04 陣列與向量（II）

Fundamental Computer Programming- C++ Lab(I)

元智大學 | C++ 程式設計實習

張維元

課程投影片： 請從元智個人 Portal 下載

C++ 程式設計實習

- 01 程式設計與開發環境
- 02 變數型態與運算
- 03 流程控制
- 04 陣列與向量
- 05 函數與遞迴函數
- 06 字串與指標
- 07 循序檔案和隨機檔案

2015

維元

(@v123582)

Web Development

Data Science

#遠端 #斜槓 #教學
#資料科學 #網站開發

擅長網站開發與資料科學的雙棲工程師，熟悉的語言是 Python 跟 JavaScript。同時也是 資料科學家的工作日常 粉專及 資料科學家的 12 堂心法課 發起人，擁有多次國內大型技術會議講者經驗，持續在不同的平台發表對 #資料科學、#網頁開發 或 #軟體職涯 相關的分享。

- 元智大學 C++/CPE 程式設計課程 講師
- ALPHACamp 全端 Web 開發 / Leetcode Camp 課程講師
- CUPOY Python 網路爬蟲實戰研習馬拉松 發起人
- 中華電信學院 資料驅動系列課程 講師
- 工研院 Python AI人工智慧資料分析師養成班 講師
- 華岡興業基金會 AI/Big Data 技能養成班系列課程 講師

site: v123582.tw / line: weiwei63
mail: weiyuan@saturn.yzu.edu.tw

2015

維元

(@v123582)

Web Development

Data Science

#遠端 #斜槓 #教學
#資料科學 #網站開發

擅長網站開發與資料科學的雙棲工程師，熟悉的語言是 Python 跟 JavaScript。同時也是 資料科學家的工作日常 粉專及 資料科學家的 12 堂心法課 發起人，擁有多次國內大型技術會議講者經驗，持續在不同的平台發表對 #資料科學、#網頁開發 或 #軟體職涯 相關的分享。

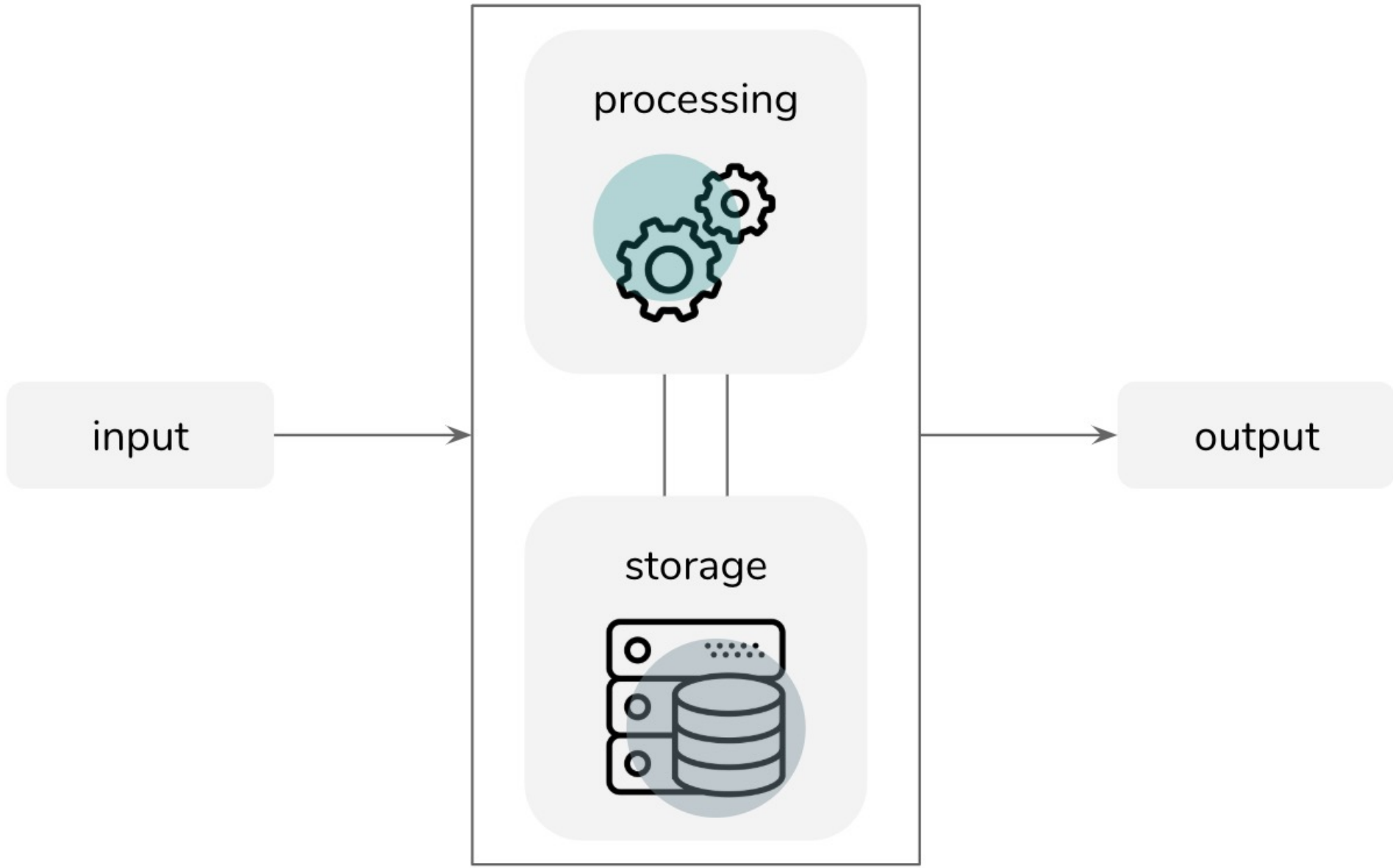
- 2018 總統盃黑客松 冠軍隊伍
- 2017 資料科學愛好者年會(DSCONF) 講師
- 2017 行動科技年會(MOPCON) 講師
- 2016 微軟 Imagine Cup 台灣區冠軍

site: v123582.tw / line: weiwei63
mail: weiyuan@saturn.yzu.edu.tw

什麼是程式？

程式語言是用來命令電腦執行各種作業的工具，是人與電腦溝通的橋樑。當電腦藉由輸入設備把程式讀入後，會儲存在主記憶體內，然後指令會依序被控制單元提取並解碼或翻譯成電腦可以執行的信號，並把信號送到各個裝置上，以執行指令所指派的動作。也就是說，人類與電腦溝通的語言稱為程式語言。

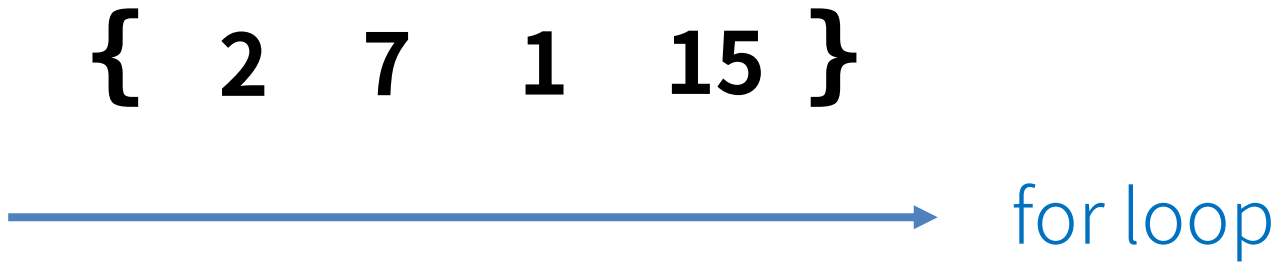
程式 = 利用一系列的指令告訴電腦如何執行工作


processing
input
output
storage

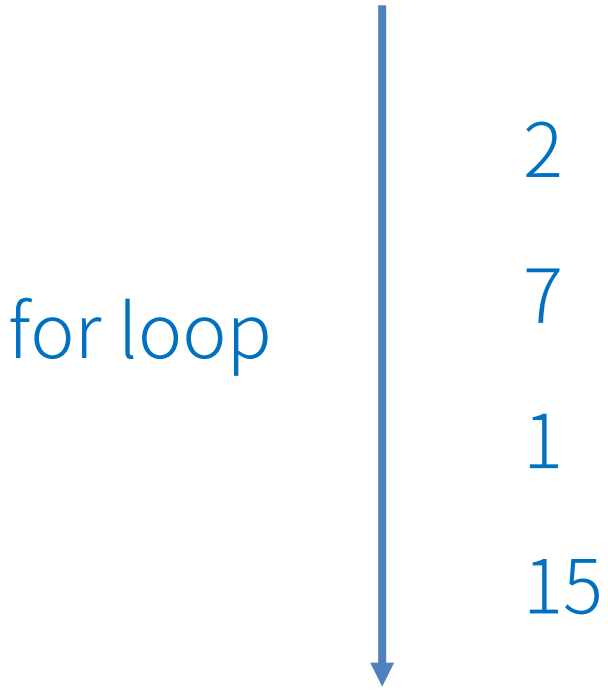
作業 #15

- #練習：Given an array of integers, return indices of the two numbers such that they add up to a specific target. You may assume that each input would have exactly one solution, and you may not use the same element twice.
- Requirements：
  1. 每次輸入一個數字 n 代表 target
  2. 找出 [2, 7, 11, 15] 陣列中兩數總和為 target 的 index 存入陣列
  3. 限制最多只能使用一個迴圈跟一個條件判斷
- Sample Input： nums = [2, 7, 11, 15], target = 9,
- Sample Output： [0, 1]

