第十章

指標
(Pointer)

大綱

- 指標變數的宣告
- 指標的運算
- 指標與函數
- 指標與陣列
- 指標的指標

變數的位址

- 記憶體每個byte有其相對應的位址(address)
- 變數會佔用一或數個連續的byte來儲存資料
- 變數的位址是其佔用空間第1個byte的位址
- 整數變數a佔用位址為0x0022FF34 ~ 0x0022FF37，並以0x0022FF34為代表
- 如何找出變數所佔用位址?

位址	內容	類型 名稱
0022FF2F		
0022FF30	5566	int b;
0022FF31		
0022FF32		
0022FF33		
0022FF34	3392	int a;
0022FF35		
0022FF36		
0022FF37		
0022FF38		

```
int a = 3392;
int b = 5566;
```

取址運算子(address)

- 利用取址運算子&，置放於變數之前，可找出變數所佔用位元組的第一個位址(address)

運算子	意義	範例	說明
&	address	&a	找出變數a的位址

```
int a=1, b=0;

printf("a=%d b=%d", a, b);       /*印出a和b的內容(value)*/

printf("&a=%p &b=%p", &a, &b); /*印出a和b的位址(address)*/
```

- 如何將找出的位址(address)用變數加以儲存?
  - char, int, short, long, float, double?

指標變數(pointer)

- 變數指標可用來儲存位址(address)

宣告語法	範例
資料型態 *指標變數; 資料型態 *指標變數1, *指標變數2;	int *p; /*宣告整數指標變數*/ char *p1,*p2; /*宣告字元指標*/

- 各變數位址須儲存於對應類型的指標變數

```
int a = 1;
char c = 'a';
double d = 3.14;

int *p1;                /*  宣告整數指標  */
p1 = &a;                /*  將變數a的位址儲存於指標p1  */
char *p2 = &c;          /*  宣告字元指標，且設定初始值  */
double *p3 = &d;        /*  宣告倍精度浮點數指標  */
```

- 得知位址有何用途?

指標變數佔用空間

- 標變數儲存其對應類型變數起始位址
- 各類型指標變數佔用一個位址大小的空間
  - 4 Bytes (32bit定址), 8 Bytes (64bit定址)

```
int a = 1;
char c = 'a';
double d = 3.14;

int *p1 = &a;
char *p2 = &c;
double *p3 = &d;

printf("%d %d %d", sizeof(a), sizeof(c), sizeof(d));
printf("%d %d %d", sizeof(&a), sizeof(&c), sizeof(&d));
printf("%d %d %d", sizeof(p1), sizeof(p2), sizeof(p3));
printf("%d %d %d", sizeof(&p1), sizeof(&p2), sizeof(&p3));
```

指標變數運算

- 指標變數搭配加、減運算可變更其內容
  - 各指標變數依其類型增減其位址
  - 一個加或減運算，增減一個對應類型變數佔用空間所需位址

```
int a = 1;
char c = 'a';
double d = 3.14;

int *p1 = &a;
char *p2 = &c;
double *p3 = &d;

printf("%p %p %p\n", p1, p2, p3);
printf("%p %p %p\n", p1+1, p2+1, p3+1); // +1 的結果?
```

間接取值運算子(indirection)

- 得知特定位址，可取得該位址的內容
  - 搭配指標變數，相當於使用該位址的變數
  - 運用間接取值運算子可變更指定位址的內容

運算子	意義	範例	說明
*	indirection	*p	找出指標p所指位址的內容

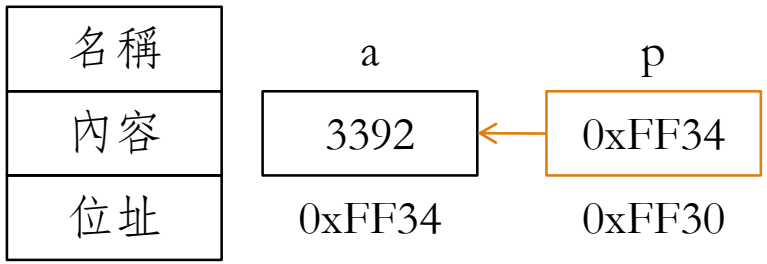
```
int a = 1, b = 2;
int *p1 = &a;

printf("%d\n", *p1); // p1指向a，*p1相當於a
p1 = &b;
printf("%d\n", *p1); // p1指向b，*p1相當於b
p1++;
printf("%d\n", *p1); // p1指向?，*p1相當於?
*p1 = 3;
printf("%d %d\n, a, b);
```

