04 陣列與向量（I）

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。

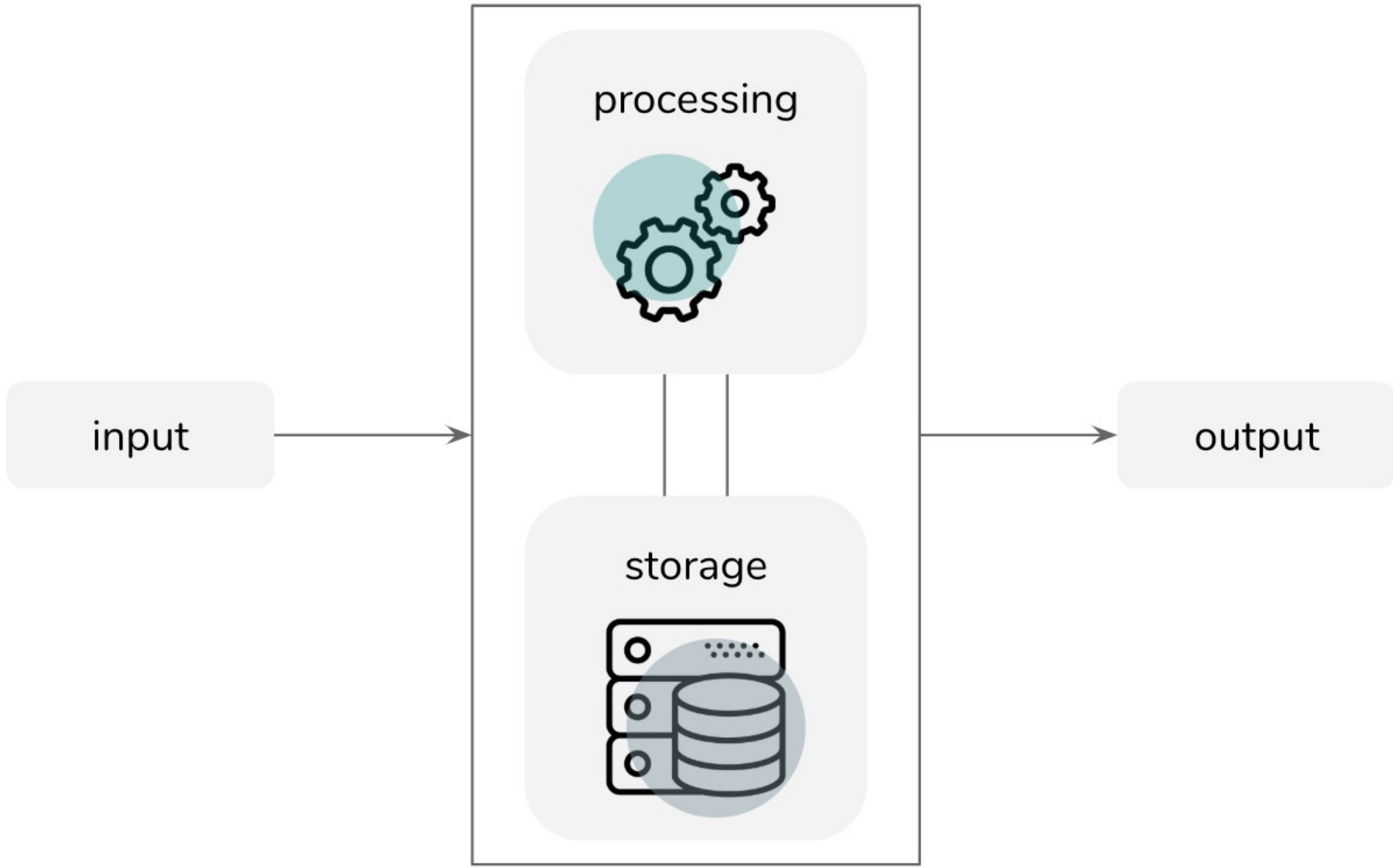
- 2018 總統盃黑客松 冠軍隊伍
- 2017 資料科學愛好者年會(DSCONF) 講師
- 2017 行動科技年會(MOPCON) 講師
- 2016 微軟 Imagine Cup 台灣區冠軍

