Chapter 5
Arrays

Yu-Hsiang Cheng
鄭宇翔 (Slighten)
GOTC Instructor
http://m101.nthu.edu.tw/~s101021120/Myweb/index_ch.html

Recall I

Variables (變數)

- 讓電腦記住資料 (data)
- 有型別、有名稱、佔記憶體空間
- `int` vs. `const int` vs. `#define`

Operators (運算子)

- 對 data (變數、常數) 的操作 (manipulation)
- 有功能、有優先次序 (precedence)、順序關聯性 (associativity)
- Type-specific -- 不同型別的 data，操作會不同
- 事實上是一個 function -- 可自行定義自己的運算子 (taught in Ch7)

Control Structures (控制結構)

- 讓電腦能做「選擇」與「重複」
- 使程式結構化 (structured)
- Conditional (Selection) -- `if`, `else`, `switch`
- Iterative (Repetition) -- `while`, `for`, `do-while`
- `break` vs. `continue`

Recall II

Functions (函式)

- 使程式結構化 (structured)
- declaration, definition, call
- 增加「可讀性」與「可重複利用性」
- 傳值 vs 傳址(傳指標) vs 傳參考
- scope, lifetime
- heap vs. stack
- global vs. local
- static global vs. global
- static local vs. (auto) local

小技巧：當你發現寫一寫有兩段程式碼很像
(e.g. 第5~10與第11~15行很像)，可考慮使用迴圈或函式簡化之。

Arrays (陣列)

- Q: 如何存放一整串連續的資料？
  - character string (一整串字元所形成的字串)
  - table of numbers (一整列數字(向量))
- A1: 這樣如何？

```
int num0;
int num1;
int num2;
int num3;
```

- 不差，但...
  1. 如果有 100, 甚至 1000 個數字呢？
  2. 如何用一個 loop 去改所有變數的值？
- A2: A better way -- the *array*.

```
int num[4];
```

- 用上述方式宣告一串 4 個整數的陣列，以下存取值:
  - `num[0], num[1], num[2], num[3].`

Array Syntax

- Declaration (宣告)

```
type array_name[num_elements];
```

array 的每個元素都相同 type！

編譯時須知元素的個數(以判斷要給這個陣列多少記憶體空間)

- Array Reference (參考): 取出陣列的某個元素

```
array_name[i];
```

這個陣列的第 i 個元素 (從 0 開始數)
no limit checking at compile-time or run-time
(可能取超過範圍 => may segmentation fault)

Question

- Array 的每個元素都相同 type！
- 那如果希望每個元素不同 type 呢？
  - 要用到 struct (結構)

Struct Syntax

- Definition

```
struct struct_name {
    char a;
    int b;
    double c;
};
```

- Declaration

```
struct struct_name variable_name;
```

- Definition & Declaration at Once

```
struct struct_name {
    char a;
    int b;
    double c;
} variable_name;
```

- Struct Reference (Member Access)(用運算子 '.')

```
variable_name.a = 'c';
variable_name.b = 10;
variable_name.c = 3.5;
```

Array Examples

```
int A[10]; /* array of ten integers */
A[0] = 42; /* access to elements */
A[1] = 100;
A[2] = A[0] + 5 * A[1];
```

Why these garbage values?
因為 local 變數是放在 stack 上，
可能有之前的資料未清空。
=> 要用之前必先初始化

- Array indexing
  - 從 0 開始
  - 最後一個元素是第 n-1 個

A	0	1	2	3	4	5	6	7	8	9
	42	100	542	-1	44	55	0	0	0	99

Accessing array using for loop

```
int A[10]; /* array of ten integers */
for (int i = 0; i < 10; i++)
    A[i] = i * 10 + i;
for (int i = 0; i < 10; i++)
    cout << A[i] << ' ';
```

A

0	1	2	3	4	5	6	7	8	9
0	11	22	33	44	55	66	77	88	99

Array bounds checking

- C++ compiler 不會檢查陣列邊界 (編譯不會錯誤)
  => 你會存取到不該存取的記憶體位置

```
int A[10]; /* array of ten integers */
int i;
for (i = -5; i < 20; i++)
     cout << (A[i] = i) << ' ';
```

有三種可能

1. 沒事，正常使用 (Dev-C++)
2. 執行發生錯誤，訊息是 segmentation fault (core dumped): outside of your address space
3. 執行發生錯誤，訊息是 bus error: trying to read a long from an address which isn't a multiple of 4

- 差異？http://stackoverflow.com/questions/838540/bus-error-vs-segmentation-fault

Array Initialization (初始化)

- Use for loop

```
int A[10]; /* array of ten integers */
int i;
for (i = 0; i < 10; i++)
    A[i] = 0;
```

Array Declaration & Initialization (宣告時初始化)

- 用 { }　// initializer-list

```
int A[10] = {42, 100, 310, 44, 55, 0, 3, 4, 0, 99};
```

- 事實上，若宣告同時初始化，可以省略 size (編譯器會幫你算)

```
int A[] = {42, 100, 310, 44, 55, 0, 3, 4, 0, 99};
```

- 若只初始化前面，其後都會是 0

```
int A[10] = {42, 100, 310}; /* only 3 initialized; others = 0 */
```

- Question: 如何宣告有 100 個元素的整數陣列並通通初始化成 0？

答：

```
int A[100] = {0};
```

Multi-dimensional Arrays

- 可以宣告多維陣列
- C++ 是 "Row major" convention:

```
int M[3][2];
```

- 3 rows(列), 2 columns(行)之矩陣(2 維陣列).

M[0][0]	M[0][1]
M[1][0]	M[1][1]
M[2][0]	M[2][1]

- 初始化
  - 利用雙層 for loop
- 宣告時初始化

1	3
2	4
3	6

1. 未定維度，須用巢狀

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

2. 若給定維度則不必巢狀 {}

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

一定得給

Array

- C array
- C++11 array-like data structure (OOP, will be included in Ch7)
  - vector (vs list) (#include <vector>)
    - http://www.cplusplus.com/reference/vector/vector/
  - array (#include <array>)
    - http://www.cplusplus.com/reference/array/array/

- C++ vs C++11
  - http://www.codeproject.com/Articles/570638/Ten-Cplusplus-Features-Every-Cplusplus-Developer

String

- C string – `char` array
  - more complex, yet detailed
  - 操作 string 的 function 要 `#include <cstring>`
- C++ string – `string` (`#include <string>`)
  - easier, and more powerful
  - string 是一個 class
  - 有很多 member functions (包括 operators) 已定義好，直接拿來用就好，不必我們再寫一次。
  - `+ (append()), = (assign()), swap(), etc.`
  - http://www.cplusplus.com/reference/string/string/

A String is an Array of Characters

- 利用創造字元陣列的方式來存放字串

```
        char outputString[16];
```

- 宣告並初始化：

```
char outputString[16] = "Result = ";
```

  - 上列是以下方式的速寫法

```
char outputString[16] = {
    'R', 'e', 's', 'u', 'l', 't', ' ',
    '=', ' ', '\0'
};
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
'R'	'e'	's'	'u'	'l'	't'	' '	'='	' '	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'	'\0'

A String is Ended by a Terminating Zero

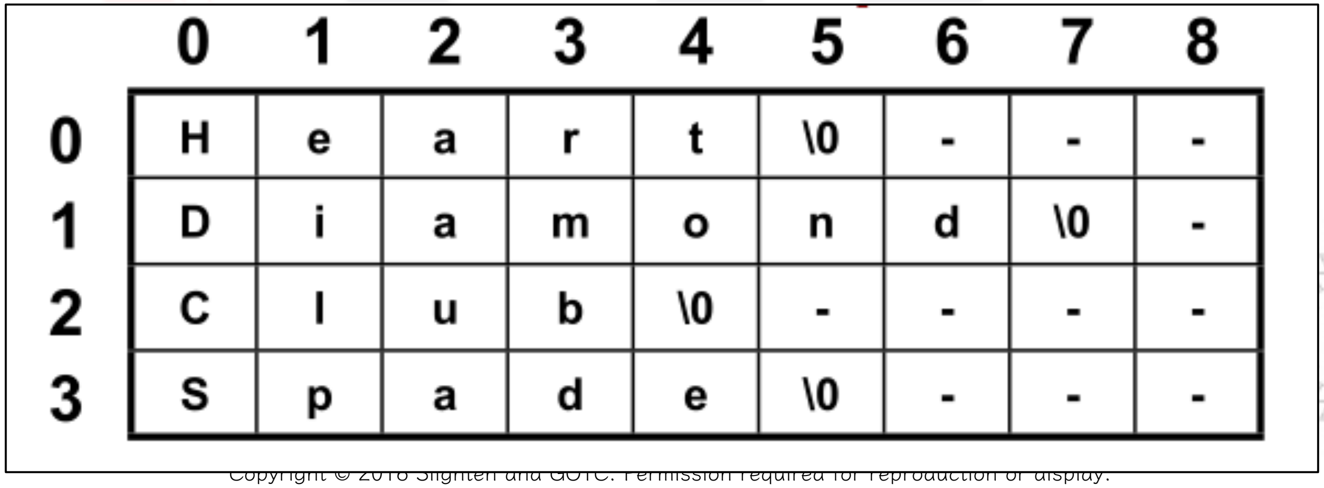
- 記得字串是用 `'\0'` 做結尾
- 所以要留一格放 `'\0'`
- 意旨 `char str[n];` 只能放 n-1 個字元(`str[0]~str[n-2]`)
- 最後一格 `str[n-1]` 放 `'\0'`
- 所以 `char str[5] = "apple";` 會錯(Dev-C++: compile error)

- Question:
- `char str[] = "abc";`
- `sizeof(str) is?` 4
- `char str2[10] = "abc";`
- `what are str2[3], str2[4] ~ str2[9]?` `'\0'`

An Example of a 2D Array

- An array of strings = A 2D array of chars

