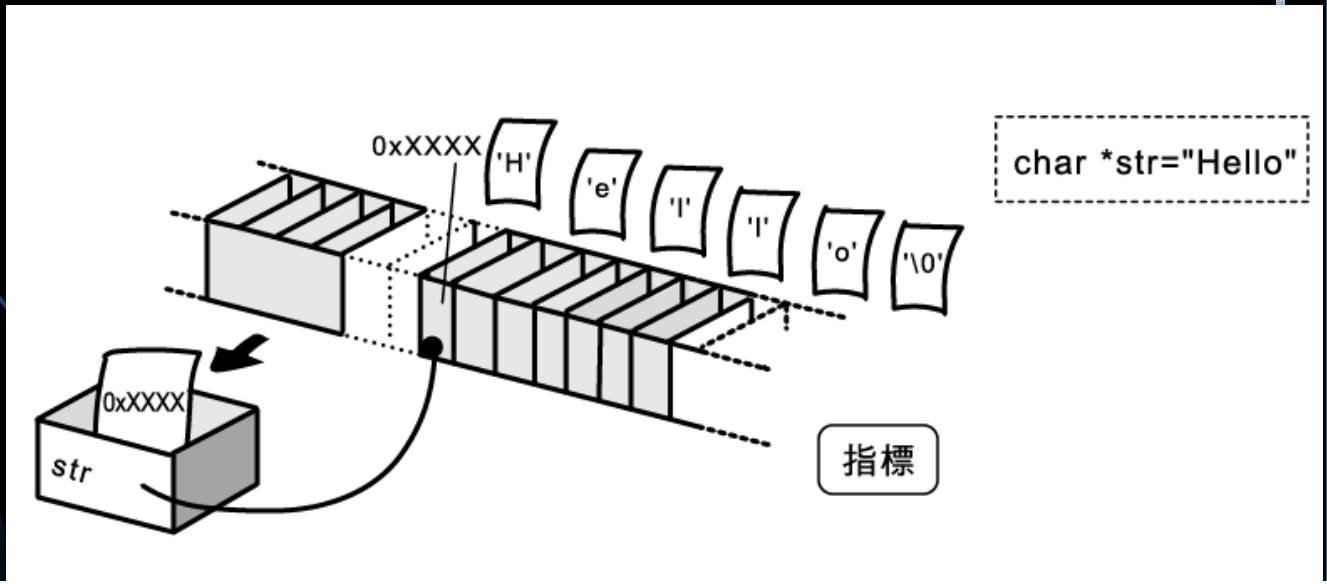
    char *str = "Hello";

    操作指向字串所在位址的指標
  - 範例3:

    printf(“字串爲%s。\n”, str);

    使用指標輸出字串





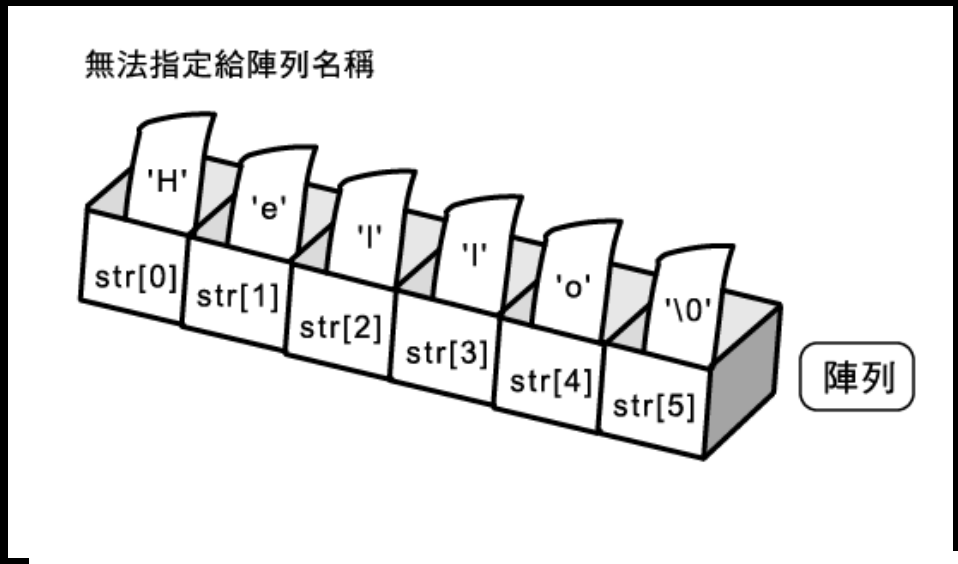
Sample8.c ▶ 以指標來輸出字串

```
#include <stdio.h>

int main(void)
{
    char *str = "Hello";

    printf("字串為 %s。\n", str);

    return 0;
}
```