指標的設定與取用

- 指標常用的運算有兩種：
  - 取出變數的位址，然後存放在指標裡
  - 取出指標變數所指向之變數的內容

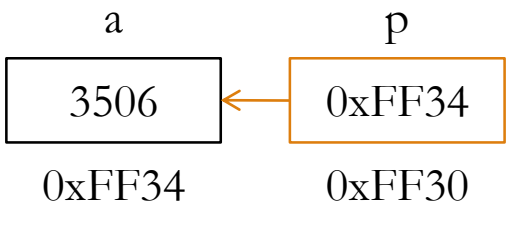
取址運算子(&)

```
int a = 3392;
int *p;
p = &a; //設定位址
```



間接取值運算子(*)

```
int a = 3392;
int *p = &a;
*p = 3506; //取用內容
```



為什麼要用指標

- 指標可隨時改變所指對象，具高度彈性
  - 處理變數
  - 呼叫不同的函數 (指標亦可指向函數)*

```
int a = 3392;
int *p;
int b = 3506;
p = &a; //指向變數a
p = &b; //指向變數b
```

```
//試比較以下敘述
a = 2121;
b = 3232;
*p = 5566;
```

位址	內容	類型 名稱
0022FF2B		
0022FF2C	3506	int b;
0022FF2D		
0022FF2E		
0022FF2F		
0022FF30	0022FF2C	int *p;
0022FF31		
0022FF32		
0022FF33		
0022FF34	3392	int a;
0022FF35		
0022FF36		
0022FF37		

指標變數的使用範例

```
int a = 5;
double b = 3.14;
int *p1;
double *p2;
int *p3;

p1 = &a;
p2 = &b;
printf("a=%8d (%d) 0x%p\n",a,sizeof(a),&a);
printf("b=%8.2f (%d) 0x%p\n",b,sizeof(b),&b);
printf("p1=%p (%d) 0x%p, *p1=%d\n",p1,sizeof(p1),&p1,*p1);
printf("p2=%p (%d) 0x%p,*p2=%.2f\n",p2,sizeof(p2),&p2,*p2);

*p1 = 10;
*p2 = 6.28;
printf("a=%d, b=%.2f\n",a,b);

p3 = p1;
printf("*p3=%d\n",*p3);
```

- 如何變更指標的內容?
- 如何變更指標所指的內容?

傳遞指標到函數

- 傳遞指標的函數

```
傳回值型態 函數名稱(資料型態 *指標變數)
{
  /* 函數的本體 */
}
```

接收的是變數的位址

```
int main(void)
{
    int num=5;
    func(&num);
    ...
}
```

傳遞變數的位址

```
void func(int *ptr)
{
    /* 函數的本體 */
}
```

接收變數的位址

傳遞內容 vs. 傳遞位址

- 傳遞數值

```
int main(void)
{
    int a = 3392;
    afunc(a);
    return 0;
}
```

afunc(3392)

```
void afunc(int x)
{
    printf("%d",x);
    return;
}
```

- 傳遞位址

```
int main(void)
{
    int a = 3392;
    afunc(&a);
    return 0;
}
```

afunc(0x0022FF34)

```
void afunc(int *x)
{
    printf("%p",x);
    return;
}
```

設定變數內容 vs. 設定位址內容

- 傳遞數值

```
int main(void)
{
    int a = 3392;
    afunc(a);
    printf("%d",a);
    return 0;
}
```

```
void afunc(int x)
{
    x = 5566;
    return;
}
```