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 #10

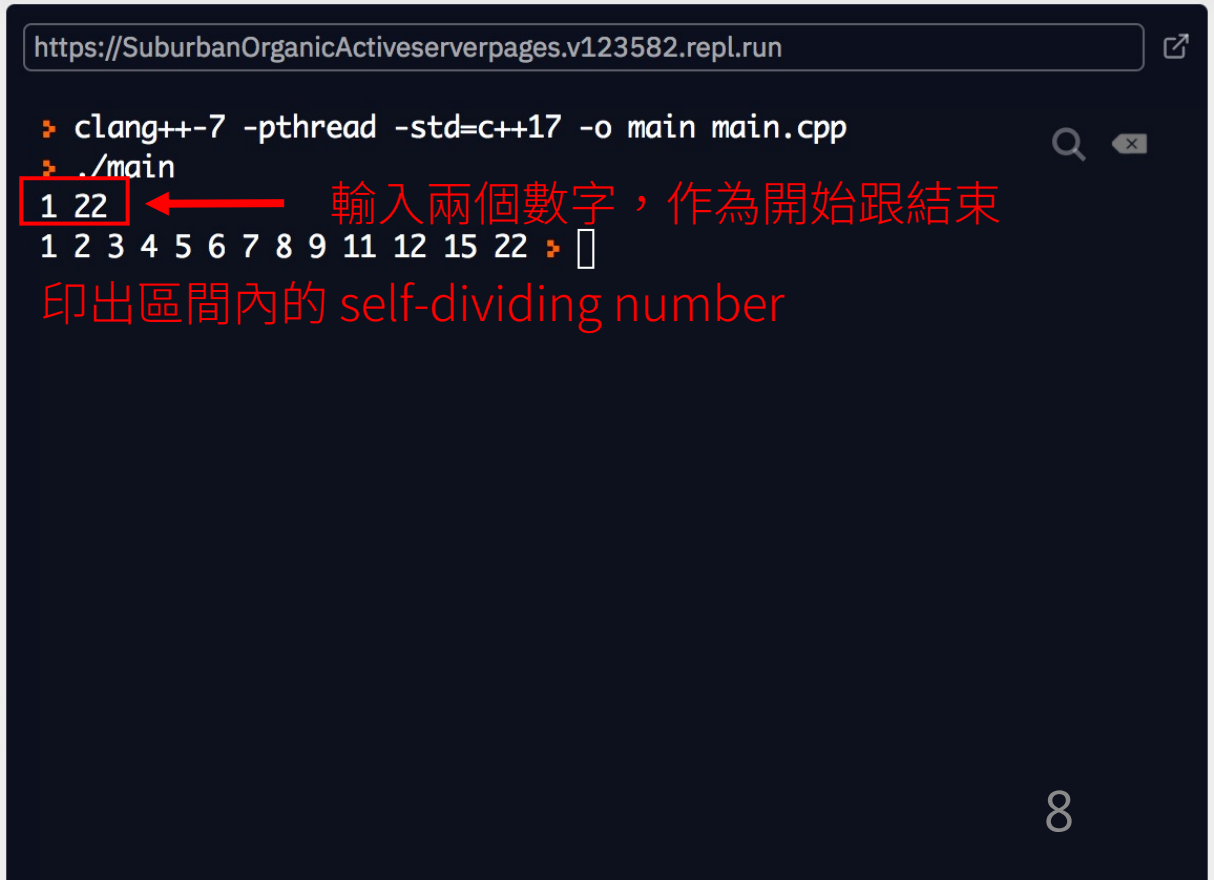
- #練習：A self-dividing number is a number that is divisible by every digit it contains. 例如，128 是一個 self-dividing，因為 1、2、8 都可以被 128 整除。
- Requirements：
  1. 輸入兩個正整數代表 left 和 right
  2. 請輸出 left 和 right 間的 self-dividing number
- Sample Input： 1, 22
- Sample Output： 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 15, 22

參考程式碼與結果

- 你可以參考下列程式碼修改，也可以自己從頭開始寫。只要執行結果必須符合右邊的格式即可。

main.cpp

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

int main() {
  int left, right;
  cin >> left >> right;

