

slido:~\$



# 指標變數

指標變數：~ 指標變數 / 前言 \$

以下是我輸出 arr 的結果，獲得了 0x7ffe2c436a60 的輸出  
這就是一個記憶體位置，也就是 **arr 是一段記憶體位置**

(更準確的來講，其實是 arr[0] 的記憶體位置)

```
int arr[5]={4, 8, 7, 6, 3};  
cout << arr << "\n";
```

```
// output: 0x7ffe2c436a60  
// (每個人的輸出不會一樣)
```

指標變數：~ 指標變數 / 前言 \$

這句話非常特別

代表著 arr 實際上存的不是整數、字串、浮點數是一個整數的記憶體位置

等價於有一種變數可以儲存整數的記憶體位置

```
int arr[5]={4, 8, 7, 6, 3};  
cout << arr << "\n";
```

```
// output: 0x7ffe2c436a60  
// (每個人的輸出不會一樣)
```

## 指標變數：~ 指標變數 / 前言 \$

以上正是指標變數的概念，簡單來說，就是一種型別可以用來儲存記憶體

`int` -> 儲存整數

`string` -> 儲存字串

[某個型別] -> 儲存記憶體

指標變數 : ~ 指標變數 / 宣告變數 \$

在 C++ 中，這種型別必須非常詳細

也就是你必須告訴程式這是**何種型別的記憶體**

儲存一個整數的記憶體 -> 宣告儲存 **整數記憶體** 的變數

儲存一個字串的記憶體 -> 宣告儲存 **字串記憶體** 的變數

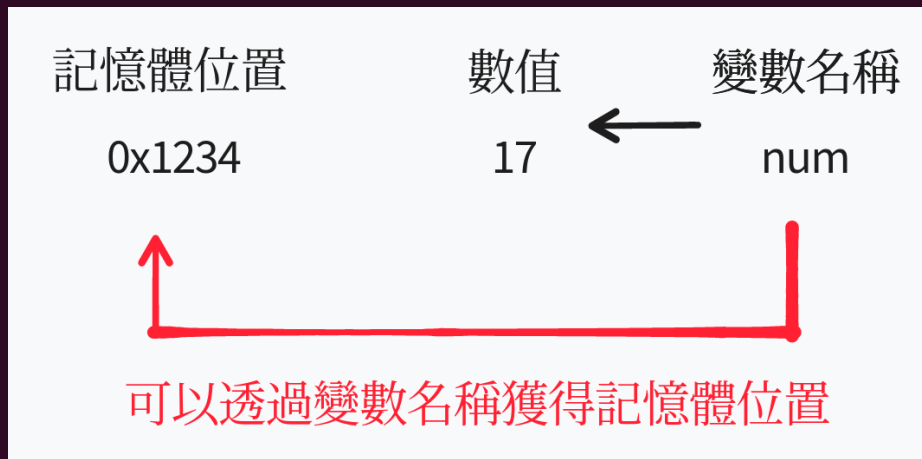
## 指標變數 : ~ 指標變數 / 宣告 \$

在宣告中，只要把原本型別加上一個 \*  
這種變數就可以儲存該型別的記憶體

```
int main(){  
  
    // variable declare  
    int n;  
    string s;  
    vector<int> v;  
    segment_tree st;  
  
    // pointer declare  
    int *np;  
    string *sp;  
    vector<int> *vp;  
    segment_tree *stp;  
  
    return 0;  
}
```

指標變數：~ 指標變數 / 賦值 \$

請記得，指標變數儲存的值為記憶體  
對普通變數 & 可以獲得該變數的記憶體





## 指標變數：~ 指標變數 / 賦值 \$

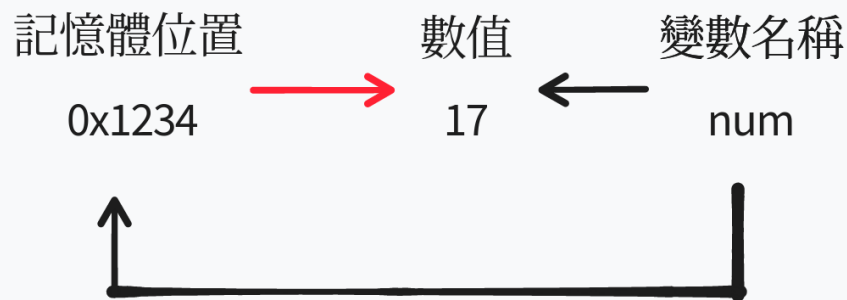
```
int main(){  
  
    // variable declare  
    int n;  
    string s;  
    vector<int> v;  
    segment_tree st;  
  
    // pointer declare  
    int *np = &n; // 宣告一個儲存整數的變數，裡面放了整數：n的記憶體位置  
    string *sp = &s; // 宣告一個儲存字串的變數，裡面放了字串：s的記憶體位置  
    vector<int> *vp = &v; // 宣告一個儲存vector<int>的變數，裡面放了vector<int>：v的記憶體位置  
    segment_tree *stp = &st; // 宣告一個儲存segment_tree的變數，裡面放了segment_tree：st的記憶體位置  
  
    return 0;  
}
```

指標變數：~ 指標變數 / 取值 \$

請記得，指標變數儲存的值為記憶體

對指標變數 \* 可以獲得該指標變數所指向的值

可以透過記憶體位置獲得數值



## 指標變數 : ~ 指標變數 / 取值 \$

```
int main(){  
  
    // variable declare  
    int n=17;  
    string s="uwu";  
    vector<int> v;  
    segment_tree st;  
  
    // pointer declare  
    int *np = &n;  
    string *sp = &s;  
    vector<int> *vp = &v;  
    segment_tree *stp = &st;  
  
    // get value  
    cout << *np << "\n";  
    cout << *sp << "\n";  
  
    return 0;  
}
```

```
~/D/G/code ↗ *... ./main  
17  
uwu
```

指標變數：~ 指標變數 / 應用 \$

為何不直接使用變數呢？主要是因為未來內容的： 函式  
可以透過指標變數直接修改值或是直接存取

聽不懂嗎？等下禮拜吧 :D

## 指標變數：~ 指標變數 / 結語 \$

以上就是指標變數的**基本概念**，我們只有講到非常皮毛的部份  
其他部份可以在題單找到詳細的介紹影片

並且，對於未來的 STL，也有類似的東西叫做 iterator  
不論是在考檢定或是競程上也是有作用的  
因此在此時就得先建立起記憶體的概念