- 傳遞位址

```
int main(void)
{
    int a = 3392;
    afunc(&a);
    printf("%d",a);
    return 0;
}
```

```
void afunc(int *x)
{
    *x = 5566;
    return;
}
```

函數傳遞位址範例

- 無法變更變數

```
int main(void)
{
    int a=1, b=2;

    swap1(a,b);
    printf("%d %d",a,b);

    return 0;
}
```

```
void swap1(int p1,int p2)
{
    int tmp;
    tmp = p1;
    p1 = p2;
    p2 = tmp;
}
```

- 交換變數內容

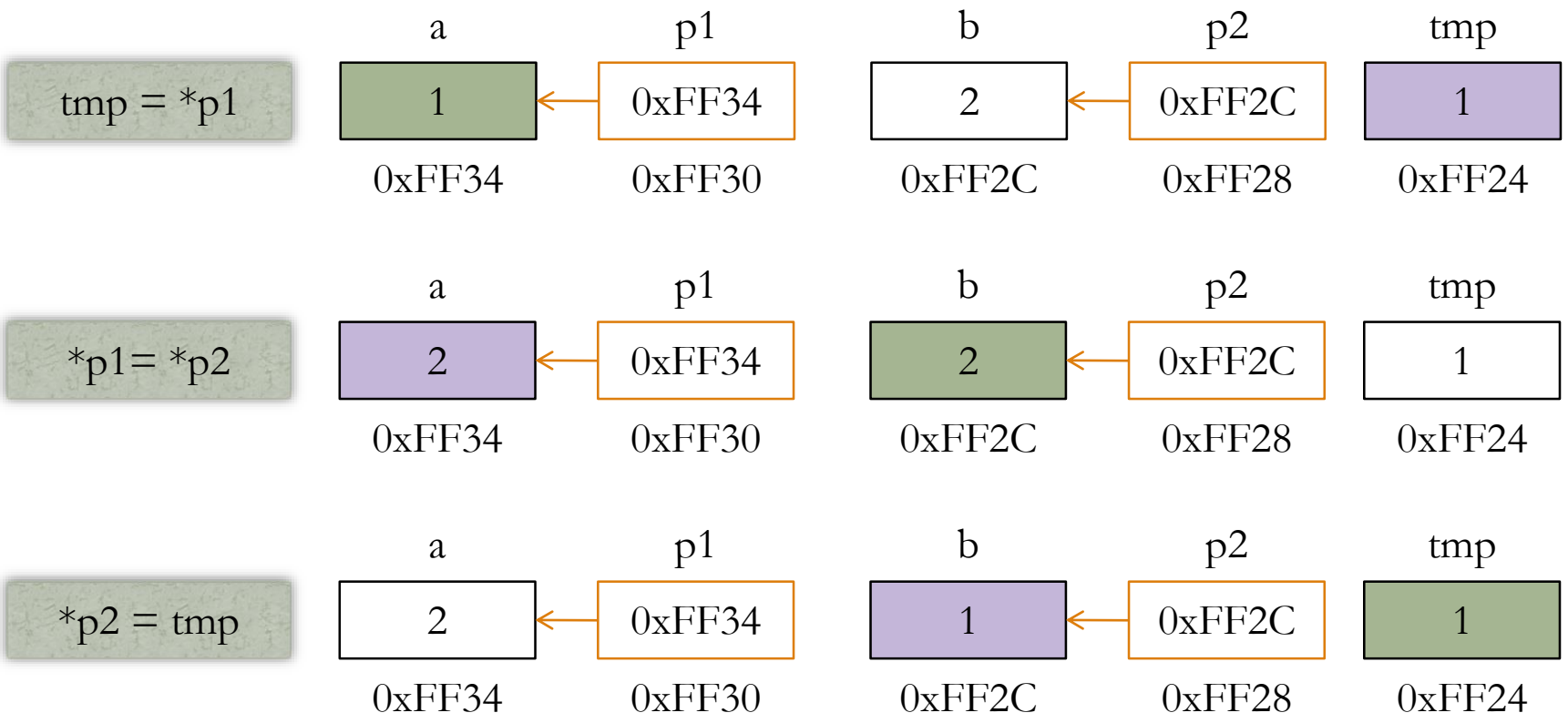
```
int main(void)
{
    int a=1, b=2;

    swap2(&a,&b);
    printf("%d %d",a,b);

    return 0;
}
```

```
void swap2(int *p1,int *p2)
{
    int tmp;
    tmp = *p1;
    *p1 = *p2;
    *p2 = tmp;
}
```