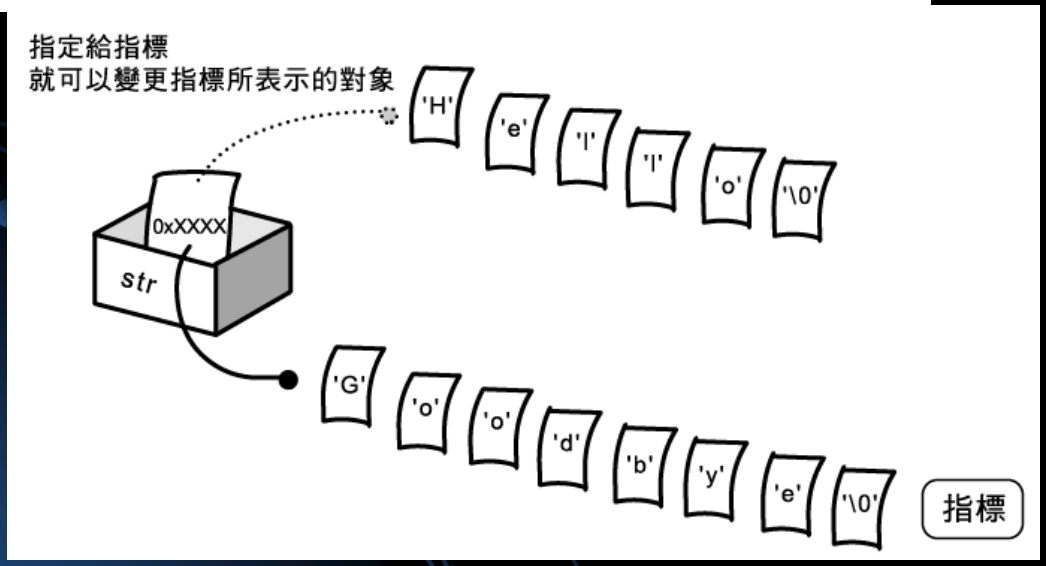
利用指標來操作字串

使用指標輸出字串

理解陣列和指標不同的地方



用陣列來操作字串時，無法藉由指派陣列名稱來改變字串。



而使用指標來操作字串時，可以指派到指標來變更所指向的字串。

字串陣列的陣列

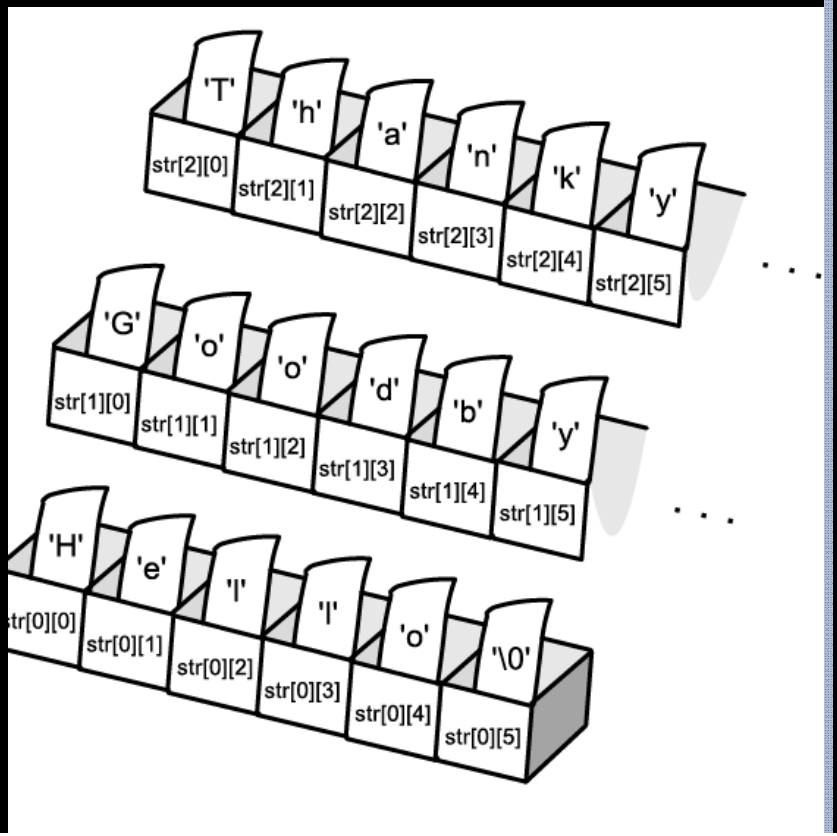
- 範例：

```
#include <stdio.h>
int main(void)
{
   char str[3][20] = {"Hello", "GoodBye", "Thankyou"};
   int i;

   for(i=0; i<3; i++){
      printf("字串爲%s。\n", str[i]);
   }

   return 0;
}
```

將一個字串限制在**19**個字元以內