```c
char matrix[4][9] = {
   /* matrix of 4 rows of 9-char buffers */
   "Heart", "Diamond", "Club", "Spade"
};
```



	0	1	2	3	4	5	6	7	8
0	H	e	a	r	t	\0	-	-	-
1	D	i	a	m	o	n	d	\0	-
2	C	l	u	b	\0	-	-	-	-
3	S	p	a	d	e	\0	-	-	-

Can't initialize an array to the content of another array

- 可允許的
  - `char x[20] = "string literal.";`
  - `char y[20] = {x[0], x[1], x[2], ... x[19]};`
- 「不」允許
  - `char y[20] = x;`

- 為什麼？
  - 因為 C++ compiler 不知道陣列 x 的長度
- 為什麼不知道？
  - 因為陣列「名稱」只代表該陣列的起始位置！ (常數字串本身也是)
  - 起始位置代表第 0 個元素的記憶體位置
  - 意旨： `x == &x[0]`
  - `cout << (x == &x[0]); is 1`
  - 所以 array 只有減法算術運算

Can't assign an array using the = operator

- 也不能
  - `int y[20];`
  - `y = x;`

- 為什麼?
  - 因為 y (as an expression) 是 *immutable* (i.e., 不能被修改)
  - 就好像 `20 = x;`
- 哪些其他是 immutable? (不能做)
  - `"hello" = "world";     // location not mutable`
  - `{int a = 3; &a = 100;} // location not mutable`
  - `"hello"[1] = 'a';      // content not mutable`

Copying Arrays with known length (1/2)

- Easy: a for loop

```
int source[MAX], dest[MAX];
for (int i = 0; i < MAX; i++)
    dest[i] = source[i];
```

- 自訂 function

```
void assignIntArray(int source[], int dest[], int size)
{
    for (int i = 0; i < size; i++)
        dest[i] = source[i];
}
```

Copying Null–terminated String

```
void assignString(char source[], char dest[]) {
    int i = 0;
    do {
        dest[i] = source[i];
    } while (source[i++]);
}
```

- How this works:
  - 至少複製 the `'\0'` terminating character
    => use `do-while` loop.
  - 當在 source array 遇到 `'\0'` 跳出 loop .
  - 但若 dest array 的長度比 source array 短呢？
  - 安全起見，改成 `while (i < MAX && source[i++]);`
  - `(#define MAX 30)`

New: Copying Arrays with known length (2/2)

- Alternative way: use `memcpy();`
- `#include <cstring>`

```
memcpy(dest, src, bytes);
```

- `char dest[] = "abcde", src[] = "ggg";`
- `memcpy(dest, src, 3);`
- `cout << dest << endl;`
- gggde
- http://www.cplusplus.com/reference/cstring/memcpy/

Common Pitfalls with Arrays in C++

- Overrun array limits
  - C++ compiler 不會檢查陣列邊界 (編譯不會錯誤)
  - compile–time 跟 run–time 都不會檢查

```
int array[10];
for (int i = 0; i <= 10; i++) array[i] = 0;
```

- `i <= 10` 應改成 `i < 10`

Array Size can be determined at Run–time

- 陣列當參數可以不給 size 或給 constant size
  - i.e., ok to say

```
void someFunction(int A[MAXSIZE]);
     void someFunction(int A[ ]);
```

  - but NOT ok to say

```
void someFunction(int size, int A[size]);
```

- 區域陣列可以有執行時再決定的 size (但這是不好的寫法)

```
void someFunction(int num_elements) {
  int temp[num_elements];
  for (int i = 0; i < num_elements; i++) temp[i] = i;
}
```

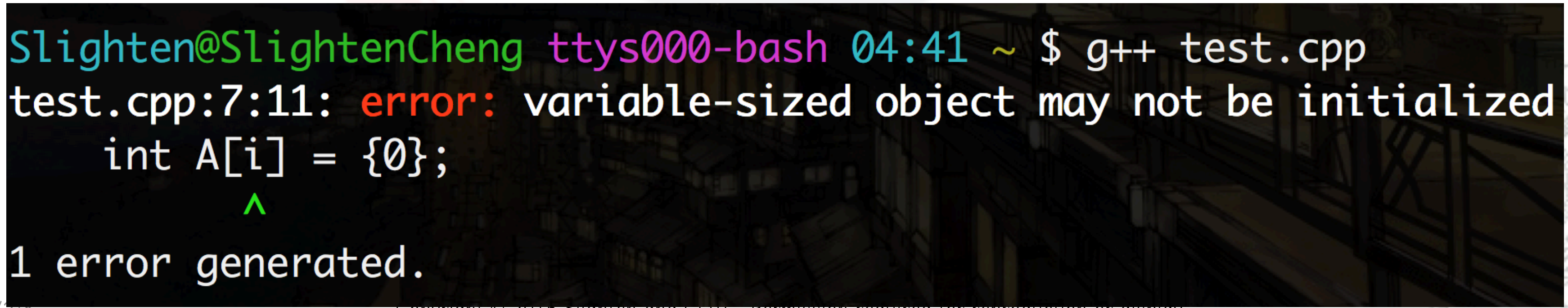
determined by actual parameter at runtime. (runtime constant)

New: Common Pitfalls II (1/3)

- 當 array size 是變數時，不能宣告並初始化

```
int i;
cin >> i;
int A[i] = {0};          // compile error
```

- 原因是編譯器不知道要給多大空間。
- 同理 `int arr[];` 也不行
- 但 `int arr[] = {0}; char arr2[] = "";` OK



```
Slighten@SlightenCheng ttys000-bash 04:41 ~ $ g++ test.cpp
test.cpp:7:11: error: variable-sized object may not be initialized
    int A[i] = {0};
          ^
1 error generated.
```

New: Common Pitfalls II (2/3)

- Solution I: 用 for-loop 或 memset();

```
int i;
cin >> i;
int A[i];
for (int j = 0; j < i; j++) A[j] = 0;
```

- Solution II: 用 memset(); (byte-wise，因此只能給 0 或 -1)
  - #include <cstring>

```
int i;
cin >> i;
int A[i];
memset(A, 0, sizeof(A));  // sizeof(A) == i*sizeof(int)
```

- http://www.cplusplus.com/reference/cstring/memset/

New: Common Pitfalls II (2/3)

- Solution III: 用動態 array (C: malloc(), calloc(), free(); C++: new, delete)

```
int i;
cin >> i;
int *A = new int[i]();              // all initialized to 0
/* … */
delete [] A;                        // 當用完後，程式結束前
```

- http://stackoverflow.com/questions/808464/c-new-call-that-behaves-like-calloc

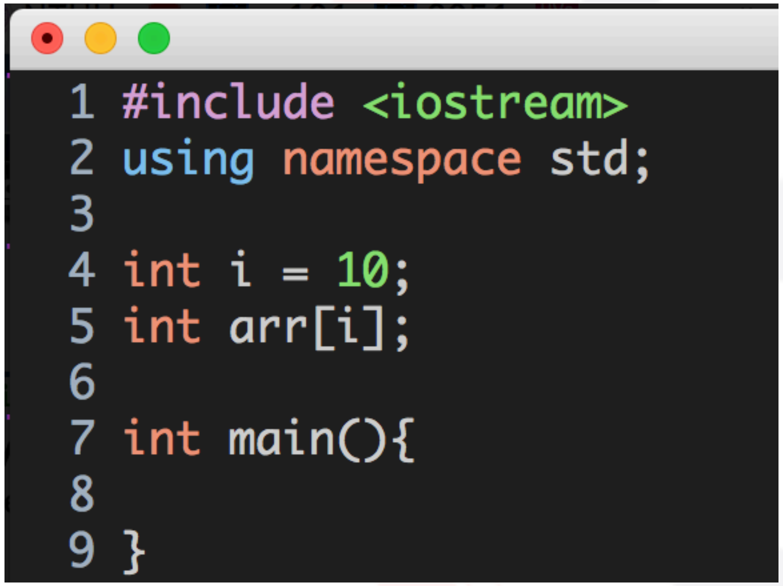
- Solution IV: 用動態 array (C++ STL: vector)
  - #include <vector>

```
int i;
cin >> i;
vector<int> A;
A.resize(i);
```

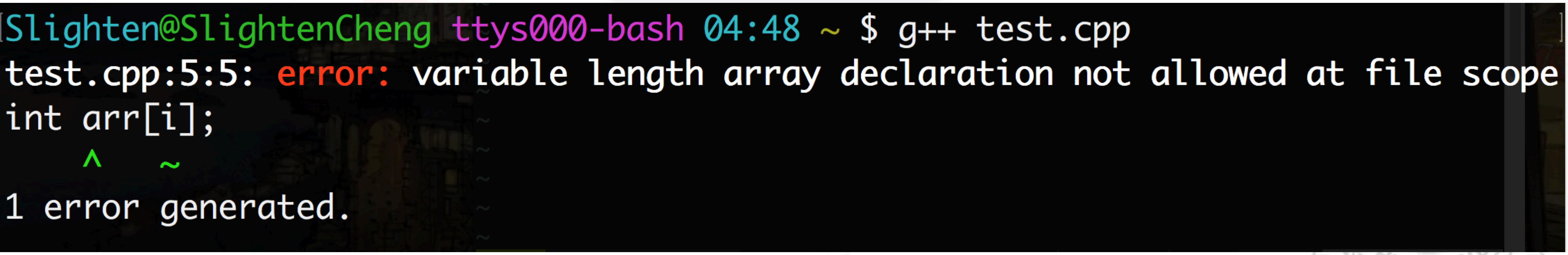
- http://www.cplusplus.com/reference/vector/vector/resize/

New: Common Pitfalls II (3/3)

- global 當然可以宣告 array，但不可以放變數 size



```cpp
#include <iostream>
using namespace std;

int i = 10;
int arr[i];

int main(){

}
```



```
Slighten@SlightenCheng ttys000-bash 04:48 ~ $ g++ test.cpp
test.cpp:5:5: error: variable length array declaration not allowed at file scope
int arr[i];
    ^   ~
1 error generated.
```

I/O with Strings