{ 2 7 1 15 }

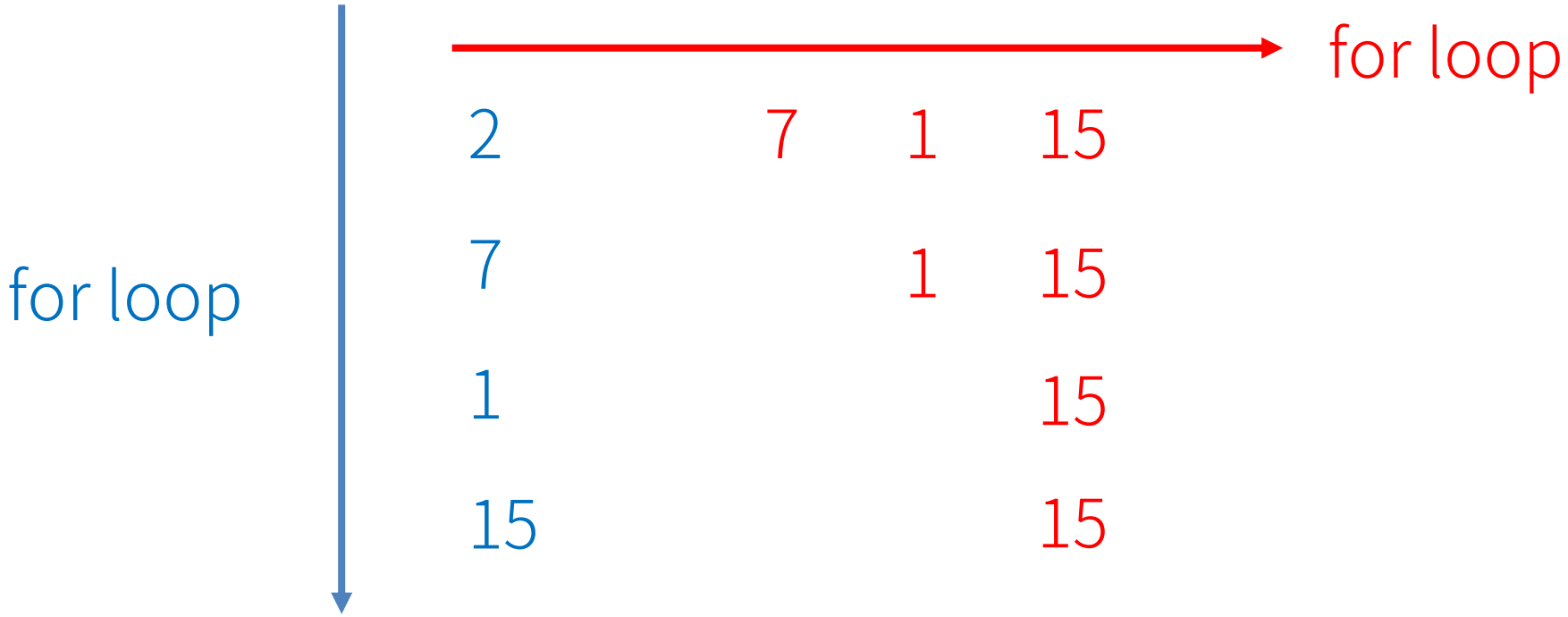

{ 2 7 1 15 }
for loop

{ 2 7 1 15 }

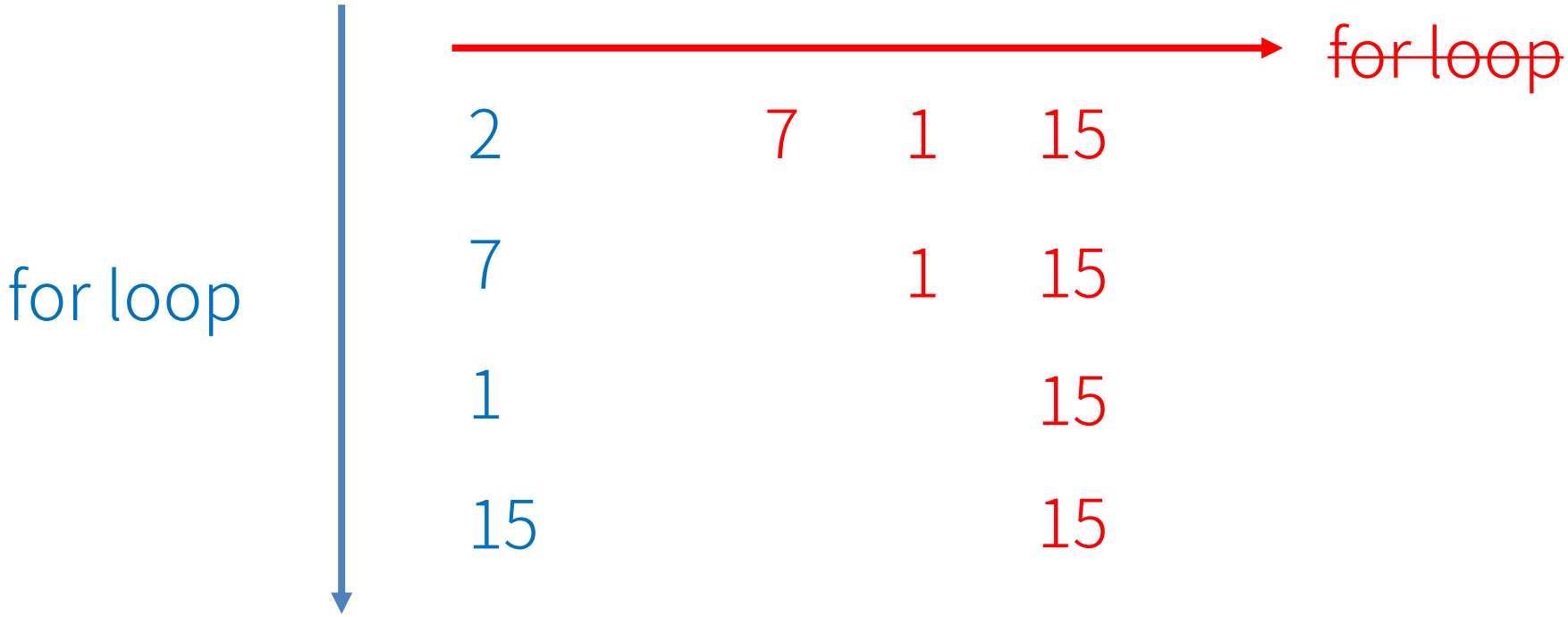


{ 2 7 1 15 }

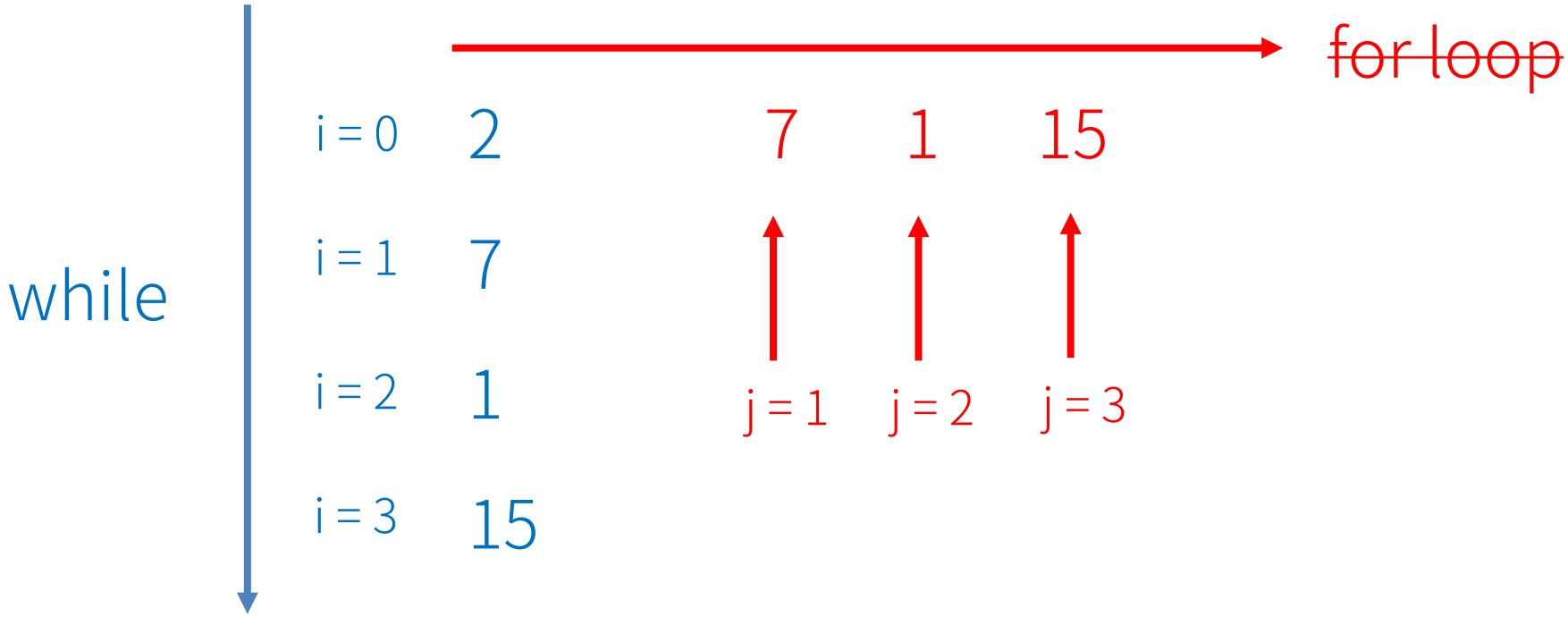
{ 2 7 1 15 }



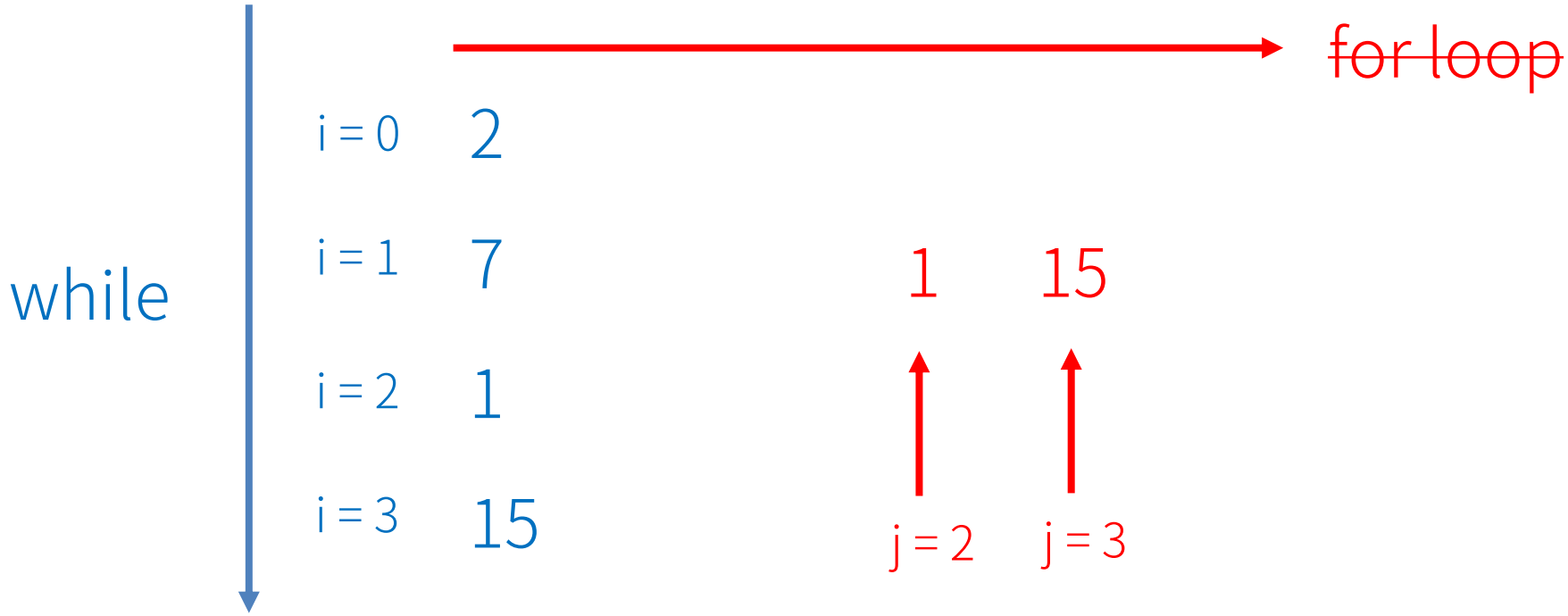
{ 2 7 1 15 }



{ 2 7 1 15 }



{ 2 7 1 15 }



{ 2 7 1 15 } t = 22

{ 2 7 1 15 } t = 22

20 15 21 7

{ 2 7 1 15 } t = 22

for loop

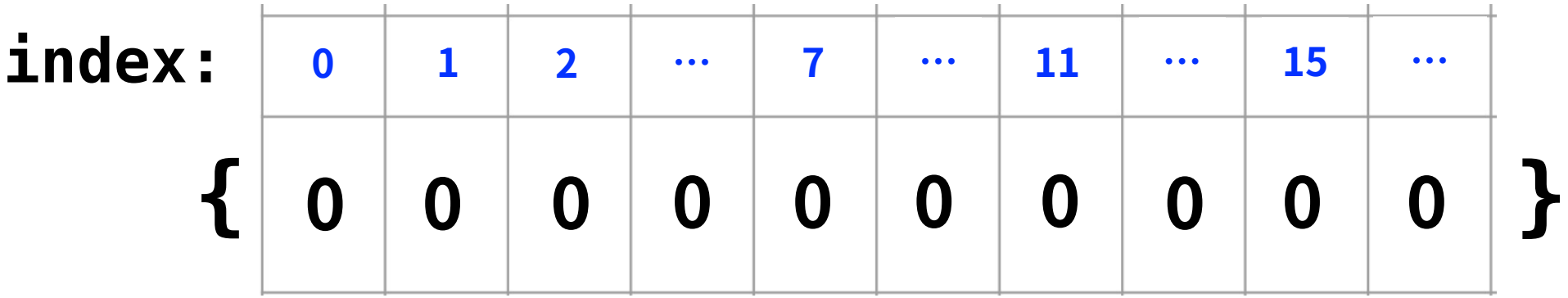
2 t – 2 = 20 ?

7 t – 2 = 18 ?

1 t – 1 = 21 ?

15 t – 15 = 7 ?

陣列索引一定是連續的數字



→ 把 Index 當成出現數字，Value 當成出現與否

{ 2 7 1 15 } t = 22

for loop

		0 1 2
2	t – 2 = 20 ?	{ 0 0 1 … }
7	t – 2 = 18 ?	
1	t – 1 = 21 ?	
15	t – 15 = 7 ?	

作業 #14

- #練習： 在 n*n 的方陣中依照回形（蛇形）的方式，由右上開始依序填入 1, 2, … n*n。
- Requirements：
  1. 輸入一個數字 N，定義一個 N*N 的二維陣列
  2. 輸出一個由 1, 2, .. N*N 的回形（蛇形）方陣