函數傳遞指標的範例(續)

```
tmp = *p1;
*p1 = *p2;
*p2 = tmp;
```

tmp = *(0xFF34);
*(0xFF34) = *(0xFF28);
*(0xFF28) = tmp;



函數傳回指標

- 傳回指標的函數

```
傳回值型態 *函數名稱(資料型態 引數)
{
  /* 函數的本體 */
}
```

傳回指標

```
int main(void)
{
    int *ptr,num;
    ptr=func(num);
    ...
}
```

接收函數所傳回的指標

```
int *func(int num)
{
      /* 函數的本體 */
}
```

傳回指向整數的指標

回傳內容 vs. 回傳位址

- 回傳內容

```
int main(void)
{
    int a = 3392;
    int b;
    b = afunc(a);
    printf("%d",b);
    return 0;
}
```

```
int afunc(int x)
{
    return x;
}
```

return 3392;

- 回傳位址

```
int main(void)
{
    int a = 3392;
    int *p;
    p = afunc(&a);
    printf("%p",p);
    return 0;
}
```

```
int *afunc(int *x)
{
    return x;
}
```

return 0x0022FF34

函數傳回指標範例

- 找出數值

```
int main(void)
{
  int a=1, b=2, c=0;
  int p;
  p = max1(&a,&b);
  printf("%d",p);
  return 0;
}
```

```
int max1(int *p1,int *p2)
{
  if (*p1 > *p2)
    return *p1;
  else
    return *p2;
}
```

- 找出位址

```
int main(void)
{
  int a=1, b=2, c=0;
  int *p;
  p = max2(&a,&b);
  printf("%d",*p);
  return 0;
}
```

```
int* max2(int *p1,int *p2)
{
  if (*p1 > *p2)
    return p1;
  else
    return p2;
}
```

指標與陣列

- 陣列名稱是一個指標常數，它指向(儲存)該陣列的起始位址
- 指標變數可以指向陣列的起始位址，或陣列任一元素的位址(運用指標＋/－運算)

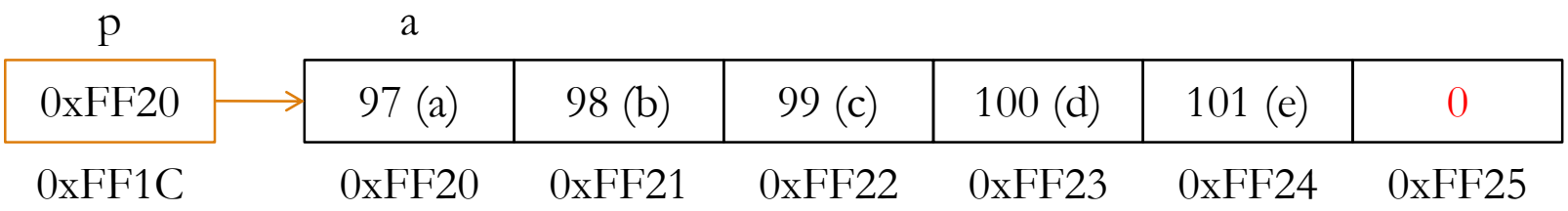
```
int a[] = {0,2,4,6,8};
int *p = a;
int i;

for (i=0;i<5;i++)
{
        printf("%p %p\n", &a[i], p+i);
        printf("%d %d %d\n", a[i], *(p+i), *p+i);
}
```

p[i]相當於*(p+i)