準備好由三個字串所組成的陣列

將字串逐一輸出

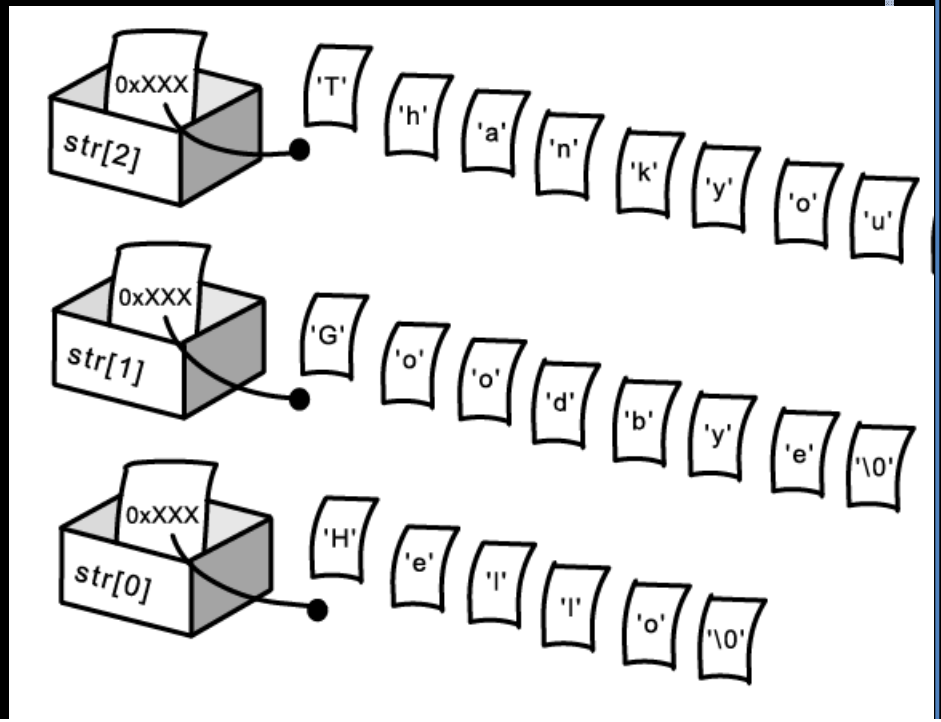


- 字串指標的陣列
  - 範例：

```
#include <stdio.h>
int main(void)
{
    char *str[3] = {"Hello", "GoodBye", "Thankyou"};
    int i;
    for(i=0; i<3; i++){
        printf("字串爲%s。\n", str[i]);
    }
    return 0;
}
```

準備好由三個字串所組成的陣列

將字串逐一輸出



10-4 字串的操作

- 使用標準程式庫函數
  - C語言的開發環境中包含了**標準程式庫函數** (standard library)。
  - 使用字串操作函數的時候，要先引入string.h這個檔案。