- Sample Input：參考下頁
- Sample Output：參考下頁
- Note：僅限 11/10、12/01 上課繳交

main.cpp

```cpp
#include<iostream>
#define N 10
using namespace std;

int a[N][N] = {};
int main(){

  int n, x, y, m = 1;
  cin >> n;
  a[x = 0][y = n-1] = m;

  // Your Code
  for(int i = 0; i < n; i++){
    for(int j = 0; j < n; j++){
      cout << a[i][j] << "\t";
    }
    cout << endl;
  }
  return 0;
}
```

https://ProductiveFussySystemadministrator.v123582.repl.run

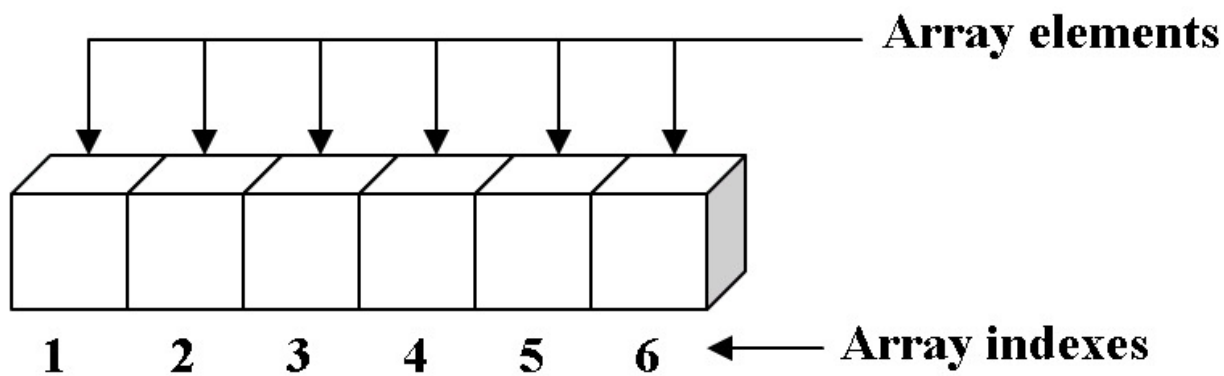
```
> clang++-7 -pthread -std=c++17 -o main main.cpp
> ./main
4
10  11  12  1
9   16  13  2
8   15  14  3
7   6   5   4
>
```

輸入 n ，印出 1 到 n*n 之間的數

	j = 0	j = 1	j = 2	j =3
i = 0	A[0][0]	A[0][1]	A[0][2]	A[0][3]
i = 1	A[1][0]	A[1][1]	A[1][2]	A[1][3]
i = 2	A[2][0]	A[2][1]	A[2][2]	A[2][3]