```
char inputString[100];
cin >> inputString;
/* 吃到空白、tab或換行時停止，最後加上一個
'\0' */
cout << inputString;
/* 印到 '\0' 停止 */
```

- input: hello world!
- output: hello

Passing parameters by reference vs. by value

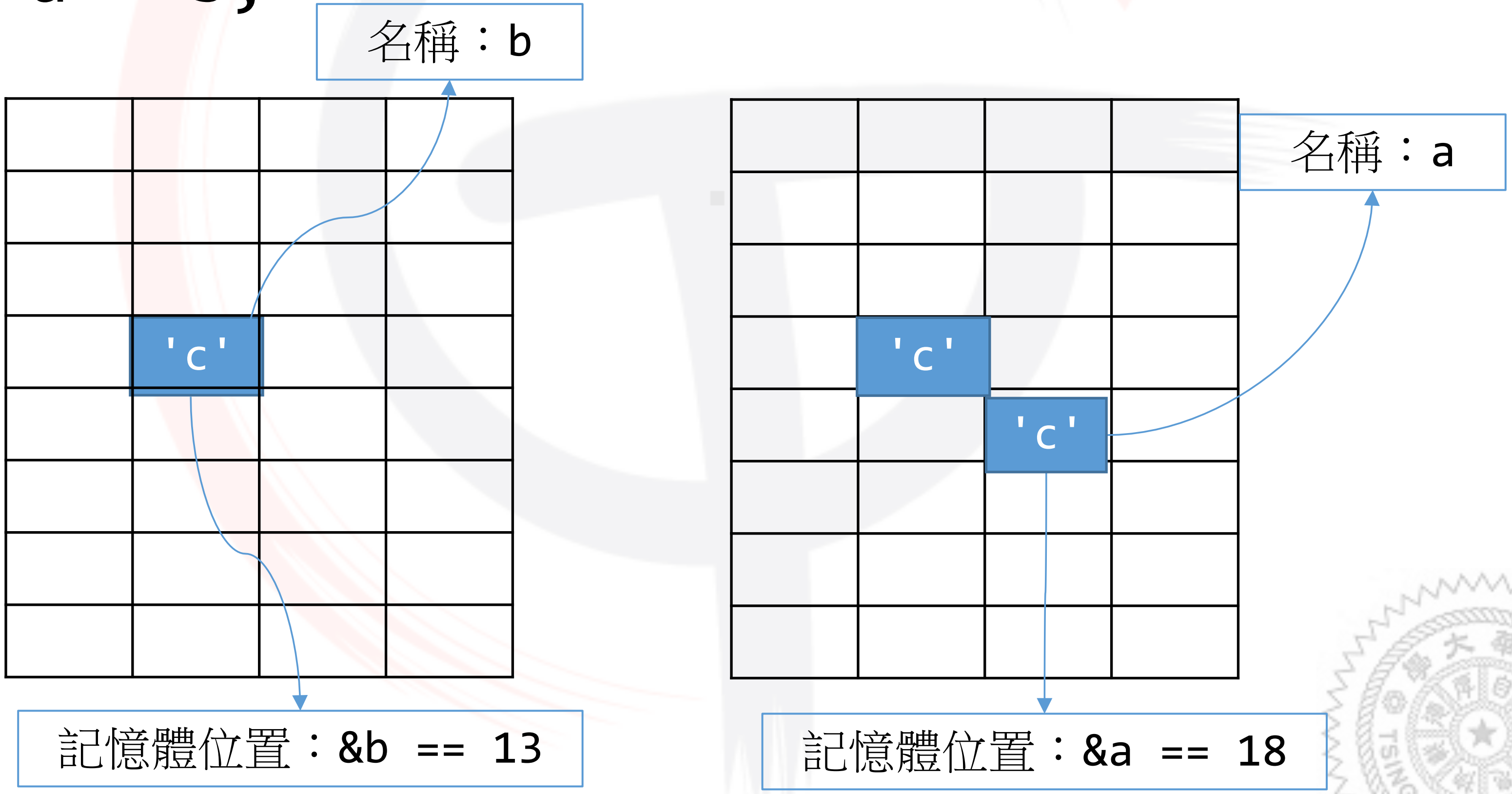
- C++ passes arrays *by reference*, rather than *by value*.
- Passing by value (傳值)
  - `int formal = actual;`
  - 形式參數得到真實參數的值的 copy
  - 形式參數是與真實參數不同的、額外的新變數
  - 對形式參數的改變不影響真實參數
  - scalars (`char`, `int`, `double`) and structs (`struct`)
- Passing by reference （傳參考）
  - `int &formal = actual;`
  - 形式參數是真實參數的參考(i.e. 別名)
  - 形式參數與真實參數是相同的變數！
  - 對形式參數的改變就是對真實參數的改變！

New: Passing parameters by address vs. by value

- C++ passes arrays *by reference*, rather than *by value*.
- Passing by value (傳值)
  - `int formal = actual;`
  - 形式參數得到真實參數的值的 copy
  - 形式參數是與真實參數不同的、額外的新變數
  - 對形式參數的改變不影響真實參數
  - scalars (`char`, `int`, `double`) and structs (`struct`)
- Passing by address （傳址、傳指標)
  - `int *formal = &actual;`
  - 形式參數拿到真實參數的地址
  - 對形式參數的改變就是對真實參數的改變！
  - arrays (`char []`, `int []`…)

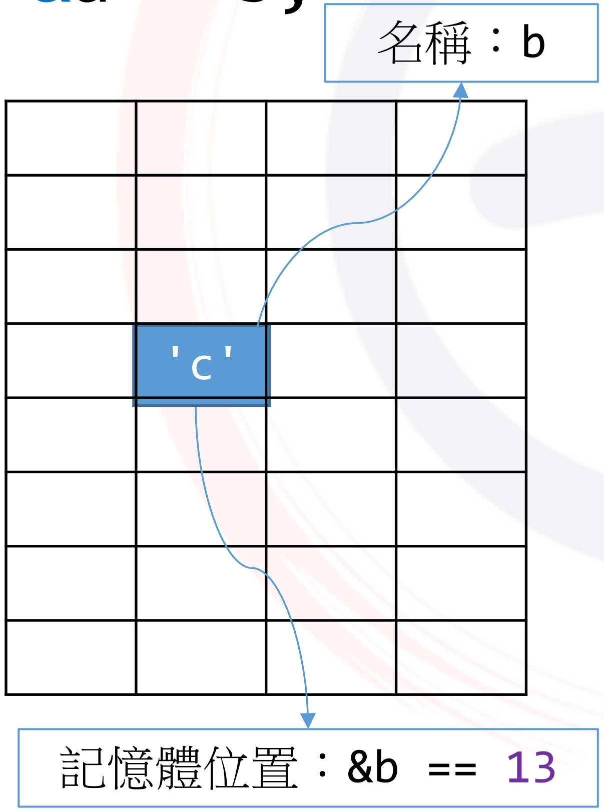
圖示 (傳值)

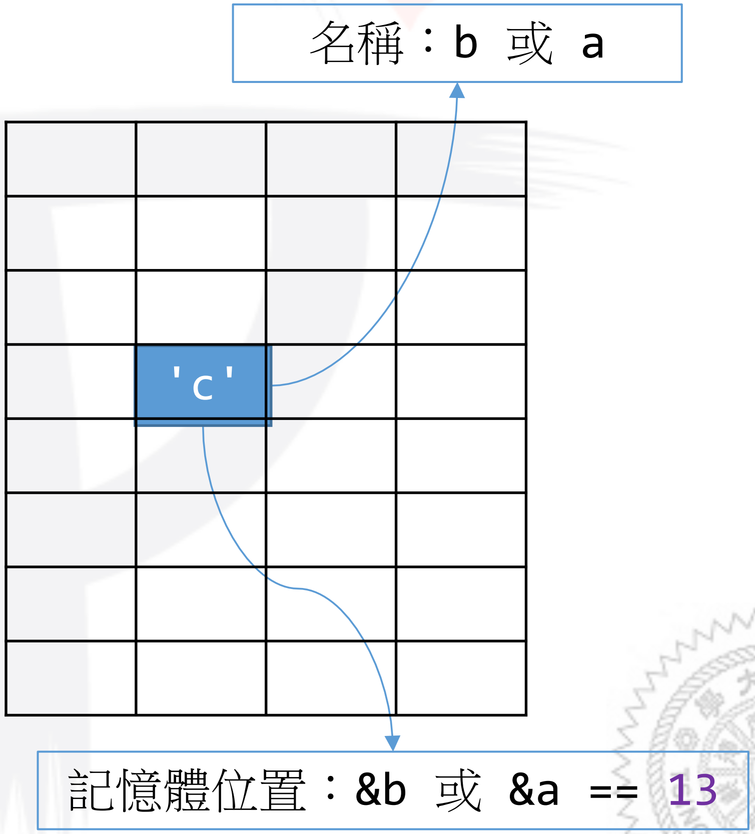
```
char b = 'c';
char a = b;
```



圖示 (傳參考)

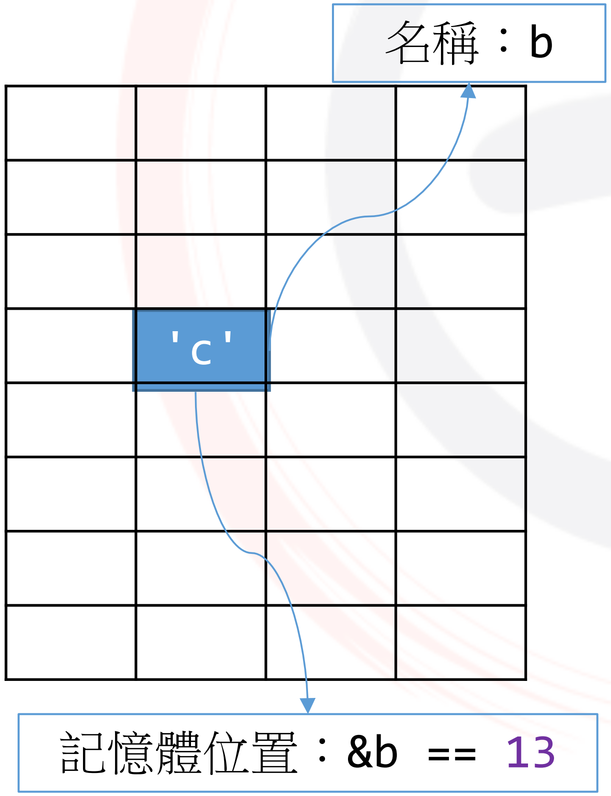
```
char b = 'c';
char &a = b;
```

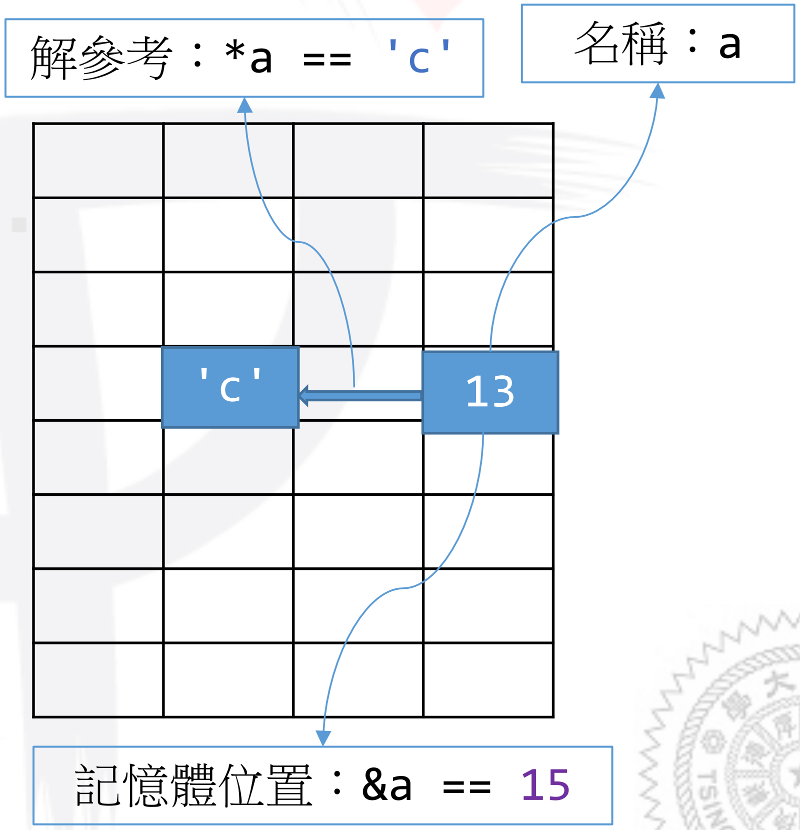




圖示 (傳址、傳指標)