表10-2：操作字串的標準程式庫函數（string.h）

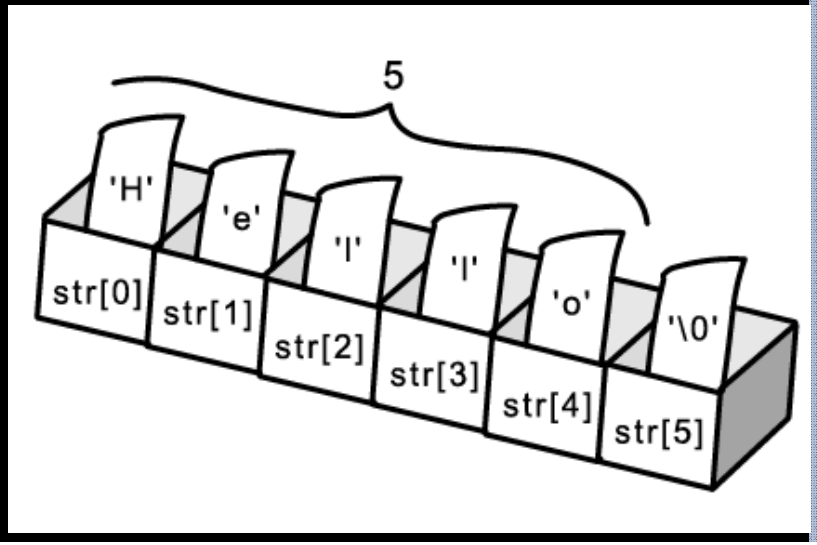
size_t strlen(const char *str);
除去字串s的NULL字元後，傳回長度。
char * strcpy(char *str1, const char *str2);
將字串str2複製到str1區域後，傳回str1。
char * strcat(char *str1, const char *str2);
將字串str2追加到字串str1的最後面，然後傳回str1。
int strcmp(const char *str1, const char *str2);
比較字串str1與字串str2，依照情況傳回以下的值。
如果字串str1比字串str2小的時候　　：負值
如果字串str1與字串str2相等的時候　　：0
如果字串str1比字串str2大的時候　　：正值

- 查詢字串的長度
  - 要查出字串的長度時，可以使用strlen()函數。
  - 範例：

```
#include <stdio.h>
#include <string.h>
int main(void)
{
    char str[100];
    printf("請輸入一個字串（英數字）。\n");
    scanf("%s", str);
    printf("字串的長度爲%d。\n", strlen(str));
    return 0;
}
```

先引入這個標頭檔

輸出字串的長度

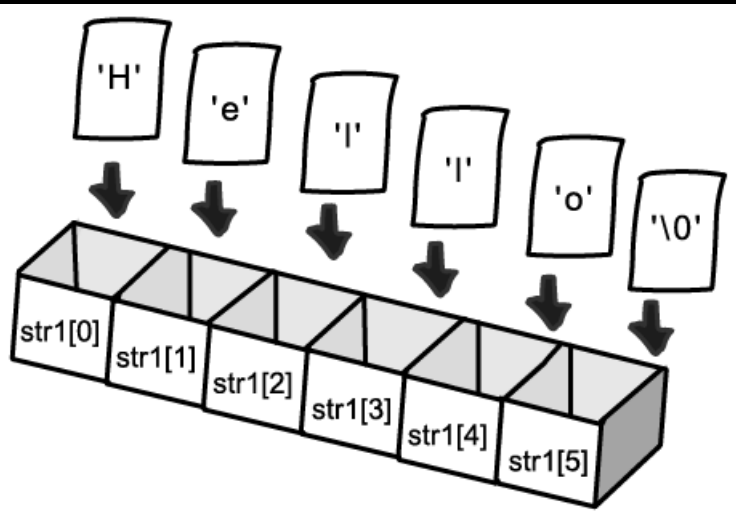


- 將字串複製到陣列當中
  - 要把字串複製到陣列的時候，可以使用strcpy()函數。
  - 範例：

```
#include <stdio.h>
#include <string.h>
int main(void)
{
    char str1[10];
    char str2[10];
    strcpy(str1,"Hello");
    strcpy(str2,"Goodbye");
    printf("陣列str1爲%s。\n", str1);
    printf("陣列str2爲%s。\n", str2);
    return 0;
}
```