陣列（Array）

由元素型態皆相同的有序串列所組成，佔用連續性的記憶體空間。

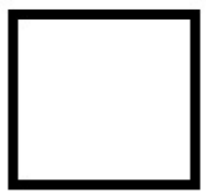


One-dimensional array with six elements

變數與陣列

→ 陣列的長度要預先定義

int a;



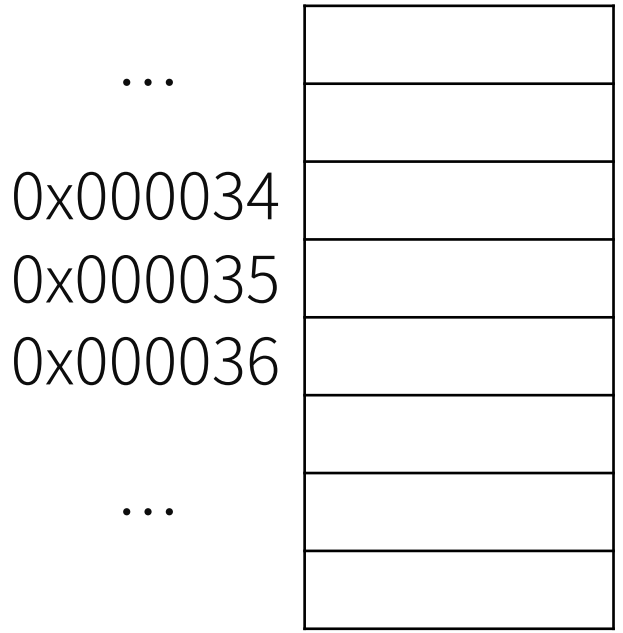
a

int a[5];

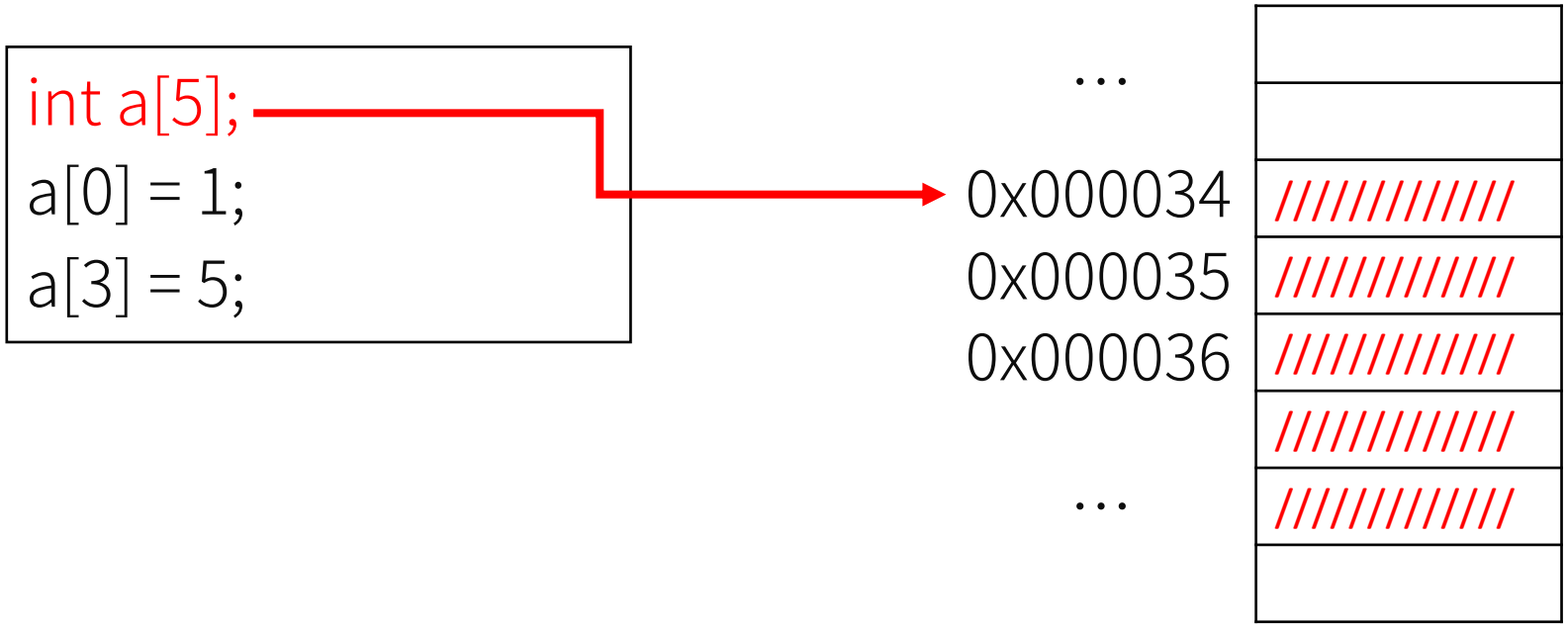
a[0] a[1] a[2] a[3] a[4]