```
char b = 'c';
char *a = &b;
```





Passing by value vs. Passing by address (1/2)

- Scalars (`int`, `char`, `double`, etc) are passed by value

```
void uselessSwap(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}
int main() {
    int x = 3, y = 10;
    uselessSwap(x, y);
    cout << "x = " << x << "y = " << y << '\n';
}
```

- a 和 b 拿到 x 和 y 的 copy
- 你是換 a 跟 b，但不是 x 跟 y
- 解決方式：傳指標或傳參考

Passing by value vs. Passing by address (2/2)

- Arrays are passed by ~~reference~~ address

```
w[] is "world", x[] is "hello"
```

```
void swapStrings(char a[ ], char b[ ]) {
    char temp[MAX];
    int i;
    /* copy a[] to temp[] */
    i = 0; do {temp[i] = a[i];}  while(a[i++]);
    /* copy b[ ] to a[ ]; */
    i = 0; do {a[i] = b[i];}     while(b[i++]);
    /* temp[ ] to b[ ] */
    i = 0; do {b[i]= temp[i];} while(temp[i++]);
}
int main() {
    char w[MAX] = "hello";
    char x[MAX] = "world";
    swapStrings(w, x);
    cout << "w[] is \"" << w << "\", x[] is \"" << x << "\"\n";
}
```

其實它是被看成 char *b
(傳指標)
因為可以做 b = b + 1;
只是我們看成傳參考也通

http://stackoverflow.com/questions/2559896/how-are-arrays-passed

Example: Array Operations

- 讀輸入到 array 可以做很多事！
  - maximum
  - minimum
  - median
  - mean
  - standard deviation..
- Strategy
  - 讀輸入(鍵盤或檔案)到 array
  - 寫函式對 array 做操作
  - 輸出結果

Example: Array Operations

```
#include <iostream>
using namespace std;
#define SIZE 100
/* prototypes */
int max(int[ ], int), min(int[ ], int);
double mean(int d[ ], int);
int main() {
    int A[SIZE];
    int score, size = SIZE;
    for (int i = 0; i < SIZE; i++) {
        if (!(cin >> score)) {
            size = i;
            break; /* end of input */
        }
        A[i] = score;
    }
    cout << "max=" << A[max(A,size)]
         << ", min=" << A[min(A,size)]
         << ", mean=" << mean(A,size) << '\n';
}
```

is `true` as you type `Ctrl-Z` or `Ctrl-D`

max and min — return the index

```
int max(int A[ ], int size) {
    int maxIndex = -1;
    for (int i = 0; i < size; i++)
        if (maxIndex < 0 || A[i] > A[maxIndex])
            maxIndex = i; /* save the index with the largest so far */
    return maxIndex;
}

int min(int A[ ], int size) {
    int minIndex = -1;
    for (int i = 0; i < size; i++)
        if (minIndex < 0 || A[i] < A[minIndex])
            minIndex = i; /* save the index with the smallest so far */
    return minIndex;
}
```

Example: mean (average)

```
double mean(int A[ ], int size) {
   int sum;
   for (i = 0; i < size; i++)
      sum += A[i];
   return (double) (sum / size);
}
```

- Please find 3 bugs in the above code!
    1. `int sum = 0;`
    2. `int i = 0;`
    3. `return (double) sum / size;`
- What about median (中位數)? or n^{th} rank (第 n 大的數)?
    - Possible answer: sorting (排序)

find max value vs. find max index

```
int findMaxIndex(int A[ ], int size) {
    int maxIndex = 0;
    for (int i = 1; i < size; i++)
        if (A[i] > A[maxIndex])
            maxIndex = i; /* save the index with the largest so far */
    return size > 0 ? maxIndex : -1;
}

int findMaxValue(int A[ ], int size) {
    /* suppose size > 0 */
    int maxValue = A[0];
    for (int i = 1; i < size; i++)
        if (A[i] > maxValue)
             maxValue = A[i]; /* save the largest value so far */
    return maxValue;
}
```

Example: Sort

- 常見的排序演算法 (sorting algorithms) (在資料結構、演算法會教)
- stable (3, 1, 3, 4, 2 → 1, 2, 3, 3, 4)
  - 氣泡排序 (bubble sort) — $O(n^2)$
  - 插入排序 (insertion sort) —$O(n^2)$
  - 桶子排序 (bucket sort) —$O(n)$;需要$O(k)$額外空間
  - 計數排序 (counting sort) —$O(n+k)$;需要 $O(n+k)$ 額外空間
  - 合併排序 (merge sort) —$O(n \log n)$;需要 $O(n)$ 額外空間
  - 基數排序 (radix sort) —$O(n \cdot k)$;需要 $O(n)$ 額外空間
- unstable (may 3, 1, 3, 4, 2 → 1, 2, 3, 3, 4)
  - 選擇排序 (selection sort) —$O(n^2)$
  - 希爾排序 (shell sort) —$O(n \log^2 n)$
  - 堆積排序 (heap sort) —$O(n \log n)$
  - 快速排序 (quick sort) —$O(n \log n)$期望時間, $O(n^2)$最壞情況;

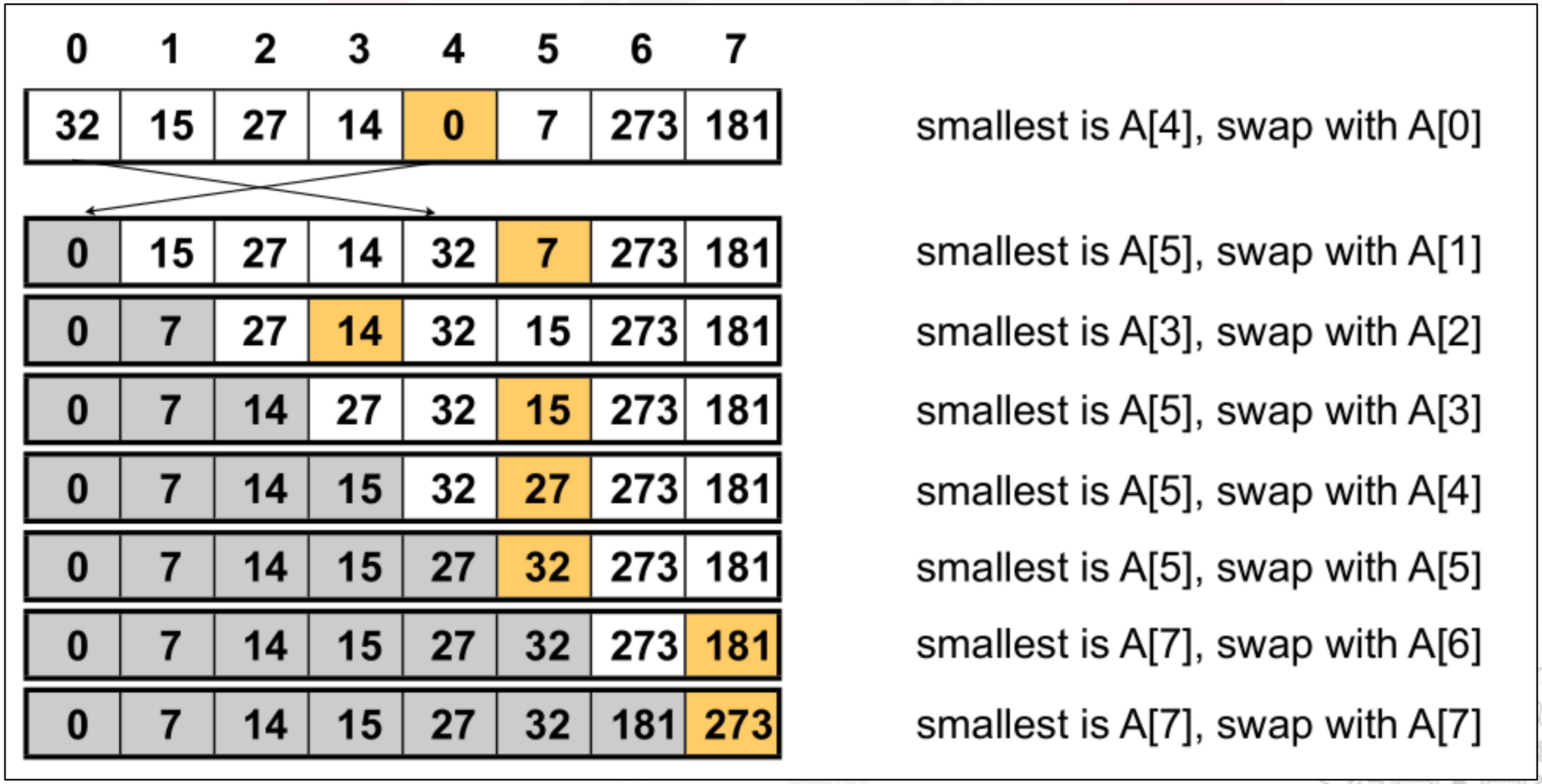
Example: Sort

- Wiki: https://zh.wikipedia.org/wiki/%E6%8E%92%E5%BA%8F%E7%AE%97%E6%B3%95
- 各種排序動畫： http://visualgo.net/sorting.html

Example: Selection Sort

- Algorithm
  1. 從 index (start ~ end) = 0 ~ n–1 中找到最小值的 index
  2. swap A[0] and A[index]
  3. 從 index (start ~ end) = 1 ~ n–1 中找到最小值的 index
  4. swap A[1] and A[index]
  5. …
  6. 直到 start == end

Selection Sort 圖示



0	1	2	3	4	5	6	7	
32	15	27	14	0	7	273	181	smallest is A[4], swap with A[0]
0	15	27	14	32	7	273	181	smallest is A[5], swap with A[1]
0	7	27	14	32	15	273	181	smallest is A[3], swap with A[2]
0	7	14	27	32	15	273	181	smallest is A[5], swap with A[3]
0	7	14	15	32	27	273	181	smallest is A[5], swap with A[4]
0	7	14	15	27	32	273	181	smallest is A[5], swap with A[5]
0	7	14	15	27	32	273	181	smallest is A[7], swap with A[6]
0	7	14	15	27	32	181	273	smallest is A[7], swap with A[7]

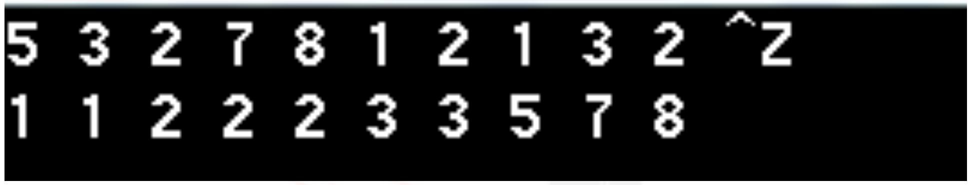
Example: code of Selection Sort