把”**Hello**”複製到**str1**當中

把”**Goodbye**”複製到**str2**當中



- 將字串連結起來
  - 要將字串連結起來，可以使用strcat()函數。
  - 範例：

```
#include <stdio.h>
#include <string.h>
int main(void)
{
   char str0[20];
   char str1[10];
   char str2[10];

   strcpy(str1,"Hello");
   strcpy(str2,"Goodbye");
   strcpy(str0,str1);
   strcat(str0,str2);

   printf(“陣列str1爲%s。\n", str1);
   printf(“陣列str2爲%s。\n", str2);
   printf(“二個陣列連結起來就變成%s。\n", str0);

   return 0;
}
```

把str1複製到str0中

在str0的最後連結上str2

輸出連結後的字串

strcpy (str1, "Hello") ;

str1

H	e	l	l	o	\0								

strcpy (str0, "Goodbye") ;

str2

G	o	o	d	b	y	e	\0		

strcpy (str0, str1) ;

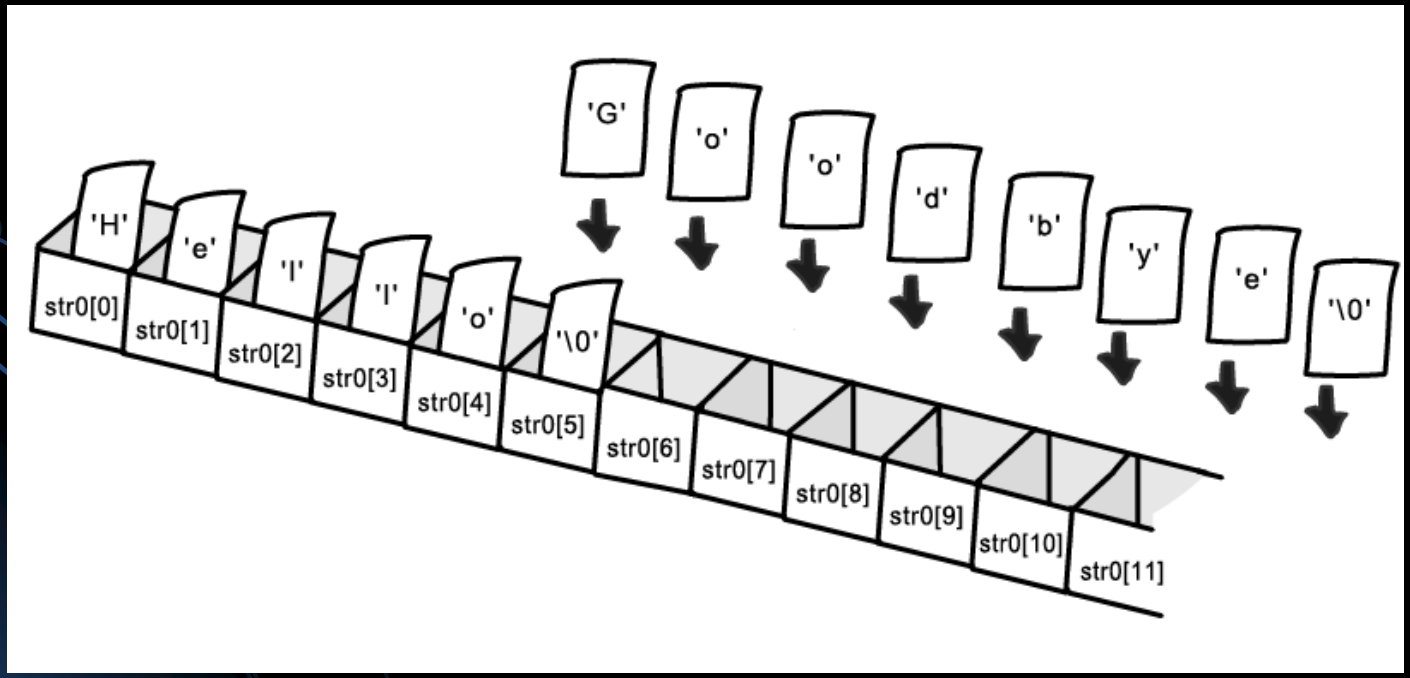
str0

H	e	l	l	o	\0								

strcat (str0, str2) ;