陣列會佔用連續的記憶體空間

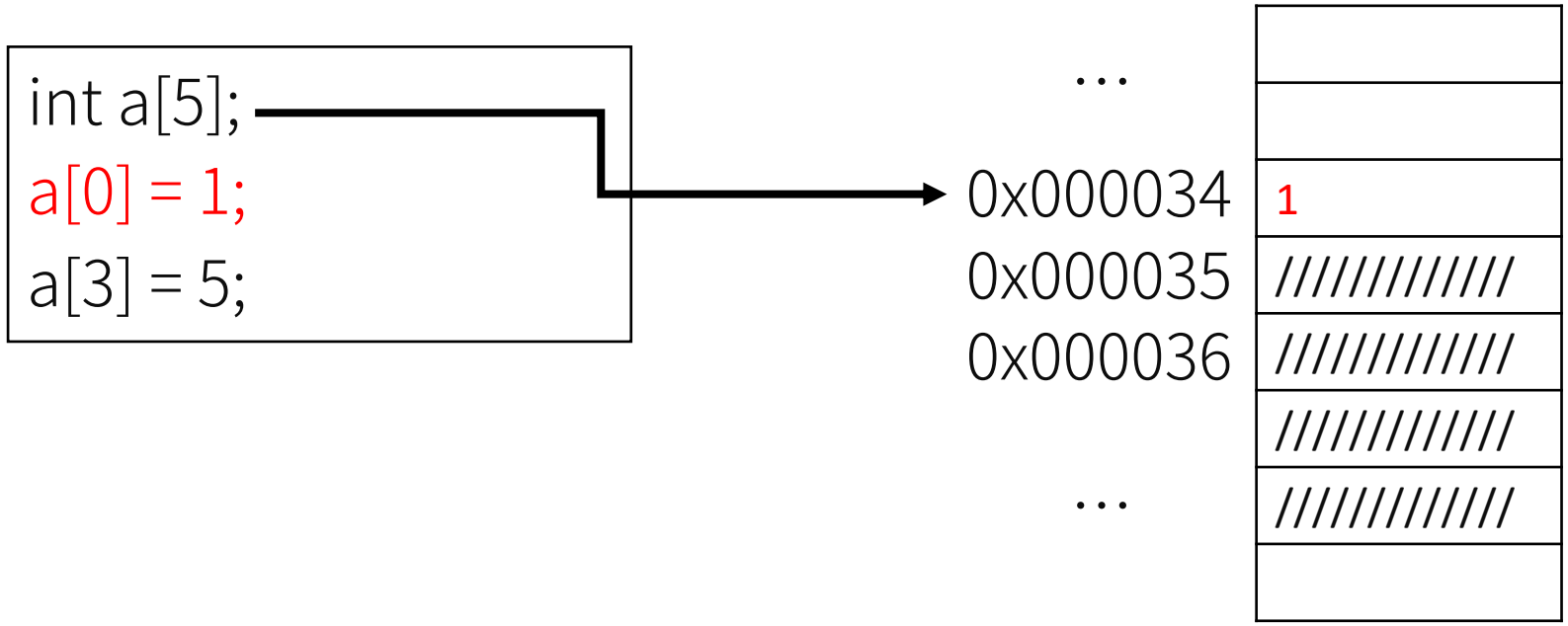
```
int a[5];
a[0] = 1;
a[3] = 5;
```



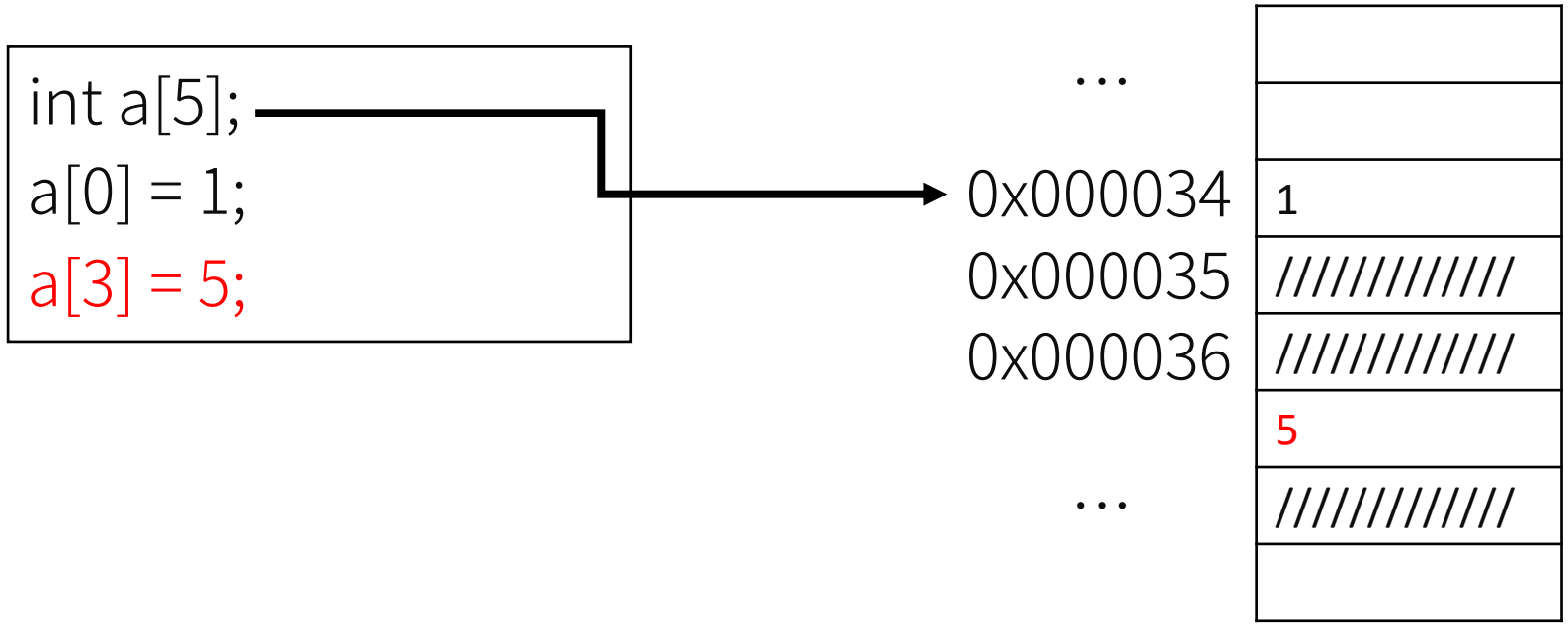
陣列會佔用連續的記憶體空間



陣列會佔用連續的記憶體空間



陣列會佔用連續的記憶體空間



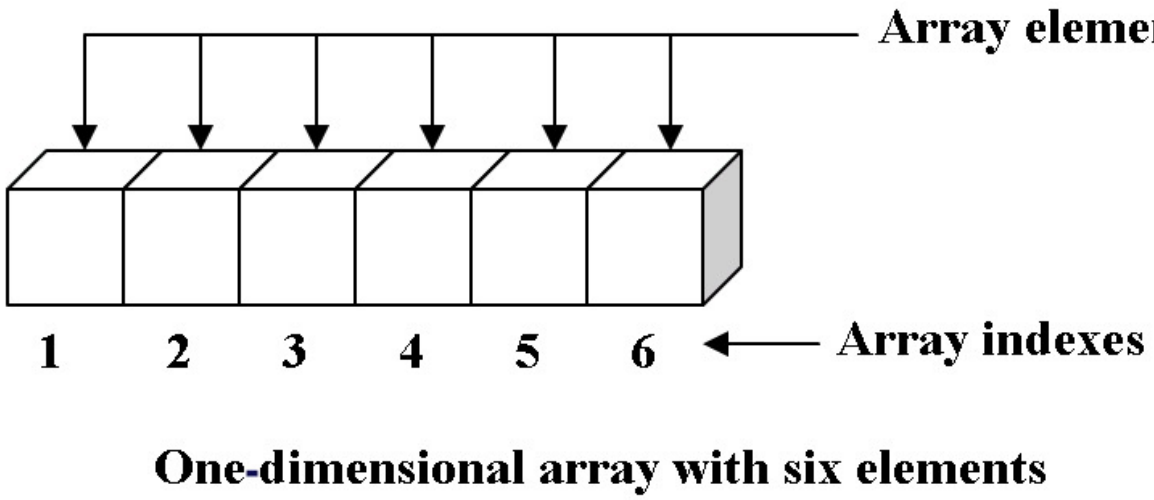
陣列是由連續的位置所組成的容器

	[	H	e	l	l	o		W	o	r	l	d]
index:		0	1	2	3	4	5	6	7	8	9	10

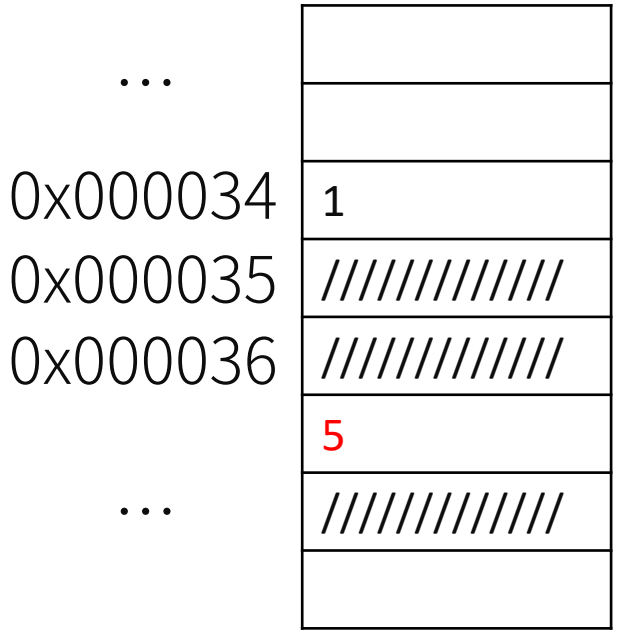
■以 { } 語法宣告陣列，以 [ ] 語法來取值與賦值

```
1
2  int s[3] = {1, 2, 3};
3  cout << s[0] << s[1] << s[2]; // 123
4
5  s[0] = 999;
6  s[2] = -s[2];
7  cout << s[0] << s[1] << s[2]; // 9992-3
8
```

陣列是由連續的位置所組成的容器



One-dimensional array with six elements



→ 使用的時候是一個容器，本質上是存在連續個記憶體位置上

陣列的宣告與使用

```
1  #include<iostream>
2  #define MAXN 10
3  using namespace std;
4
5  int a[MAXN];
6  int main(){
7    for(int i = 0; i < MAXN; i++){
8      a[i] = i * i;
9    }
10   int n = sizeof(a)/sizeof(a[0]);
11
12   cout << "a = " << a << endl;
13   cout << "a[0] = " << a[0] << endl;
14   cout << "a[MAXN-1] = " << a[MAXN-1] << endl;
15   cout << "a[n] = " << a[n-1] << endl;
16
17   return 0;
18 }
```

陣列的宣告與使用

```
1  #include<iostream>
2  #define MAXN 10 → 陣列的長度要預先定義
3  using namespace std;
4
5  int a[MAXN];
6  int main(){
7    for(int i = 0; i < MAXN; i++){
8      a[i] = i * i;
9    }
10   int n = sizeof(a)/sizeof(a[0]);
11
12   cout << "a = " << a << endl;
13   cout << "a[0] = " << a[0] << endl;
14   cout << "a[MAXN-1] = " << a[MAXN-1] << endl;
15   cout << "a[n] = " << a[n-1] << endl;
16
17   return 0;
18 }
```

陣列的宣告與使用

```
1  #include<iostream>
2  #define MAXN 10
3  using namespace std;
4
5  int a[MAXN]; → 建議宣告再 main 的外面
6  int main(){
7    for(int i = 0; i < MAXN; i++){
8      a[i] = i * i;
9    }
10   int n = sizeof(a)/sizeof(a[0]);
11
12   cout << "a = " << a << endl;
13   cout << "a[0] = " << a[0] << endl;
14   cout << "a[MAXN-1] = " << a[MAXN-1] << endl;
15   cout << "a[n] = " << a[n-1] << endl;
16
17   return 0;
18 }
```

陣列的宣告與使用

```
1  #include<iostream>
2  #define MAXN 10
3  using namespace std;
4
5  int a[MAXN];
6  int main(){
7    for(int i = 0; i < MAXN; i++){
8      a[i] = i * i; → 利用迴圈填入數值
9    }
10   int n = sizeof(a)/sizeof(a[0]);
11
12   cout << "a = " << a << endl;
13   cout << "a[0] = " << a[0] << endl;
14   cout << "a[MAXN-1] = " << a[MAXN-1] << endl;
15   cout << "a[n] = " << a[n-1] << endl;
16
17   return 0;
18 }
```

陣列的宣告與使用

```
1  #include<iostream>
2  #define MAXN 10
3  using namespace std;
4
5  int a[MAXN];
6  int main(){
7    for(int i = 0; i < MAXN; i++){
8      a[i] = i * i;
9    }
10   int n = sizeof(a)/sizeof(a[0]); → 陣列長度的計算
11
12   cout << "a = " << a << endl;
13   cout << "a[0] = " << a[0] << endl;
14   cout << "a[MAXN-1] = " << a[MAXN-1] << endl;
15   cout << "a[n] = " << a[n-1] << endl;
16
17   return 0;
18 }
```