```
void swap(int A[], int x, int y){
    int temp = A[x];
    A[x] = A[y];
    A[y] = temp;
}

int findMinIndex(int A[], int start, int size) {
    int minIndex = -1;
    for (int i = start; i < size; i++)
        if (minIndex < 0 || A[i] < A[minIndex])
            minIndex = i;
    return minIndex;
}

void selectionSort(int A[], int size) {
    for (int i = 0; i < size; i++) {
        int m = findMinIndex(A, i, size);
        swap(A, i, m);
    }
}
```

Example: code of Selection Sort (2/2)

```
int main() {
    int A[100], size = 0, i = 0;
    while (size < 100 && cin >> A[size++]);
    selectionSort(A, --size); // Why do we do -
-size?
    while (i < size)
        cout << A[i++] << ' ';
}
```

Ctrl-Z or Ctrl-D to make this become `false`



```
5 3 2 7 8 1 2 1 3 2 ^Z
1 1 2 2 2 3 3 5 7 8
```

評估 Selection Sort 時間複雜度

1. 第 1 圈從足標 = 0~n–1 找最小值，比較 n 次
2. 第 2 圈從足標 = 1~n–1 找最小值，比較 n–1 次
3. 第 3 圈從足標 = 2~n–1 找最小值，比較 n–2 次
4. …
5. 第 n–1 圈從足標 = n–1~n–1 找最小值，比較 1 次

- 時間複雜度： $\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \approx O(n^2)$

Example: Insertion Sort

- Idea
  - Partition: index [0,j–1] sorted, index [j,n–1] unsorted
  - 每次挑 A[j] 與 A[0~j–1] 排序(從 j–1 比到 0)，j++
  - 直到 j == n
- Example
  - Input: [5 2 4 6 1 3]
  - j = 1: [5][2 4 6 1 3]
  - j = 2: [5 2][4 6 1 3], shift element 2 up=> [2 5][4 6 1 3]
  - j = 3: [2 5 4][6 1 3], shift element 4 up=> [2 4 5][6 1 3]
  - j = 4: [2 4 5 6][1 3], no shift => [2 4 5 6][1 3]
  - j = 5: [2 4 5 6 1][3], shift element 1 up=> [1 2 4 5 6][3]
  - j = 6: [1 2 4 5 6 3], shift element 3 up => [1 2 3 4 5 6]

Code for Insertion Sort

```
void insertionSort(int list[]) {
    for (int j = 1; j < MAX_NUMS; j++) { // j: unsorted index
        int unsortedItem = list[j];
        for (int sorted = j - 1;
            (sorted >= 0) && (list[sorted] > unsortedItem);
            sorted--)
            list[sorted+1] = list[sorted]; /* 往後搬 */
        list[sorted+1] = unsortedItem;
    }
}
```

評估 Insertion Sort 時間複雜度

1. 第 1 圈 list[1] 跟 list[0] 比，比較 1 次
2. 第 2 圈 list[2] 跟 list[1] 和 list[0] 比，比較 2 次 (最多)
3. 第 3 圈 list[3] 跟 list[2] ~ list[0] 比，比較 3 次 (最多)
4. …
5. 第 n–1 圈 list[n–1] 跟 list[n–2] ~ list[0] 比，比較 n–1 次（最多)

- 時間複雜度： $\sum_{i=1}^{n-1} i = \frac{(n-1)n}{2} \approx O(n^2)$

C String Manipulation

`#include <cstring>`

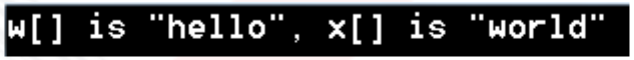
- 回傳字串長度
  - `size_t strlen(const char* str);`
- 複製字串 src 給字串 dest
  - `char* strcpy(char* dest, const char* src);`
  - `char* strncpy(char* dest, const char* src, size_t num);`
- 將字串 src 串接到字串 dest 後面
  - `char* strcat(char* dest, const char* src);`
  - `char* strncat(char* dest, const char* src, size_t num);`
- 比較兩字串 str1, str2是否相等
  - `int strcmp(const char* str1, const char* str2);`
  - `int strncmp(const char* str1, const char* str2, size_t num);`
  - 回傳 0 代表相等； < 0 代表 str1 的字典排序比 str2 前面(小)； > 0 反之。
- http://www.cplusplus.com/reference/cstring/

C++ String

- `#include <string>`
- 與 C 不同：(1) 不是用陣列，沒有邊界問題 (2) 傳值而非傳參考
- 宣告：
  - C string:
    - `char str1[100];`
  - C++ string:
    - `string str2;`
- 宣告時初始化：
  - C string:
    - `char str1[100] = "hello";`
  - C++ string:
    - `string str2 = "hello";`
    - `string str2("hello");`
    - `string str2 = str1;`
    - `string str2(str1);`
    - `string str3(5, 'h');`
- Assignment:
  - C string: 不允許！ // use `strcpy(str2, str1);`
  - C++ string: `str2 = str1; // C strings get promoted to C++ strings`

Passing by value vs. Passing by reference (1/2)

- Strings are passed by value

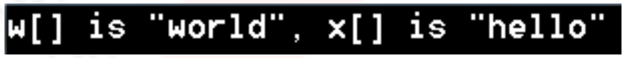


```
w[] is "hello", x[] is "world"
```

```cpp
void swapStrings(string a, string b) {
    string temp;
    temp = a;
    a = b;
    b = temp;
}
int main() {
    string w = "hello";
    string x = "world";
    swapStrings(w, x);
    cout << "w[] is \"" << w << "\", x[] is \""
        << x << "\"\n";
}
```

Passing by value vs. Passing by reference (2/2)

- How to fix it? add &s. (傳參考)



```
w[] is "world", x[] is "hello"
```

```cpp
void swapStrings(string& a, string& b) {
    string temp;
    temp = a;
    a = b;
    b = temp;
}
int main() {
    string w = "hello";
    string x = "world";
    swapStrings(w, x);
    cout << "w[] is \"" << w << "\", x[] is \"" <<
        x << "\"\n";
}
```

C++ String's Operators

運算子	功能	用法
=	指派	str1 = "abc"; strcpy(str1, "abc");
+	串接	str1 + str2
+=	串接並指派	str1 += str2; strcat(str1, str2);
==	等於	str1 == str2 !strcmp(str1, str2);
!=	不等於	str1 != str2 strcmp(str1, str2);
<	小於	str1 < str2, 依 ASCII 比較
<=	小於等於	str1 <= str2, 依 ASCII 比較
>	大於	str1 > str2, 依 ASCII 比較
>=	大於等於	str1 >= str2, 依 ASCII 比較
[]	佇標	str1[2]
<<	輸出	cout << str1;
>>	輸入	cin >> str1;

C++ String's Member Functions

成員函數	功能	用法
assign()	指派	str1.assign(str2);
append()	串接並指派	str1.append(str2);
compare()	比較兩個字串	str1.compare(str2);
insert()	插入字串	str1.insert(開始位置,str2);
replace()	取代字串	str1.replace(開始位置,長度,str2);

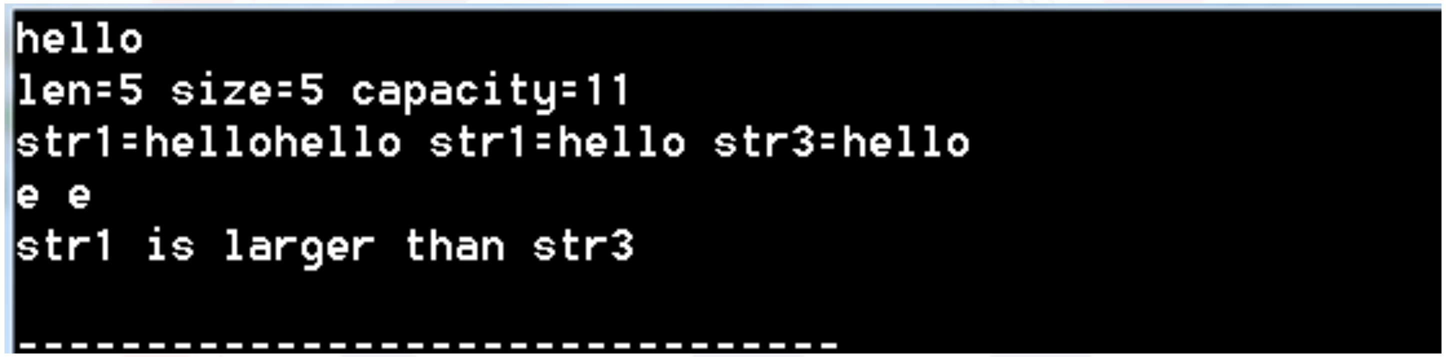
erase()	清除字串	str1.erase(開始位置,清除字元數);
		str1.erase(開始iterator,結尾iterator);

length()	取得字串長度	str1.length()
size()	取得字串長度	str1.size()
capacity()	取得字串容量	str1.capacity()
find()	找尋字串	str1.find(str2)
swap()	對調字串	str1.swap(str2);
empty()	是否為空字串	str1.empty() // true if empty
at()	取得第 n 個位置的字元	str1.at(3)
substr()	取得部分字串	str1.substr(開始位置,長度)

Example: C++ String