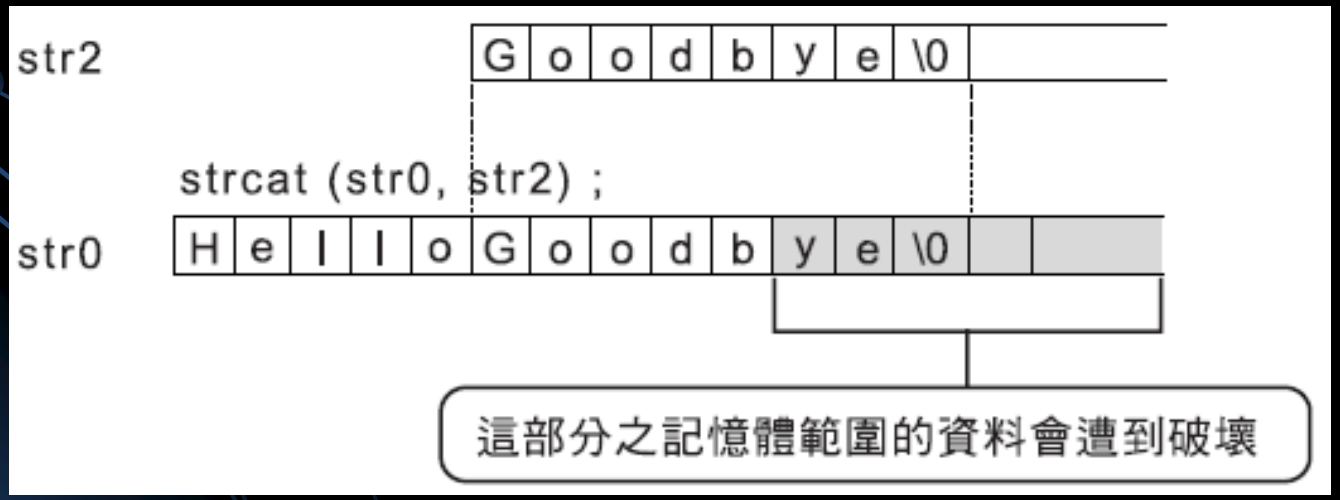
str0

H	e	l	l	o	G	o	o	d	b	y	e	\0		



- 注意陣列的大小
  - 在把字串複製到陣列的情況下，要注意不可以超出陣列的大小。
  - 範例：

```
int main(void)
{
    char str0[10];   <-------  不可以使用大小不足的陣列
    char str1[10];
    char str2[10];
    …
```



- 將字串進行比較
  - 要比較字串的時候，可以使用strcmp()函數。
  - 範例：

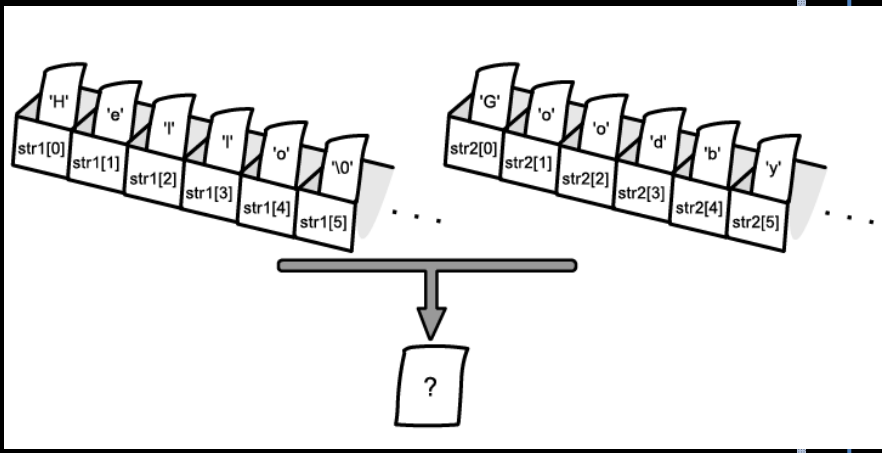
```
#include <stdio.h>
#include <string.h>
int main(void)
{
    char str1[100];
    char str2[100];
    printf("請輸入第1個字串。\n");
    scanf("%s", str1);
    printf("請輸入第2個字串。\n");
    scanf("%s", str2);
    if(strcmp(str1, str2) == 0){
        printf("2個字串是相同的。\n");
    }
    else
    {
        printf("這2個字串不一樣。\n");
    }
    return 0;
}
```