陣列的宣告與使用

```
1  #include<iostream>
2  #define MAXN 10
3  using namespace std;
4
5  int a[MAXN];
6  int main(){
7    for(int i = 0; i < MAXN; i++){
8      a[i] = i * i;
9    }
10   int n = sizeof(a)/sizeof(a[0]);
11
12   cout << "a = " << a << endl; → 直接印出陣列會印出記憶體位置
13   cout << "a[0] = " << a[0] << endl;
14   cout << "a[MAXN-1] = " << a[MAXN-1] << endl;
15   cout << "a[n] = " << a[n-1] << endl;
16
17   return 0;
18 }
```

陣列的宣告與使用

```
1  #include<iostream>
2  #define MAXN 10
3  using namespace std;
4
5  int a[MAXN];
6  int main(){
7    for(int i = 0; i < MAXN; i++){
8      a[i] = i * i;
9    }
10   int n = sizeof(a)/sizeof(a[0]);
11
12   cout << "a = " << a << endl;
13   cout << "a[0] = " << a[0] << endl;
14   cout << "a[MAXN-1] = " << a[MAXN-1] << endl; → 利用索引取值
15   cout << "a[n] = " << a[n-1] << endl;
16
17   return 0;
18 }
```

一維陣列與二維陣列

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

i = 0	i = 1	i = 2	i = 3
A[0]	A[1]	A[2]	A[3]

```
int a[3][4] = {
  {10, 11, 12, 13},
  {14, 15, 16, 17},
  {18, 19, 20, 21},
};
```

	j = 0	j = 1	j = 2	j =3
i = 0	A[0][0]	A[0][1]	A[0][2]	A[0][3]
i = 1	A[1][0]	A[1][1]	A[1][2]	A[1][3]
i = 2	A[2][0]	A[2][1]	A[2][2]	A[2][3]

動態宣告陣列

當無法預知陣列的大小時，而是由使用者輸入後才做決定的話。這時候可以使用動態宣告陣列的技巧，根據程式運行的過程才向記憶體要空間。

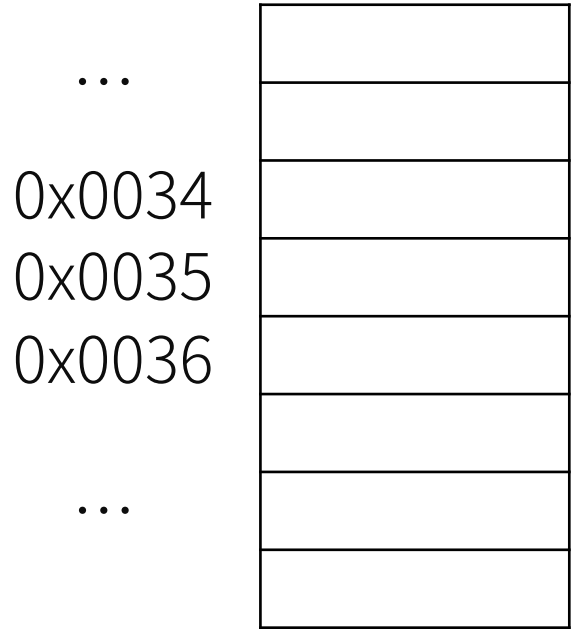
```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5    int n;
6    cin >> n;
7    int a[n];
8
9    for(int i = 0; i < n; i++){
10     cout << i << " => "<<a[i]
11   }
12 }
```