  for(int i = left; i <= right; i++){
    cout << i << endl;
    // Your Code
  }
  return 0;
}
```



解題思路

- 1. 先計算一個數字怎麼判斷 self-dividing number
- 2. 再從 left 檢查到 right
- 3. 檢查邊界條件

解題思路

- 1. 先計算一個數字怎麼判斷 self-dividing number
  - self-dividing number = 每個位置都可以被整除

解題思路

- 1. 先計算一個數字怎麼判斷 self-dividing number
  - self-dividing number = 每個位置都可以被整除

解題思路

- 1. 先計算一個數字怎麼判斷 self-dividing number
  - self-dividing number = 每個位置都可以被整除

解題思路

- 1. 先計算一個數字怎麼判斷 self-dividing number
  - self-dividing number = 每個位置都可以被整除
    = 被整除的數量 = 數字長度

解題思路

- 1. 先計算一個數字怎麼判斷 self-dividing number
  - self-dividing number = 每個位置都可以被整除
    = 被整除的數量 = 數字長度
  - 不是 self-dividing number = 至少一個不可以被整除

解題思路

- 1. 先計算一個數字怎麼判斷 self-dividing number
  - self-dividing number = 每個位置都可以被整除
    = 被整除的數量 = 數字長度
  - 不是 self-dividing number = 至少一個不可以被整除
    = 先假設為 True，只要發現有一個不能整除就代表不是

解題思路

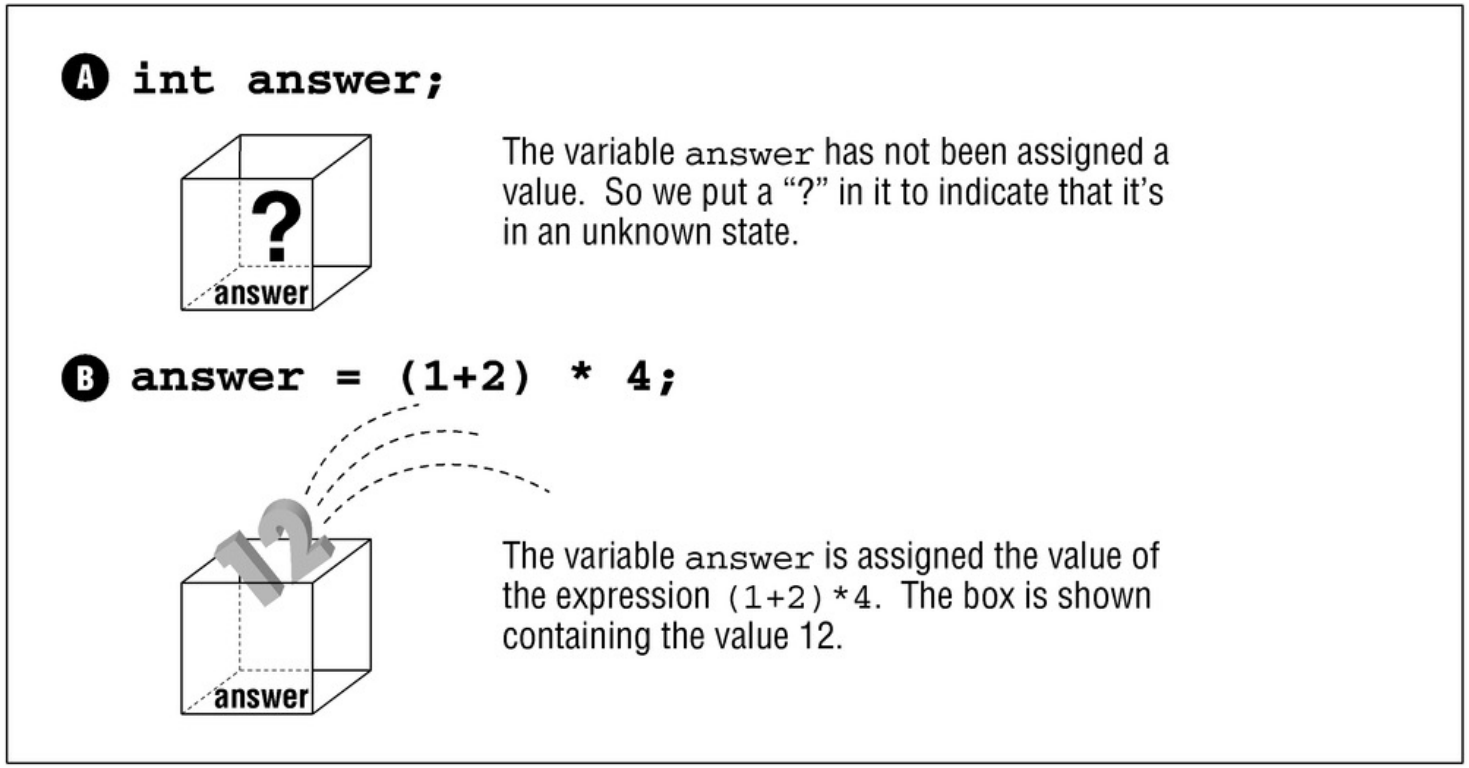
- 1. 先計算一個數字怎麼判斷 self-dividing number
  - self-dividing number = 每個位置都可以被整除
    = 被整除的數量 = 數字長度