```
#include <iostream>
#include <string>
using namespace std;

int main(){
     string str1, str2, str3;
     char c[100] = "hello";
     string str("Test string");
     str = c;
     cin >> str1;
     cout << "len=" << str.length()
          << " size=" << str.size()
          << " capacity=" << str.capacity() << endl;
     str2 = str1;
     str3 = str1 + str2;
     str1.swap(str3);
     cout << "str1=" << str1
          << " str1=" << str2
          << " str3=" << str3 << endl;
     cout << str1.at(1) << ' ' << str1[1] << endl;
     if (str1 > str3)
          cout << "str1 is larger than str3\n";
}
```

Output



```
hello
len=5 size=5 capacity=11
str1=hellohello str1=hello str3=hello
e e
str1 is larger than str3

--------------------------------
```

題外話 (1/5)

為什麼我要一直強調在 C++ 裡，operator 也是一個 function?

- 除了可以自己定義 operator 之外還有什麼意義？

- 想想為什麼 expression 會有值？
- 因為 expression 的值就是 function 的回傳值啦！
- Recall: In Abstract Algebra, + is a function!
- 事實上 string 的 = 有點類似以下

$$+: \mathbb{Z}\times\mathbb{Z} \to \mathbb{Z}$$
$$(a,b) \vdash a+b$$
$$+(a,b) = a+b$$

```
string& operator=(const string& b){
   this->assign(b); // this is a pointer that points to this object
   return *this;
}
```

- 這樣讓我們可以做連續指派：

```
string a, b, c;
a = b = c;
// 先 b = c，回傳 b 的值，再 a = b，再回傳 a 的值(沒人接)
```

題外話 (2/5)

- 若定義 string 的 = 成以下：

```
void operator=(string &b){
  this->assign(b);
}
```

- 那就沒辦法做連續指派了喔！

```
string a, b, c;
a = b = c;
// 先 b = c，沒有回傳值，GG，a 不知道要 gets 什麼東西
```

- 只能

```
string a, b, c;
b = c;
a = b;
```

題外話 (3/5)

- 為什麼要 return reference 而不是直接 return string?

```
string operator=(const string& b){
   this->assign(b); // this is a pointer that points to this object
   return *this;
}
```

- return by reference 讓 function call 可以放在 = 左邊

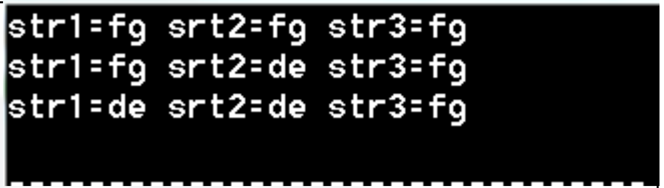
```
#include <iostream>
using namespace std;
int n;
int& test() {
    return n;
}
int main() {
    test() = 5;
    cout << n;
}
```

```
#include <iostream>
using namespace std;
int n;
int test() {
    return n;
}
int main() {
    test() = 5; // compile error!!
    cout << n;
}
```

題外話 (4/5)

- return by reference 讓 function call 可以放在 = 左邊，且可以繼續 get 右邊的值

