



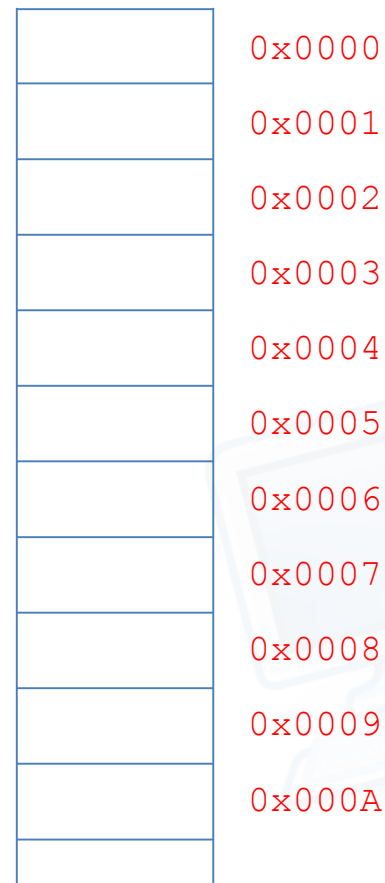
# 指標





# 記憶體與位址

- 在電腦中的記憶體就像可存放物品的一個個小格子。
- 每一個格子可以存於1 Byte的資料。
- 每個格子都是一直線的相鄰排列在一起，並擁有連續的編號---「位址」。





## 以存放一個 int 為例

- `int a = 74658;`

- 把 74658 轉成 2 進位

00000000 00000001 00100011 10100010

- 轉成 16 進位

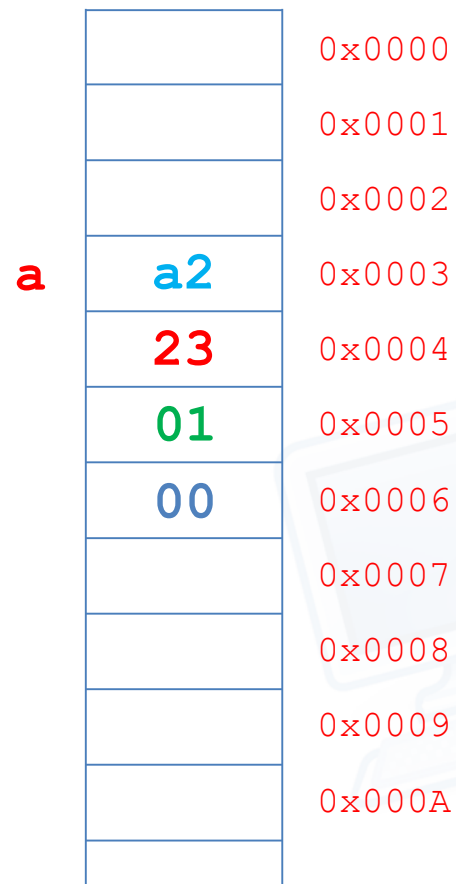
00 01 23 a2





# 以存放一個 int 為例

- `int a = 74658;`  
    00 01 23 a2
- 在程式載入後，a 可能會被安排在記憶體的任何地方。
- 每次執行時可能都在不同位置。





## & 取址運算子

- 使用 & 運算子，可以取得變數目前所在的記憶體位址

```
int main()
{
    int a = 74658;

    cout << &a << endl;
    cout << a << endl;

    return 0;
}
```

```
0x14dff0c
```

```
74658
```

```
Process returned 0 (0x0)   execution time : 0.026 s
```

```
Press any key to continue.
```