準備長度相同的陣列

如果比較字串之後的結果是0…

則輸出「相同」的意思

否則就輸出「不一樣」的意思



- 執行的時候才決定字串的長度
  - 範例：

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    char *str;
    int num, i;
    printf("要準備幾個a？\n");
    scanf("%d", &num);
    str = (char *) malloc (sizeof(char) * (num+1));
    if(!str){
        printf("無法配置記憶體。\n");
        return 1;
    }
    for(i=0; i<num; i++){
        *(str+i) = 'a';
    }
        *(str+num) = '\0';
    printf("準備了%s。\n", str);
    free(str);
    return 0;
}
```

引入這個標頭檔

得到表示被配置的記憶體位置之指標

指定的數量 **+1**

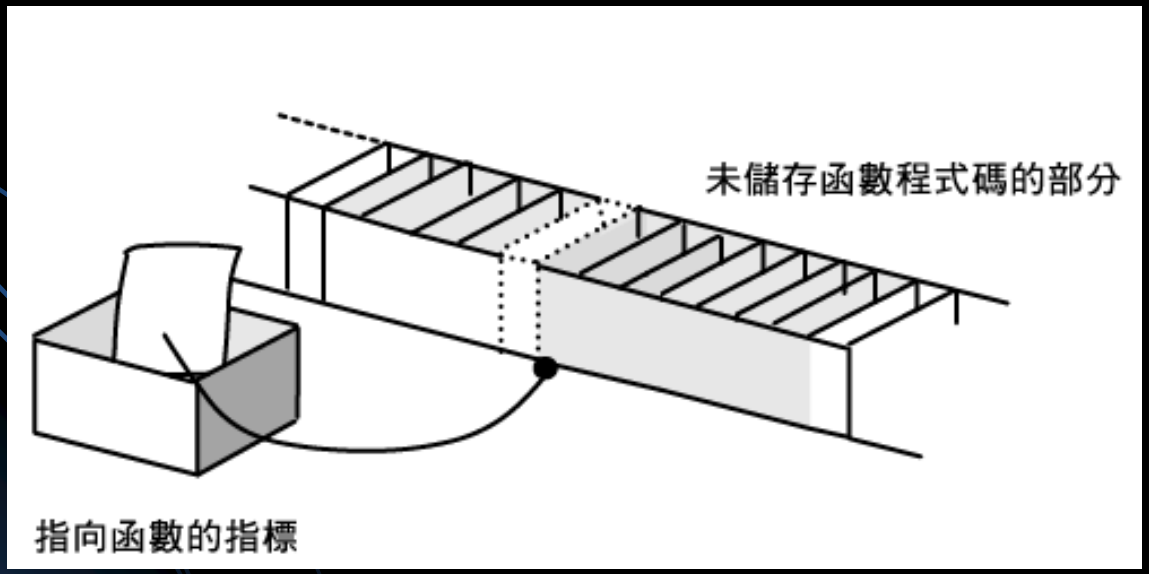
配置**char**型態大小的記憶體

使用指標儲存到陣列中

最後加上**\0**當成字串

10-5 函數指標

- 理解函數指標
  - 函數擁有記憶體上的位址，函數指標當中則儲存了函數的位址。
  - 儲存這種「函數位址」的指標就叫做**函數指標** (function pointer)。



- 宣告函數指標
  - 宣告函數指標的語法：

    傳回值的型態 (*指標) (引數列表)

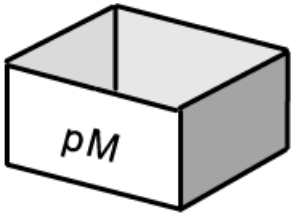
  - 範例：

    函數指標pM的宣告

    int (*pM)(int x, int y);

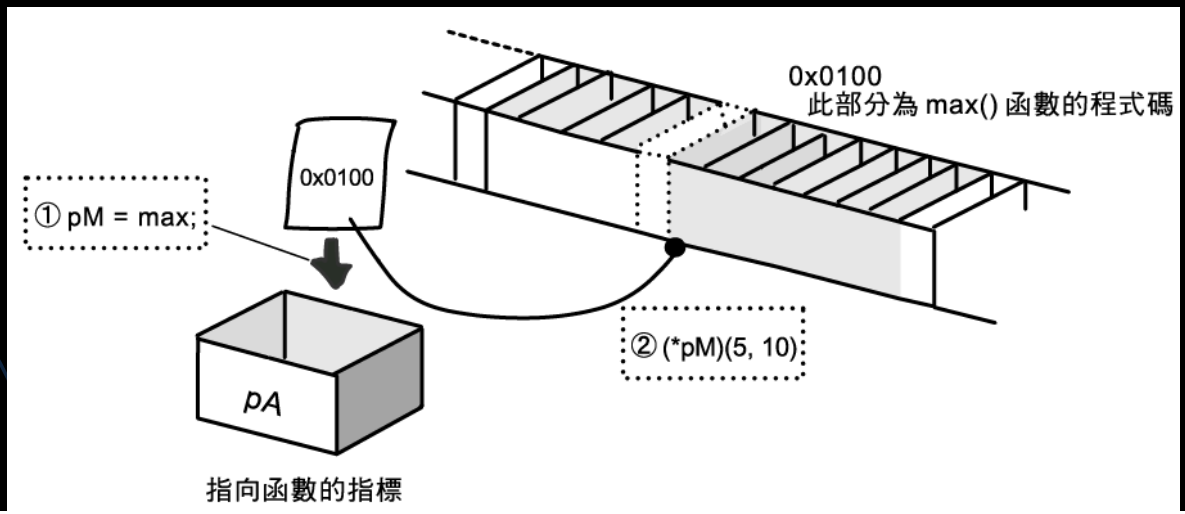
    指定函數傳回值的型態

    指定函數的引數型態



指標指向傳回值為int型態、
有2個參數之int型態的函數

- 將位址儲存於函數指標
  - 指定指向函數指標的位址之語法：
    函數指標 = 函數名稱;
  - 範例：
    pM = max;
  - 呼叫利用函數指標的函數之語法：
    (*函數指標) (引數列表);
  - 範例：
    res = pM(5, 10);



- 運用函數指標
  - 可以使用函數指標來呼叫出函數。
  - 範例：