指標與一維陣列

- 指標可以指向並處理陣列
  - 指標與陣列的類型必須一致

```
char a[] = "abcde";
char *p;

for (p=a ; *p!=0 ; p++)
      printf("%c",*p);
```



指標與字串範例

- 字串列印

```
int main(void)
{
  char a[]={'A','B','C',0};
  char *p;

  p = a;
  printf("%s", p+1);

  return 0;
}
```

- 字串處理

```
int main(void)
{
  char a[]={'A','B','C',0};
  char *p;

  for (p=a;*p!=0;p++)
    *p = *p - 'A' + 'a';

  printf("%s", a);
  return 0;
}
```

指標與一維陣列範例

- 陣列計算

```
int main(void)
{
  int a[]={66,88,77,-1};
  int *p=a;
  int sum=0;

  for (p=a;*p>=0;p++)
    sum += *p;

  printf("%d",sum);
  return 0;
}
```

- 陣列搜尋

```
int main(void)
{
  int a[]={66,88,77,-1};
  int *p=a, *q=a;

  for (p=a;*p>=0;p++)
    if (*p>*q)
      q = p;

  printf("%d",*q);
  return 0;
}
```

指標、陣列、函數

- 傳入陣列，傳回數值

```
int main(void)
{
  char a[] = "12321";

  printf("%d",afunc(a));

  return 0;
}
```

```
int afunc(char *p)
{
  char *q;
  for (q=p;*q!=0;q++)
    ;
  return q-p;
}
```

- 傳入陣列，傳回指標

```
int main(void)
{
  char a[] = "12321";

  printf("%p",afunc(a));

  return 0;
}
```

```
char* afunc(char *p)
{
  char *q=p;
  for (;*p!=0;p++)
    if (*p>*q)
      q=p;
  return q;
}
```

指標與二維陣列

- 指標指向多維陣列需有相同基底

```
int a[][2]={{1,2},{3,4},{5,6}};
int (*p)[2]; //*p指向int[2]

printf("%d",a[0][1]);
printf("%d",(*(a+1))[1]);
printf("%d",*((*(a+2))+1));

p = a;
printf("%d",*(*p+1));
p++;
printf("%d",*(*p+1));
p++;
printf("%d",*(*p+1));
```

指標與二維陣列範例

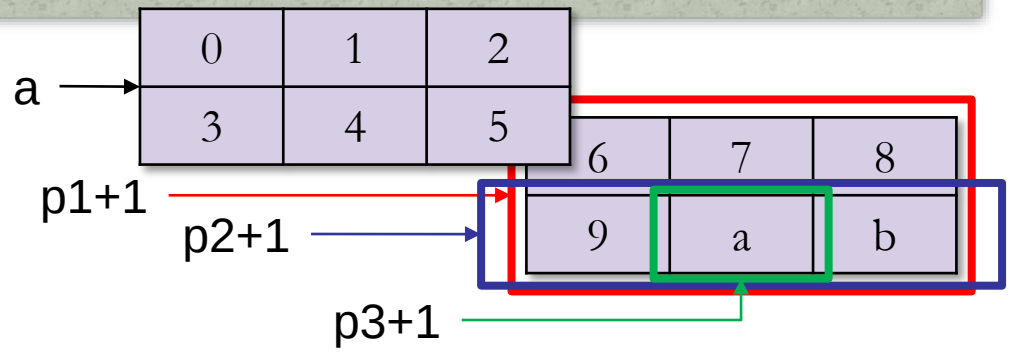
- 善用指向一維及二維陣列的指標

```
int a[][2]={{1,2},{3,4},{5,6}};
int (*p)[2];
int *q;
int i,j;

p = a;
for (i=0;i<3;i++)
{
  q = *p++;
  for (j=0;j<2;j++)
    printf("%d ",*q++);
  printf("\n");
}
```