# 使用 debug 工具的 memory dump

The screenshot shows the Code::Blocks IDE interface. On the left, a terminal window displays the memory address `0x14dff0c`. The main IDE window shows a C++ program being debugged. A context menu is open over the code, with the "Memory dump" option highlighted. Below the IDE, a "Memory" window is open, showing the address `0x14dff0c` and a 32-byte dump. The dump shows the following data:

| Address   | Hex   | ASCII           |
|-----------|---|-----------------|
| 0x14dff0c | a2 23 01 00 90 6f 42 00   30 ff 4d 01 94 ff 4d 01 |  #...oB.0ÿM.ÿM. |
| 0x14dff1c | fd 10 40 00 28 ff 4d 01   34 9e 71 75 00 e0 fd 7e | ÿ.0.(ÿM.4qu.àÿ~ |



# 指標

- 指標也是一種型別
- 它不是用來存整數、浮點數.....
- 它是用來存放「位址」





# 指標

```
int main()
{
    int a = 74658;
    int *pa;

    pa = &a;
    cout << pa << endl;

    return 0;
}
```

0x14dff0c

Process returned 0 (0x0) execution time : 0.026 s  
Press any key to continue.

為什麼宣告指標型別的變數是

```
int *pa;
```

而不是

```
pointer pa;
```

為什麼要有個 `int` 在前面？







## 指標

- 儲存一個 int，需要 4 Byte
- 儲存一個 char，需要 1 Byte
- 如果我只知道有個變數放在 0x14dff0c .....
- 我怎麼知道該取出 4 Byte 還是 1 Byte 的資料來使用？





## 指標

- 儲存一個 int，需要 4 Byte
- 儲存一個 float，需要 4 Byte
- 雖然都是 4 Byte，但格式不同
- 我怎麼知道要把這塊記憶體解讀為 int 還是 float？





# 補充: float 格式

- 176.375

$$= 1*2^7 + 0*2^6 + 1*2^5 + 1*2^4 + 0*2^3 + 0*2^2 + 0*2^1 + 0*2^0 + 0*2^{-1} + 1*2^{-2} + 1*2^{-3}$$

$$= (10110000.011)_2$$

$$= (1.011000011)_2 * 2^7$$

– 正負號(1 bit) : + → 0

– 指數(8 bit) : 7 → 00000111

– 有效數(23 bit) : 01100000110000000000000

- 00000011 10110000 01100000 00000000





## 指標的宣告

- 指標也是一種變數，只是它存的是「位址」
- 指標在宣告時一定要說明是指向「什麼型別」的指標

int \*pa;

這個指標儲存的位址  
是放什麼型別的資料

變數名稱





## 那顆星號 (\*)

- \* 出現在宣告的地方，表示你宣告的變數是一個「指標」
- 出現在其他地方的指標前，表示取值/提領(dereference)





# 那顆星號 (\*)

```
int main()
{
    ▶ int a = 74658;
      int *pa;

    pa = &a;
    cout << pa << endl;
    cout << *pa << endl;

    return 0;
}
```

0x14dff0c  
74658

|   |    |            |
|---|----|------------|
|   |    | 0xa3000000 |
| a | a2 | 0xa3000001 |
|   | 23 | 0xa3000002 |
|   | 01 | 0xa3000003 |
|   | 00 | 0xa3000004 |
|   |    | 0xa3000005 |
|   |    | 0xa3000006 |
|   |    | 0xa3000007 |
|   |    | 0xa3000008 |
|   |    | 0xa3000009 |
|   |    | 0xa300000A |