```
#include <stdio.h>
/*max函數的宣告 */
int max(int x, int y);
int main(void)
{
    int num1, num2, ans;
    int (*pM)(int x, int y);
    pM = max;
    printf("請輸入第1個數值。\n");
    scanf("%d",&num1);
    printf("請輸入第2個數值。\n");
    scanf("%d",&num2);
    ans = (*pM)(num1, num2);
    printf("最大值爲%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;
}
```

宣告函數指標

指定函數的位址

使用指標呼叫出函數

- 應用函數指標之範例：

```
#include <stdio.h>
/*  函數的宣告 */
void show0(void);
void show1(void);
void show2(void);
int main(void)
{
   void (*pM[3])(void);
   int num;
   pM[0] = show0;
   pM[1] = show1;
   pM[2] = show2;
   printf("要呼叫哪一種交通工具？（0:汽車   1:賽車   2:飛機）\n");
   scanf("%d",&num);
   if(0<= num && num <= 2)
      (*pM[num])();
   return 0;
}
/* 函數的定義 */
void show0(void)
{
   printf("汽車。\n");
}
void show1(void)
{
   printf("賽車。\n");
}
   void show2(void)
{
   printf("飛機。\n");
}
```

宣告函數指標的陣列

將函數的指標指定給陣列元素

使用指標呼叫出函數