```
#include <iostream>
#include <string>
using namespace std;
string myAssign(string &a, string &b){
     a.assign(b);
     return a;
}
int main() {
     string str1 = "abc", str2 = "de", str3 = "fg";
     str1 = str2 = str3;
     cout << "str1=" << str << " srt2=" << str2 << " str3=" << str3 << endl;
     str1 = "abc", str2 = "de", str3 = "fg";
     (str1 = str2) = str3;
     cout << "str1=" << str << " srt2=" << str2 << " str3=" << str3 << endl;
     str1 = "abc", str2 = "de", str3 = "fg";
     myAssign(str1,str2) = str3;
     cout << "str1=" << str << " srt2=" << str2 << " str3=" << str3 << endl;
}
```



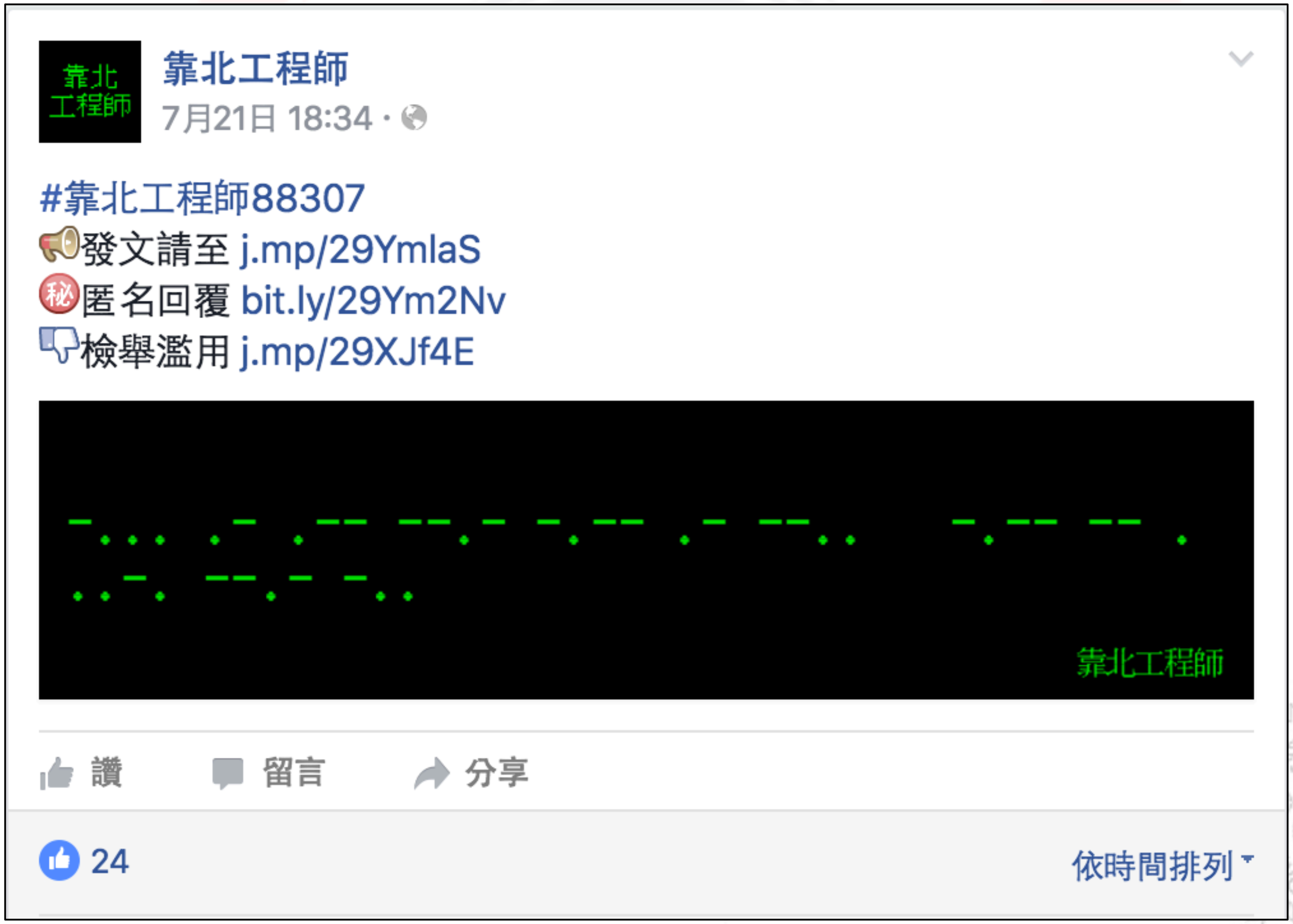
```
str1=fg srt2=fg str3=fg
str1=fg srt2=de str3=fg
str1=de srt2=de str3=fg
```

題外話 (5/5)

- 參考資料
- http://www.programiz.com/cpp-programming/return-reference
- http://www.tutorialspoint.com/cplusplus/returning_values_by_reference.htm

Some Practices

Mini Project II
摩斯密碼轉換器



靠北工程師

靠北工程師
7月21日 18:34 ·

#靠北工程師88307
發文請至 j.mp/29YmlaS
匿名回覆 bit.ly/29Ym2Nv
檢舉濫用 j.mp/29XJf4E

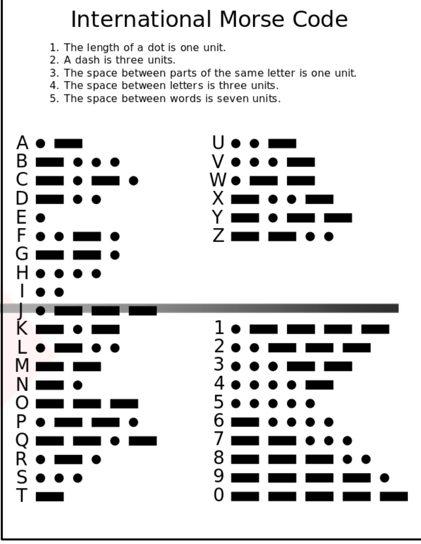
```
-... .- .-- --.- -.-- .- --..   -.-- -- .
..-. --.- -..
```

靠北工程師

讚 留言 分享

24 依時間排列

Mini Project II
摩斯密碼轉換器



International Morse Code

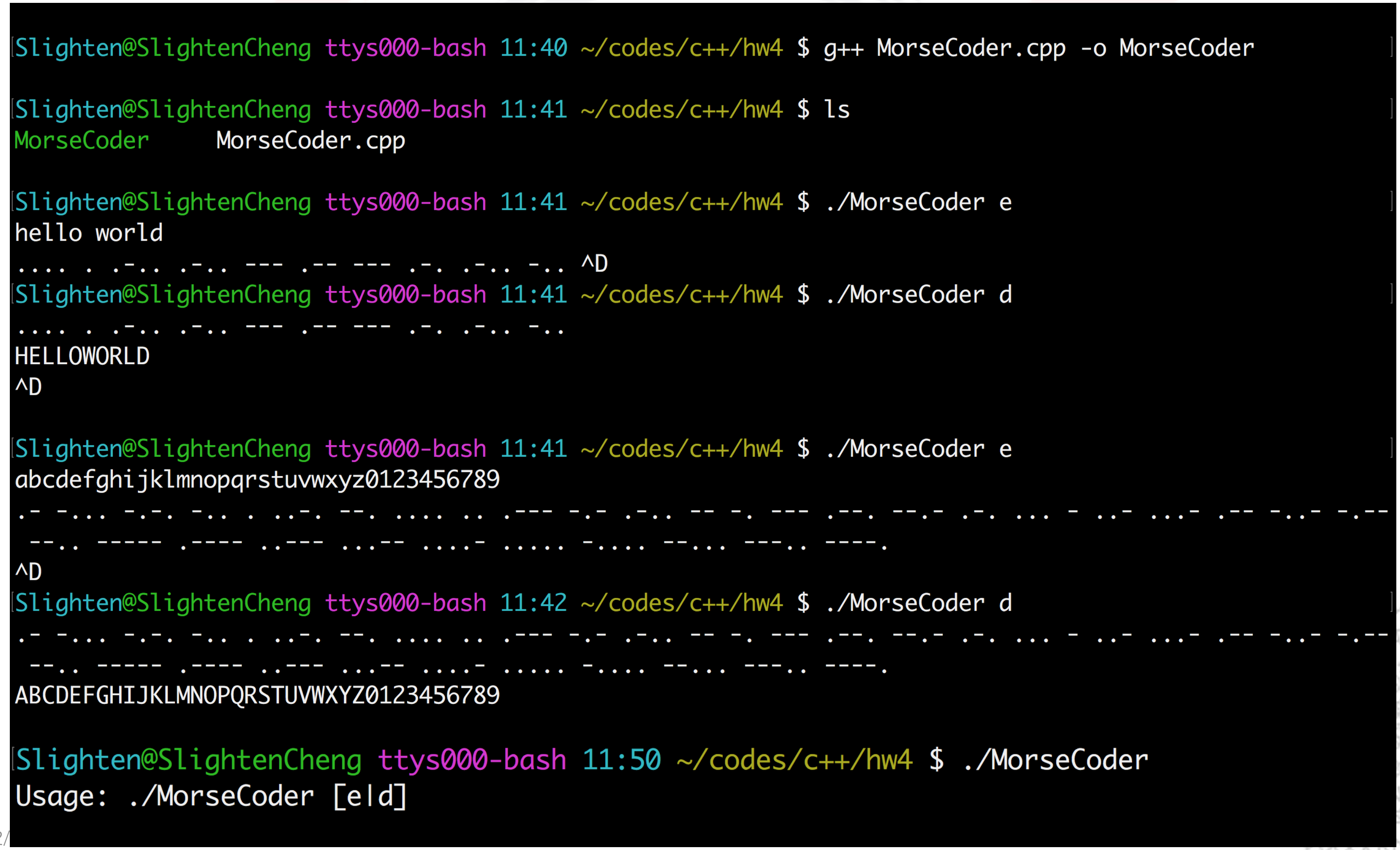
1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

A	● ▬	U	● ● ▬
B	▬ ● ● ●	V	● ● ● ▬
C	▬ ● ▬ ●	W	● ▬ ▬
D	▬ ● ●	X	▬ ● ● ▬
E	●	Y	▬ ● ▬ ▬
F	● ● ▬ ●	Z	▬ ▬ ● ●
G	▬ ▬ ●		
H	● ● ● ●		
I	● ●		
J	● ▬ ▬ ▬		
K	▬ ● ▬	1	● ▬ ▬ ▬ ▬
L	● ▬ ● ●	2	● ● ▬ ▬ ▬
M	▬ ▬	3	● ● ● ▬ ▬
N	▬ ●	4	● ● ● ● ▬
O	▬ ▬ ▬	5	● ● ● ● ●
P	● ▬ ▬ ●	6	▬ ● ● ● ●
Q	▬ ▬ ● ▬	7	▬ ▬ ● ● ●
R	● ▬ ●	8	▬ ▬ ▬ ● ●
S	● ● ●	9	▬ ▬ ▬ ▬ ●
T	▬	0	▬ ▬ ▬ ▬ ▬

- 檔名 MorseCoder.cpp
- `% g++ MorseCoder.cpp -o MorseCoder`
- 加密(Eng2Morse) `% ./MorseCoder e`
- 解密(Morse2Eng) `% ./MorseCoder d`
- 你必須處理右上角上那張表的A~Z與0~9
  (https://zh.wikipedia.org/wiki/%E6%91%A9%E5%B0%94%E6%96%AF%E7%94%B5%E7%A0%81?oldformat=true)
- 解密只需顯示大寫
- 加密大小寫都要能加密
- 請上 Morse Translator 網站產生測資
  (http://morsecode.scphillips.com/translator.html)

Sample Results

- ^D is Ctrl–D



```
Slighten@SlightenCheng ttys000-bash 11:40 ~/codes/c++/hw4 $ g++ MorseCoder.cpp -o MorseCoder

Slighten@SlightenCheng ttys000-bash 11:41 ~/codes/c++/hw4 $ ls
MorseCoder      MorseCoder.cpp

Slighten@SlightenCheng ttys000-bash 11:41 ~/codes/c++/hw4 $ ./MorseCoder e
hello world
.... . .-.. .-.. --- .-- --- .-. .-.. -.. ^D
Slighten@SlightenCheng ttys000-bash 11:41 ~/codes/c++/hw4 $ ./MorseCoder d
.... . .-.. .-.. --- .-- --- .-. .-.. -..
HELLOWORLD
^D

Slighten@SlightenCheng ttys000-bash 11:41 ~/codes/c++/hw4 $ ./MorseCoder e
abcdefghijklmnopqrstuvwxyz0123456789
.- -... -.-. -.. . ..-. --. .... .. .--- -.- .-.. -- -. --- .--. --.- .-. ... - ..- ...- .-- -..- -.--
 --.. ----- .---- ..--- ...-- ....- ..... -.... --... ---.. ----.
^D
Slighten@SlightenCheng ttys000-bash 11:42 ~/codes/c++/hw4 $ ./MorseCoder d
.- -... -.-. -.. . ..-. --. .... .. .--- -.- .-.. -- -. --- .--. --.- .-. ... - ..- ...- .-- -..- -.--
 --.. ----- .---- ..--- ...-- ....- ..... -.... --... ---.. ----.
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

Slighten@SlightenCheng ttys000-bash 11:50 ~/codes/c++/hw4 $ ./MorseCoder
Usage: ./MorseCoder [e|d]
```

Hints

1. You Need `argc, argv`
2. You may need `toupper(), isalpha(), isdigit()` in `<cctype>`
3. You may need `strcmp(), strlen()` in `<cstring>`
4. Recall in Ch2, how `'0', '1', '2'`… `->` `0, 1, 2`… `( - '0')`
5. Recall in Ch2, how `'A', 'B', 'C'`… `->` `0, 1, 2`… `( - 'A')`
6. Recall type–casting: `int i = 65; cout << (char) i << endl;`

Template (1/2)

```
#include <iostream>
#include <cstring> // strlen(), strcmp()
#include <cstdlib> // exit(), atoi()
#include <cctype> // toupper(), isalnum(), isalpha(),
isdigit()
using namespace std;