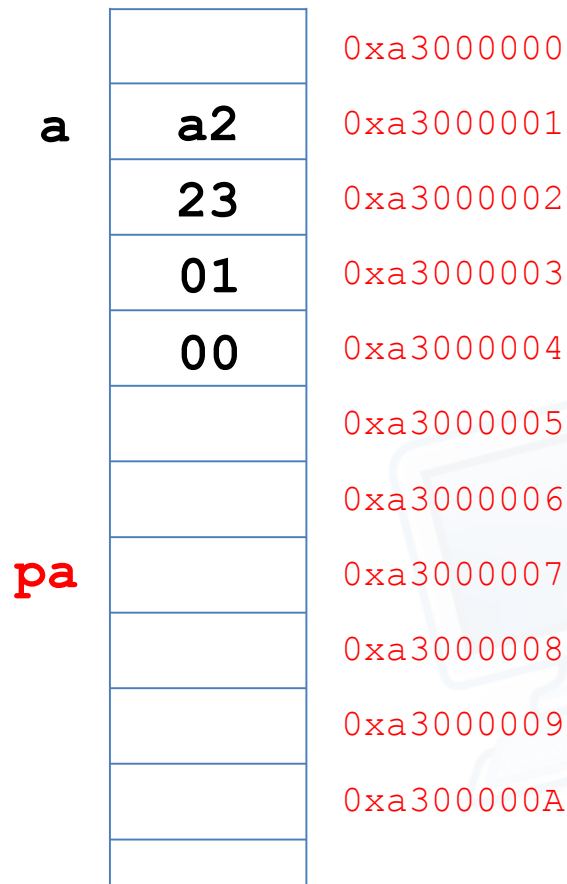
# 那顆星號 (\*)

```
int main()
{
    int a = 74658;
    ▶ int *pa;

    pa = &a;
    cout << pa << endl;
    cout << *pa << endl;

    return 0;
}
```

0x14dff0c  
74658





# 那顆星號 (\*)

```

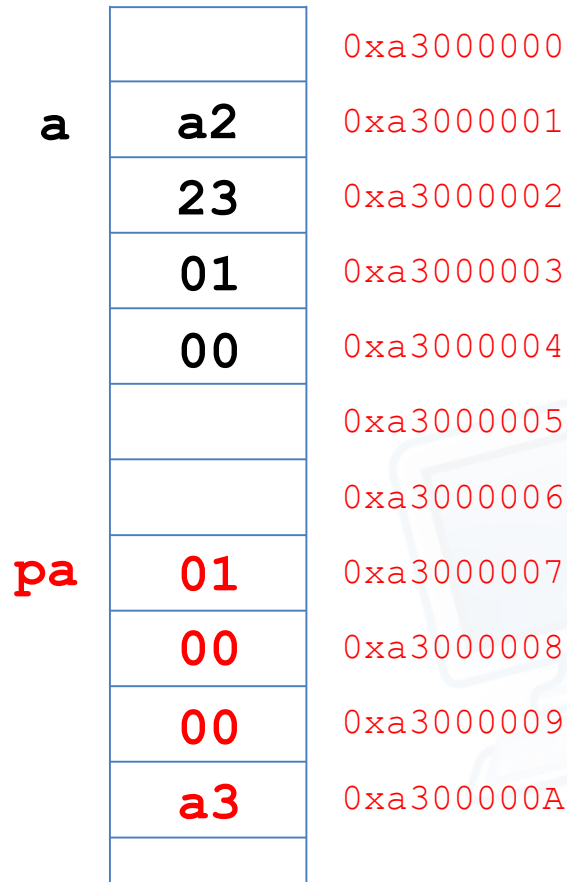
int main()
{
    int a = 74658;
    int *pa;

    ▶ pa = &a;
    cout << pa << endl;
    cout << *pa << endl;

    return 0;
}