指標與三維陣列

- 處理多維陣列需留意指標的參照層次

```
char a[2][2][3] = {{{'0','1','2'},{'3','4','5'}},
                   {{'6','7','8'},{'9','a','b'}}};
char (*p1)[2][3];
char (*p2)[3];
char *p3;

p1 = a;  // p1指向陣列a
p2 = *(p1+1);
p3 = *(p2+1);

printf("%c",*(p3+1));
```

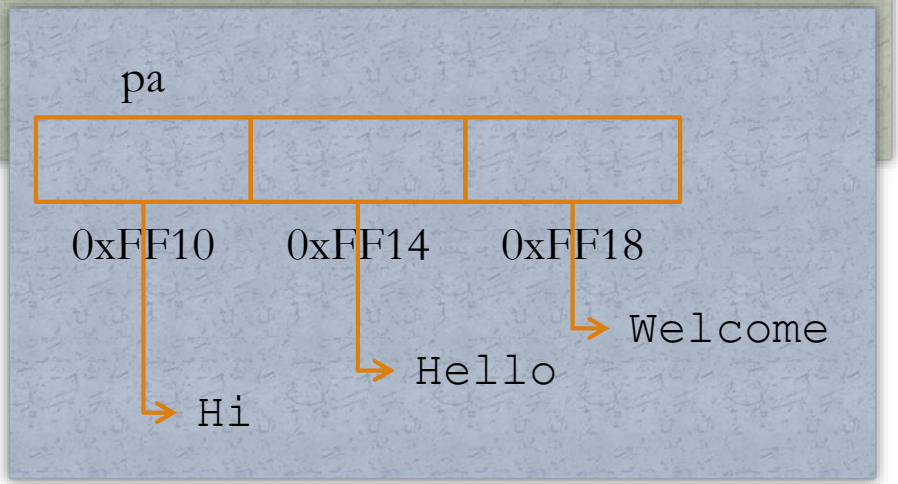


指標陣列

- 一維指標陣列(元素為指標)的宣告格式：

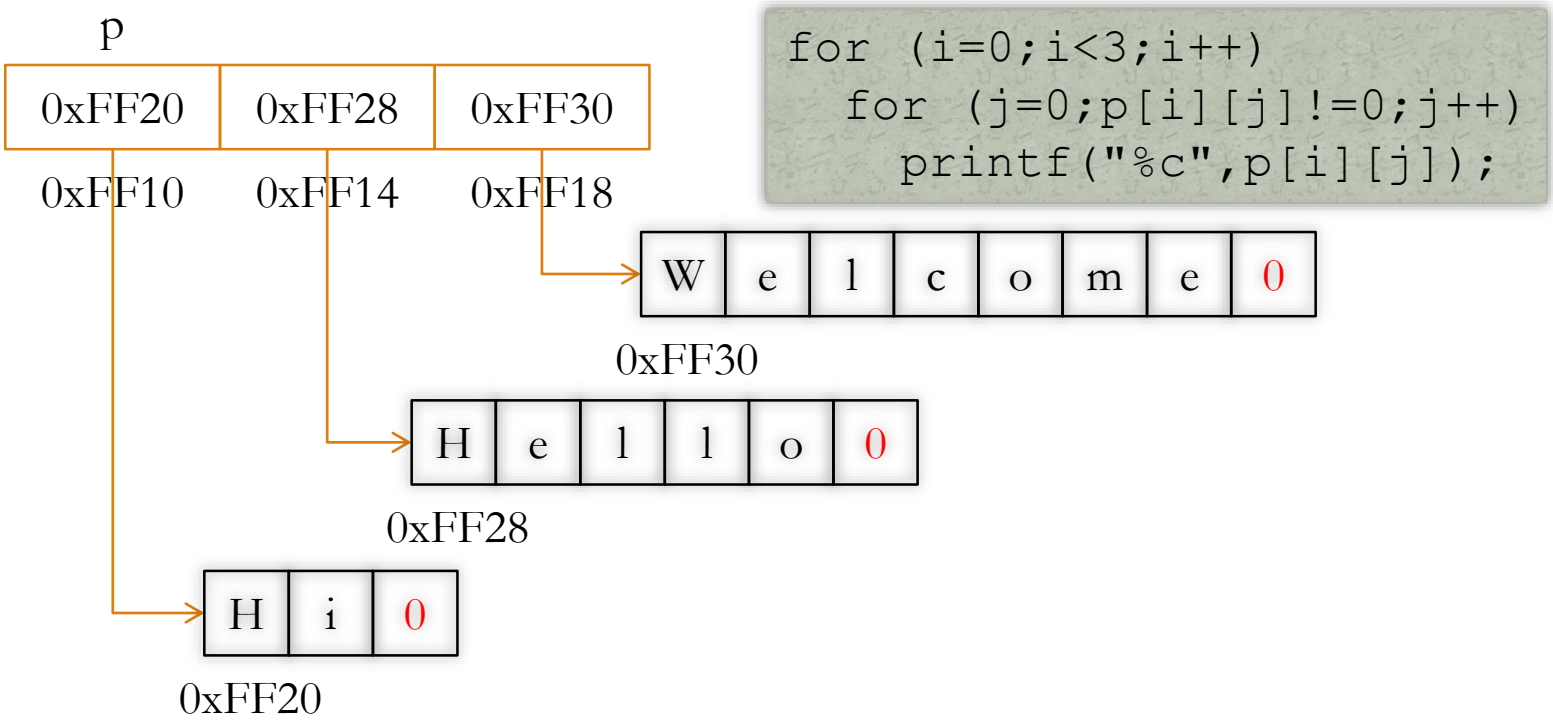
資料型態 *陣列名稱[元素個數];

```
char a[] = "Hi";
char b[] = "Hello";
char c[] = "Welcome";
char *p[] = {a, b, c};
int i;

for (i=0;i<3;i++)
  printf("%s\n",p[i]);
```