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5    int n;
6    cin >> n;
7    int *a = new int[n];
8    // int *a;
9    // a = new int[n];
10
11   for(int i = 0; i < n; i++){
12     cout << i << " => "<<a[i] << endl;
13   }
14 }
```

動態宣告陣列

當無法預知陣列的大小時，而是由使用者輸入後才做決定的話。這時候可以使用動態宣告陣列的技巧，根據程式運行的過程才向記憶體要空間。

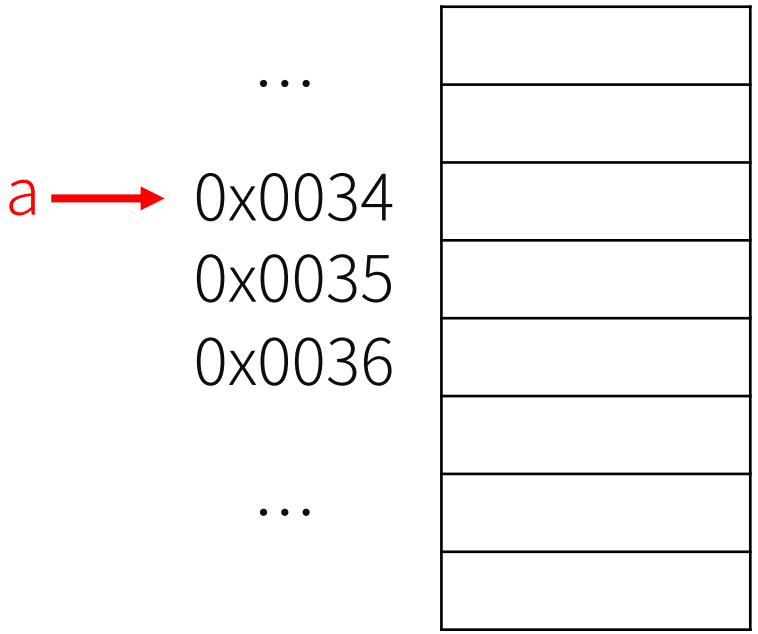
```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5    int n;
6    cin >> n;
7    int *a;
8    a = new int[n];
9
10   for(int i = 0; i < n; i++){
11     cout << i << " => "<<a[i] << endl;
12   }
13 }
```



動態宣告陣列

當無法預知陣列的大小時，而是由使用者輸入後才做決定的話。這時候可以使用動態宣告陣列的技巧，根據程式運行的過程才向記憶體要空間。

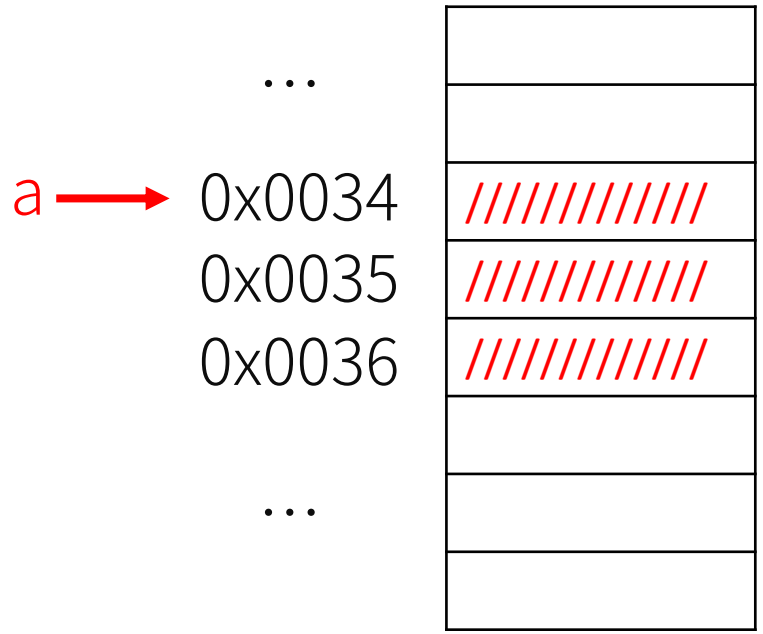
```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5    int n;
6    cin >> n;
7    int *a; → 宣告 a 指標，指向一個記憶體位置
8    a = new int[n];
9
10   for(int i = 0; i < n; i++){
11     cout << i << " => "<<a[i] << endl;
     }
}
```



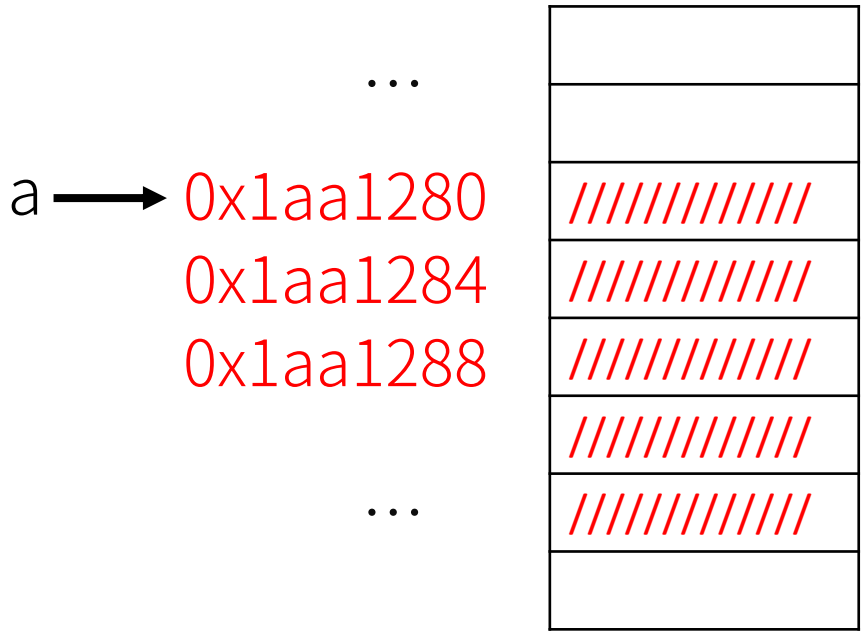
動態宣告陣列

當無法預知陣列的大小時，而是由使用者輸入後才做決定的話。這時候可以使用動態宣告陣列的技巧，根據程式運行的過程才向記憶體要空間。

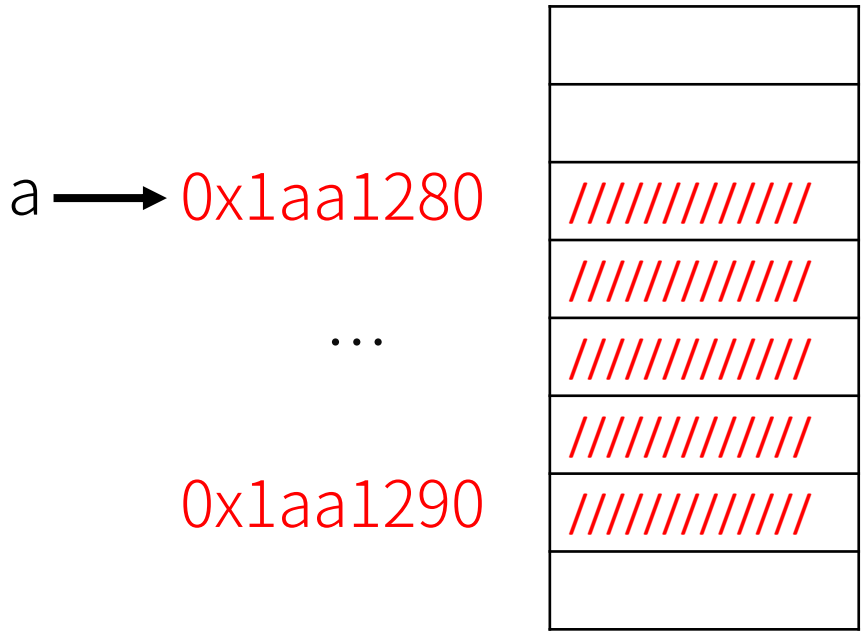
```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5    int n;
6    cin >> n;
7    int *a;
8    a = new int[n]; → 從 a 位置上，新增 n 個大小
9
10   for(int i = 0; i < n; i++){
11     cout << i << " => "<<a[i] << endl;
     }
   }
```



動態宣告陣列實際宣告出來的大小



動態宣告陣列實際宣告出來的大小



二維陣列的動態記憶體配置

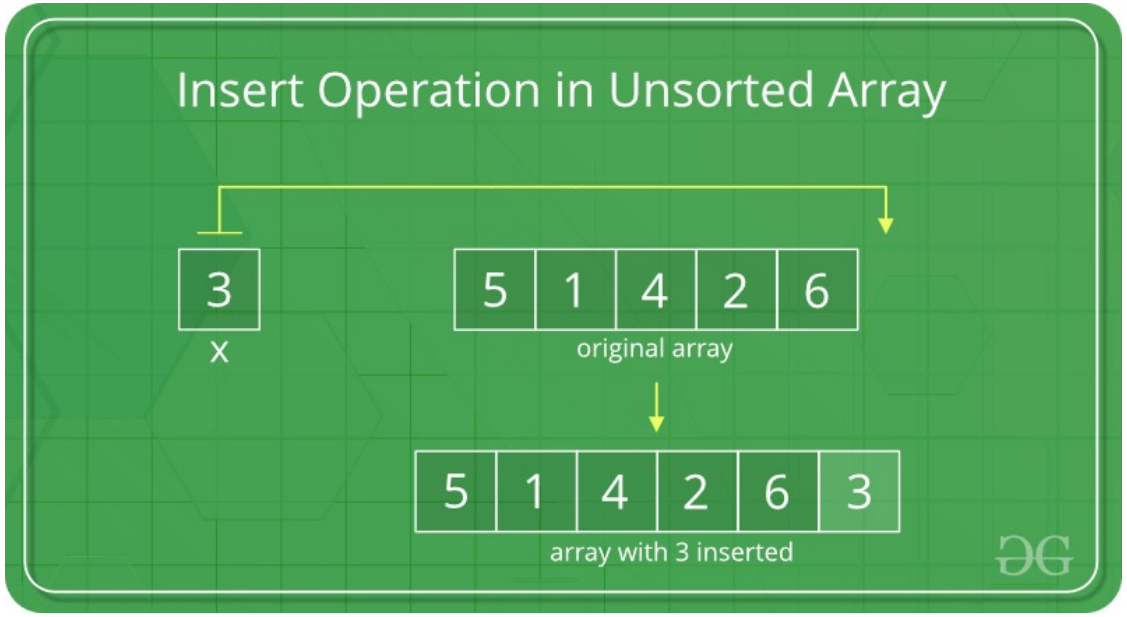
```
   // 動態記憶體配置
1  int **a;
2  a = new int*[n];
3  for(int i = 0; i < n; i++){
4    a[i] = new int[n];
5  }
6
7  for(int i = 0; i < n; i++){
8    for(int j = 0; j < n; j++){
9      a[i][j] = 0;
10     cout << a[i][j] << "\t";
       }
11   cout << endl;
   }
```

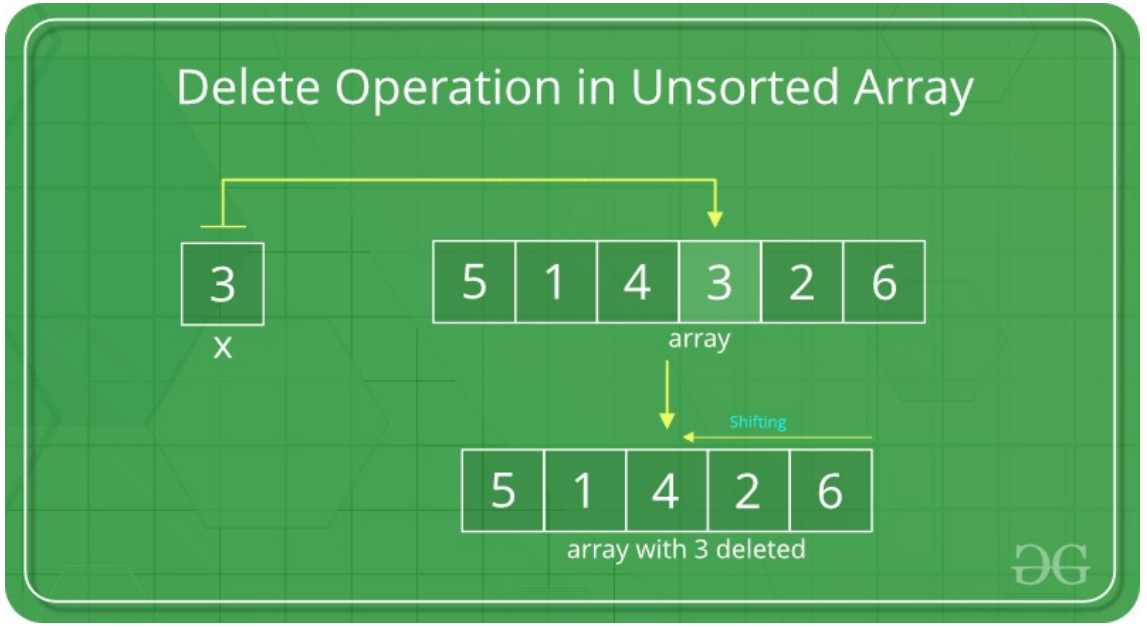
釋放動態記憶體配置空間