```

0x14dff0c  
74658







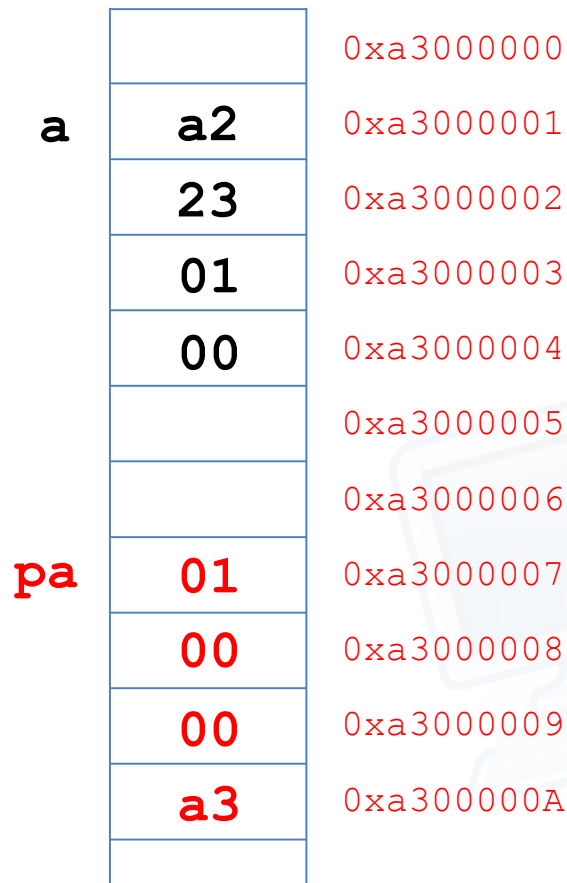
# 那顆星號 (\*)

- 我們可以取得 `pa` 所指向位址的值

```
cout << *pa;
```

- 也可以改變 `pa` 所指向位址的值

```
*pa = 8;
```





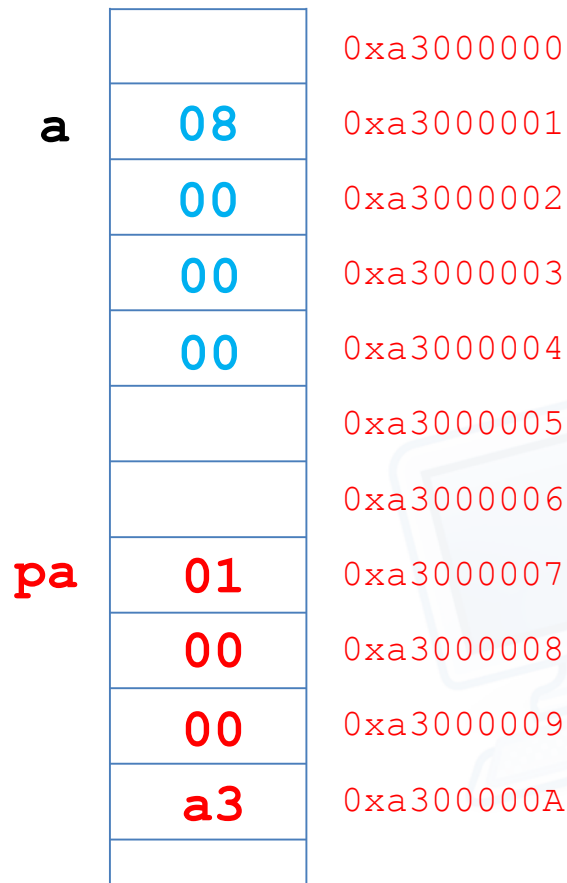
# 那顆星號 (\*)

- 我們可以取得 pa 所指向位址的值

```
cout << *pa;
```

- 也可以改變 pa 所指向位址的值

```
*pa = 8;
```





# 這有什麼用？

- 可以解決這個問題

```
void swap(int a, int b)
{
    int c = a;
    a = b;
    b = c;
}

int main()
{
    int a = 3, b=5;

    swap(a, b);
    cout << a << ", " << b;

    return 0;
}
```

3, 5





```
void swap(int a, int b)
{
    int c = a;
    a = b;
    b = c;
}

int main()
{
    int a = 3, b=5;

    swap(a, b);
    cout << a << ", " << b;

    return 0;
}
```

3, 5

```
void swap(int *a, int *b)
{
    int c = *a;
    *a = *b;
    *b = *c;
}

int main()
{
    int a = 3, b=5;

    swap(&a, &b);
    cout << a << ", " << b;

    return 0;
}
```

5, 35