指標陣列 (續)

- 陣列p的元素為指標
  - p[i]：第i個字串的起始位址
  - p[i][j]：第i個字串的第j個字元



指標、陣列、指標陣列

- 宣告的差異
  - 指向陣列的指標
  - 由指標構成的陣列

```
int a[][2] = {{1,2},{3,4},{5,6}};

int *p1[2];    // 相當於 *(p1[2])
int (*p2)[2]; //宣告一個指標
int *(p3[2]); //宣告一個陣列，元素為指標

printf("%d",sizeof(p1));
printf("%d",sizeof(p2));
printf("%d",sizeof(p3));
```

指標與陣列探索

- 指標可以探索陣列的內容

```
int a[][2] = {{1,2},{3,4},{5,6}};
int *p,i;

p = (int *)a;  // p指向陣列a

for (i=0;i<6;i++)
  printf("%d",*p++);
```

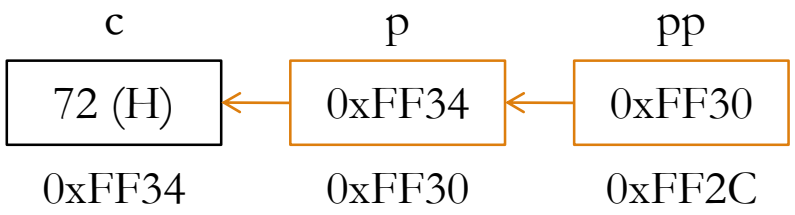
```
char *pa[] = {"Hi","Hello"};
char *p;

p = *pa;
printf("%c%c",*p,*(p+1));

p = *(pa+1);
printf("%c%c",*p,*(p+1));
```

指向指標的指標

- 指向指標的指標

資料型態 **雙重指標;

```
char c = 'H';
char *p = &c;
char **pp = &p;

printf("c=0x%p addr=0x%p\n",c,&c);
printf("p=0x%p addr=0x%p *p=%c\n", p, &p, *p);
printf("pp=0x%p addr=0x%p *pp=%p\n", pp, &pp, *pp);
```



Any question?