```
1  int *ptr;
2  ptr = new int[10];
3
4  // 釋放動態記憶體配置空間
5  delete [] ptr;
```

```
1  int **a;
2  a = new int*[n];
3  for(int i = 0; i < n; i++){
4    a[i] = new int[n];
5  }
6  // 釋放動態記憶體配置空間
7  for(int i = 0; i < n; i++)
8     delete [] a[i];
9  delete [] a;
10
```

Array 佔用連續性的記憶體空間

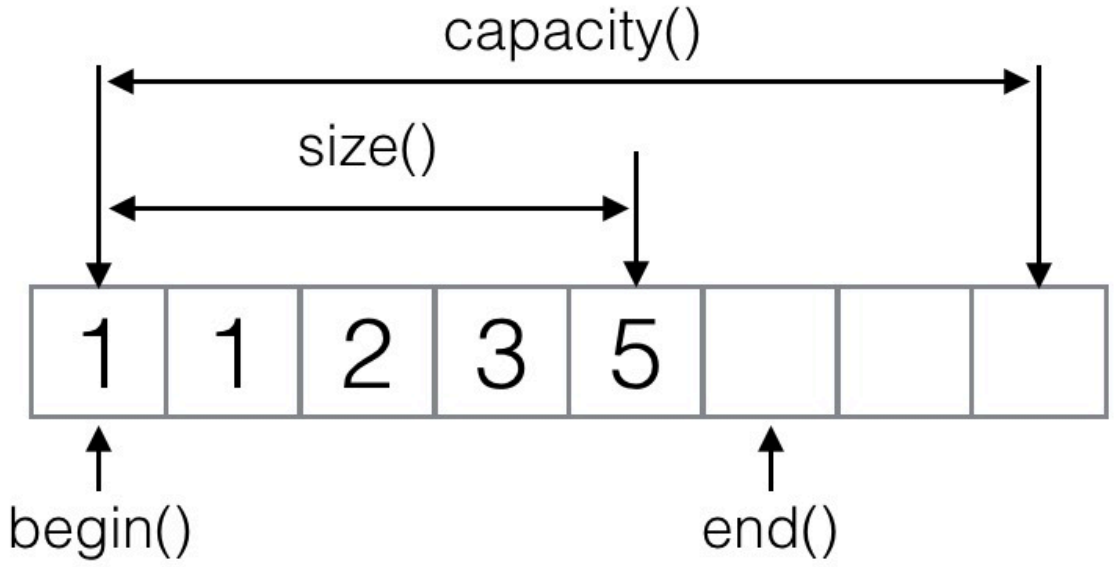




→ 不易新增或刪除的操作

Vector

Vector 是 C++ 標準程式庫中的一個 class，可視為會自動擴展容量的陣列，是C++標準程式庫中的眾多容器（container）之一，以循序 (Sequential) 的方式維護變數集合。



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

   int main() {
 1   // initialize
 2   int a[4] = {1, 2, 3};
 3
 4   // push
 5   a[3] = 4;
 6
 7   // insert
 8   a[3] = a[2];
 9   a[2] = a[1];
10   a[1] = 999;
11
     int n = sizeof(a) / sizeof(a[0]);
     for(int i = 0; i < n; i++)
       cout << a[i] << endl;
   }
```

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

 1
 2 int main() {
 3   // initialize
 4   vector<int> v{ 1, 2, 3 };
 5
 6   // push
 7   v.push_back(4);
 8
 9   // insert
10   v.insert(v.begin()+1, 999);
11
     for(int i = 0; i < v.size(); i++)
       cout << v[i] << endl;
   }
```

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

int main() {
  // initialize
  int a[4] = {1, 2, 3};

  // push
  a[3] = 4;

  // insert
  a[3] = a[2];
  a[2] = a[1];
  a[1] = 999;

  int n = sizeof(a) / sizeof(a[0]);
  for(int i = 0; i < n; i++)
    cout << a[i] << endl;
}
```

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

int main() {
  // initialize
  vector<int> v{ 1, 2, 3 };

  // push
  v.push_back(4);

  // insert
  v.insert(v.begin()+1, 999);

  for(int i = 0; i < v.size(); i++)
    cout << v[i] << endl;
}
```

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

   int main() {
 1   // initialize
 2   int a[4] = {1, 2, 3};
 3
 4   // push
 5   a[3] = 4;
 6
 7   // insert
 8   a[3] = a[2];
 9   a[2] = a[1];
10   a[1] = 999;
11
     int n = sizeof(a) / sizeof(a[0]);
     for(int i = 0; i < n; i++)
       cout << a[i] << endl;
   }
```

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

 1
 2 int main() {
 3   // initialize
 4   vector<int> v{ 1, 2, 3 };
 5
 6   // push
 7   v.push_back(4);
 8
 9   // insert
10   v.insert(v.begin()+1, 999);
11
     for(int i = 0; i < v.size(); i++)
       cout << v[i] << endl;
   }
```

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

   int main() {
 1   // initialize
 2   int a[4] = {1, 2, 3};
 3
 4   // push
 5   a[3] = 4;
 6
 7   // insert
 8   a[3] = a[2];
 9   a[2] = a[1];
10   a[1] = 999;
11
     int n = sizeof(a) / sizeof(a[0]);
     for(int i = 0; i < n; i++)
       cout << a[i] << endl;
   }
```

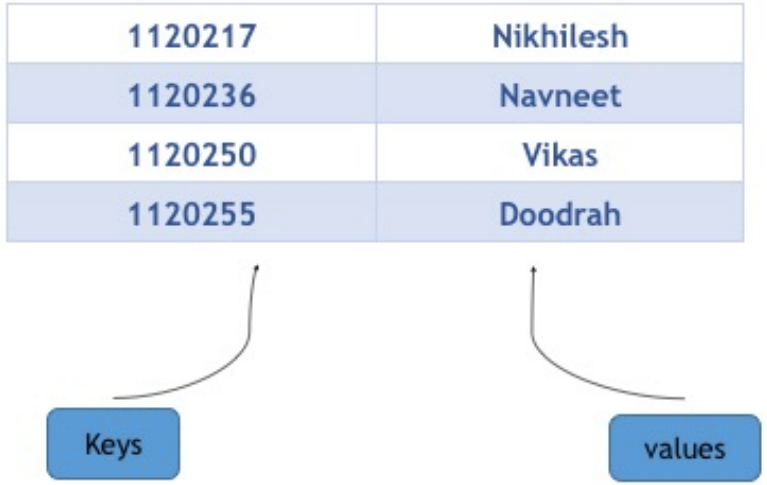
```
   #include <iostream>
   #include <vector>
   using namespace std;

 1
   int main() {
 2   // initialize
 3   vector<int> v{ 1, 2, 3 };
 4
 5   // push
 6   v.push_back(4);
 7
 8   // insert
 9   v.insert(v.begin()+1, 999);
10
11   for(int i = 0; i < v.size(); i++)
       cout << v[i] << endl;
   }
```

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

   int main() {
 1   // initialize
 2   int a[4] = {1, 2, 3};
 3
 4   // push
 5   a[3] = 4;
 6
 7   // insert
 8   a[3] = a[2];
 9   a[2] = a[1];
10   a[1] = 999;
11
     int n = sizeof(a) / sizeof(a[0]);
     for(int i = 0; i < n; i++)
       cout << a[i] << endl;
   }
```

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

 1 int main() {
 2   // initialize
 3   vector<int> v{ 1, 2, 3 };
 4
 5   // push
 6   v.push_back(4);
 7
 8   // insert
 9   v.insert(v.begin()+1, 999);
10
11   int n = v.size();
     for(int i = 0; i < n; i++)
       cout << v[i] << endl;
   }
```

Map

Map 是 C++ 標準程式庫中的一個 class，為眾多容器（container）之一。它提供搜尋和插入友善的資料結構，並具有一對一 mapping 功能。第一個稱為關鍵字 (key)，每個關鍵字只能在 map 中出現一次。第二個稱為該關鍵字的值 (value)。

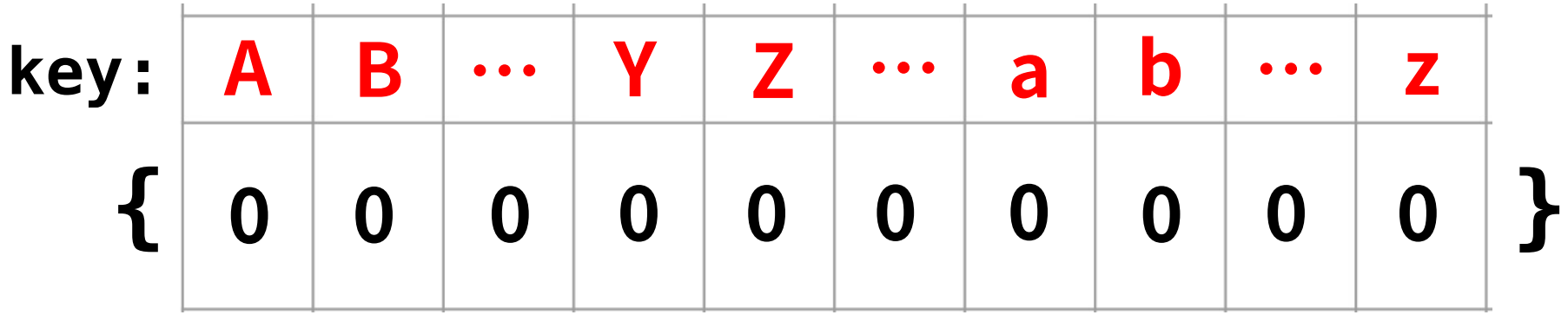


利用陣列索引表示儲存字元

char:	A	B	…	Y	Z	…	a	b	…	z
index:	65	66	…	89	90	…	97	98	…	122
{	0	0	0	0	0	0	0	0	0	0 }

→ 把字母的 ascii code 當成 Index，Value 當成出現次數

Map



→ Map 可以自定義 key-value 的組合

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

int main() {
  int A[100];
  A[int('A')-65] = 999;
  A[int("z"[0])-65] = 888;

  for(int i = 0; i <= 57; i++){
    if(A[i] > 0)
      cout << char(i+65) << " => "
      cout << A[i] << endl;
  }
}
```

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

int main() {
  map<char, int> m;
  m['A'] = 999;
  m['z'] = 888;

  map<char,int>::iterator it;
  for(it = m.begin(); it != m.end();
      ++it){
    cout << it->first << " => ";
    cout << it->second << endl;
  }
}
```

Thanks for listening.

元智大學 | C++ 程式設計實習
Wei-Yuan Chang