char map[36][6] = {
    ".-", "-...", "-.-.", "-..", ".",
    "..-.", "--.", "....", "..", ".---",
    "-.-", ".-..", "--", "-.", "---",
    ".--.", "--.-", ".-.", "...", "-",
    "..-", "...-", ".--", "-..-", "-.--", "--..",
    /* A~Z */
    "-----", ".----", "..---", "...--", "....-",
    ".....", "-....", "--...", "---..", "----.",
    /* 0~9 */
};
```

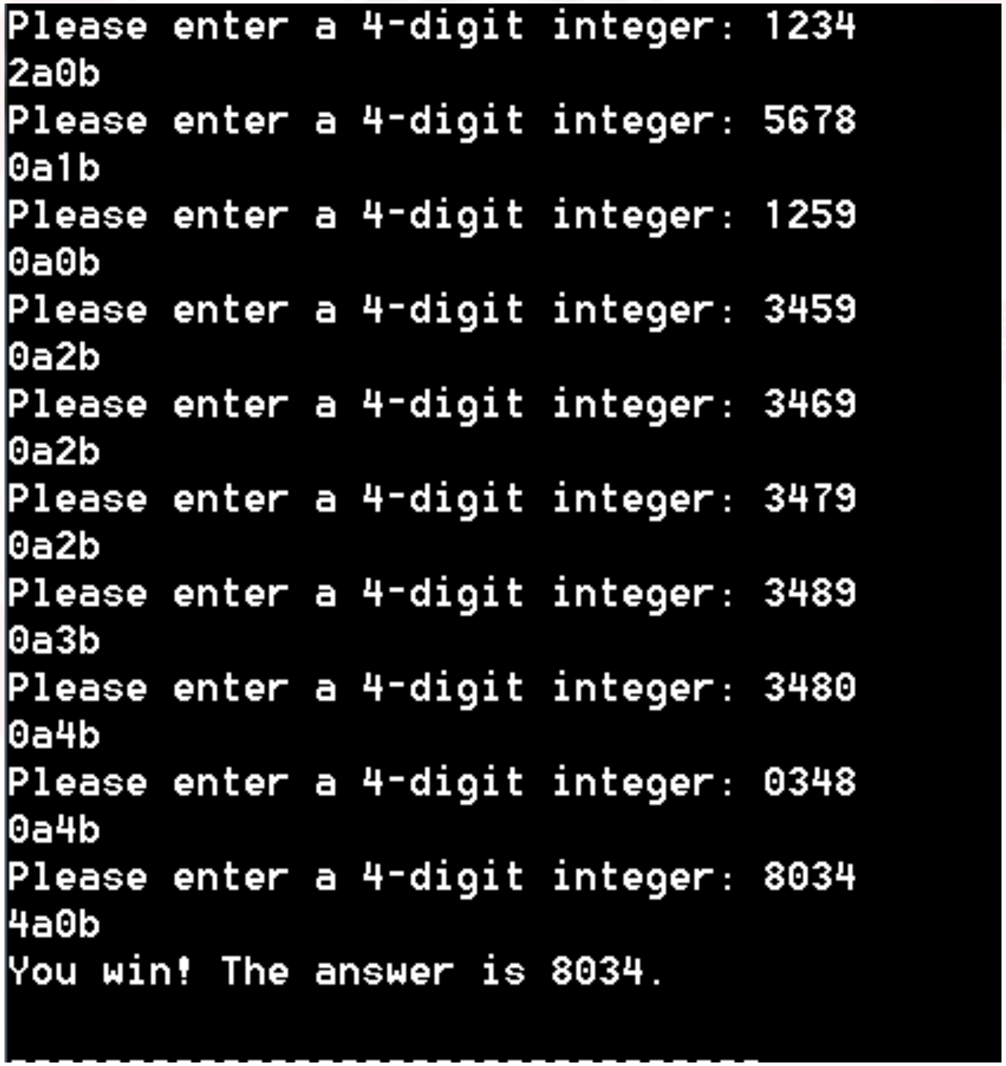
Template (2/2)

```
void decode(){ /* @@@TODO: Morse to English */ }
void encode(){ /* @@@TODO: English to Morse */ }

int main(int argc, char ** argv){
    if (argc != 2 ||
      (argv[1][0] != 'd' && argv[1][0] != 'e')) {
        cerr << "Usage: " << argv[0] << " [e|d]\n";
        return 1;
    }
    else if (argv[1][0] == 'd') decode();
    else encode();
    return 0;
}
```

Mini Project III: Bulls and Cows (幾a幾b、猜數字)

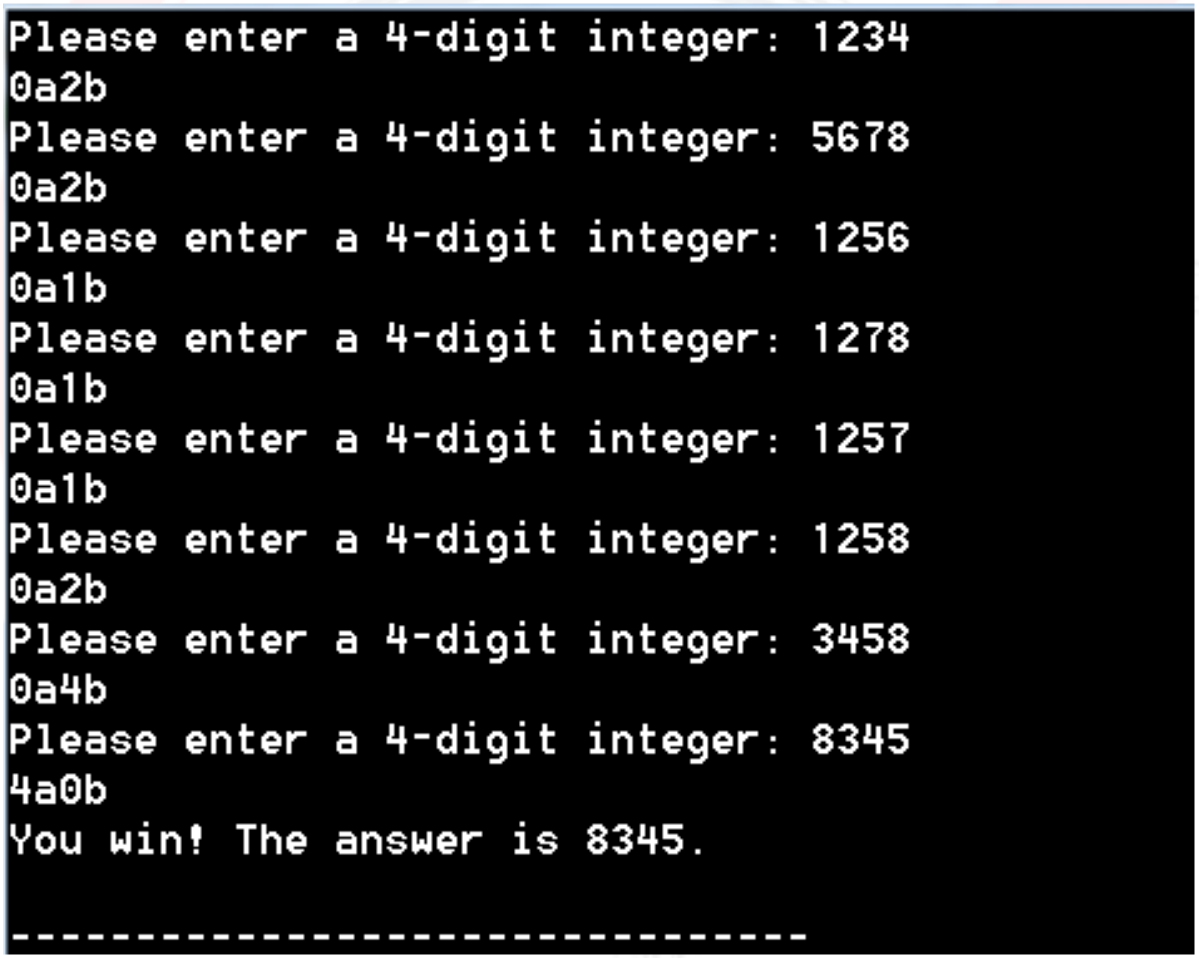
- 檔名叫做 nanb.cpp
- 遊戲規則就是一般的幾 a 幾 b 的規則
- 先產生一組 4 位數數字為答案(每個 digit 都不同)
- 假設答案是 5487
- 若使用者猜 1234，則顯示 0a1b
- 若使用者猜 5678，則顯示 1a2b
- 若使用者猜 5478，則顯示 2a2b
- 若使用者猜 5487，則顯示 4a0b
- 猜中，遊戲結束！

Sample Output I



```
Please enter a 4-digit integer: 1234
2a0b
Please enter a 4-digit integer: 5678
0a1b
Please enter a 4-digit integer: 1259
0a0b
Please enter a 4-digit integer: 3459
0a2b
Please enter a 4-digit integer: 3469
0a2b
Please enter a 4-digit integer: 3479
0a2b
Please enter a 4-digit integer: 3489
0a3b
Please enter a 4-digit integer: 3480
0a4b
Please enter a 4-digit integer: 0348
0a4b
Please enter a 4-digit integer: 8034
4a0b
You win! The answer is 8034.
```

Sample Output II



```
Please enter a 4-digit integer: 1234
0a2b
Please enter a 4-digit integer: 5678
0a2b
Please enter a 4-digit integer: 1256
0a1b
Please enter a 4-digit integer: 1278
0a1b
Please enter a 4-digit integer: 1257
0a1b
Please enter a 4-digit integer: 1258
0a2b
Please enter a 4-digit integer: 3458
0a4b
Please enter a 4-digit integer: 8345
4a0b
You win! The answer is 8345.

--------------------------------
```

Flow

1. 先 random 產生一組每位數都不同的答案
   ① 該用什麼存？
   ② 如何確保每位數都不同數字？
2. prompt "Please enter a 4-digit integer: "
3. cin
   - 該用什麼存 cin？
4. 判斷幾 a 幾 b
   - 如何判斷？
5. output "?a?b"
6. (a) 若正確 (4a0b)，則遊戲結束，並output "You win! The answer is " + answer + '.'
   (b) 若否，則回到第 2 步

Hint

- Use int arrays.
- Every digit is an element of the array.
- Use a nested loop to compare every digit of answer with new random digit.
- Use a nested loop to compare every digit of answer with guess.

Template (1/2)

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

/* random generate a 4 distinct-digit number */
void generateAnswer(int answer[]){
    /* ??? */
}

/* calculate how much a and b are
 * think why passing by reference instead of by value */
void calNaNb(int answer[], int guess[], int &a, int &b){
    /* ??? */
}
```

Template (2/2)

```
int main(int argc, char **argv){
    int answer[4], num, a = 0, b = 0;
    srand(time(NULL));
    generateAnswer(answer);
    do {
        a = b = 0; // Why?
        cout << "Please enter a 4-digit integer: ";
        cin >> num;
        int guess[4] = {?,?,?,?}; // recall 4-digit.cpp in Ch2
        calNaNb(answer, guess, a, b);
        cout << a << 'a' << b << 'b' << endl;
    } while (a != 4);
    cout << "You win! The answer is ";
    for (int i = 0; i < 4; i++) cout << answer[i];
    cout << '.' << endl;
    return 0;
}
```

想一想

- 如果遊戲要改成 n-digit 而不是 4-digit 那麼哪些程式碼要改？ 有 4 都要改

- A better way

```
#define LEN 4
```

```
int answer[LEN];
int guess[LEN];
for (int i = 0; i < LEN; i++)
    cout << answer[i];
```