  - 不是 self-dividing number = 至少一個不可以被整除
    = 先假設為 True，只要發現有一個不能整除就代表不是
- 2. 再從 left 檢查到 right

解題思路

- 1. 先計算一個數字怎麼判斷 self-dividing number
  - self-dividing number = 每個位置都可以被整除
  - 不是 self-dividing number = 至少一個不可以被整除
- 2. 再從 left 檢查到 right
- 3. 檢查邊界（例外）條件

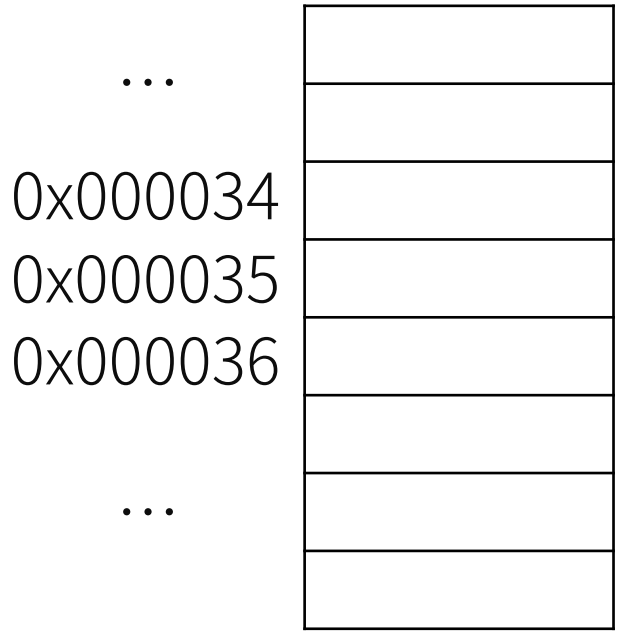
變數（Variables）與常數

C++在使用變數之前，必須先告訴電腦「我要使用變數」，電腦會幫我們準備記憶空間儲存該變數，這件事稱之為變數宣告。

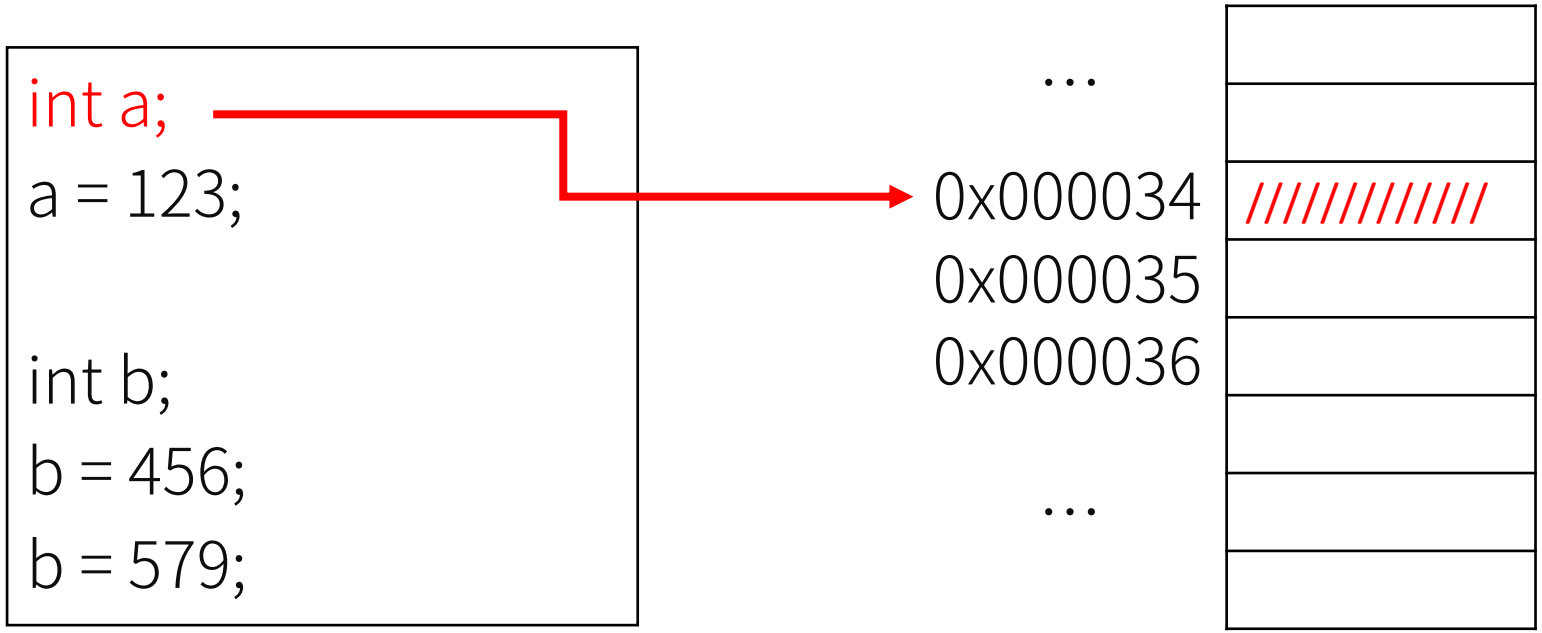


變數會存在電腦的記憶體裡

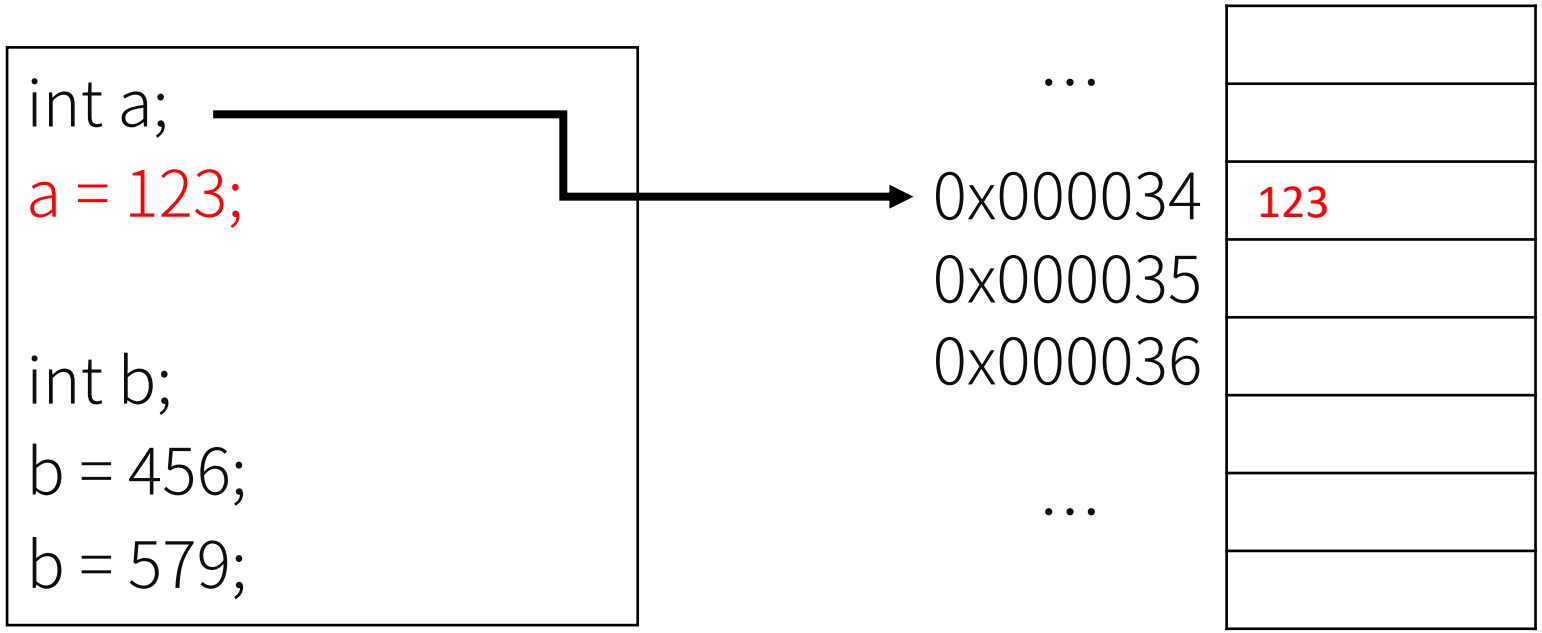
```
int a;
a = 123;

int b;
b = 456;
b = 579;
```



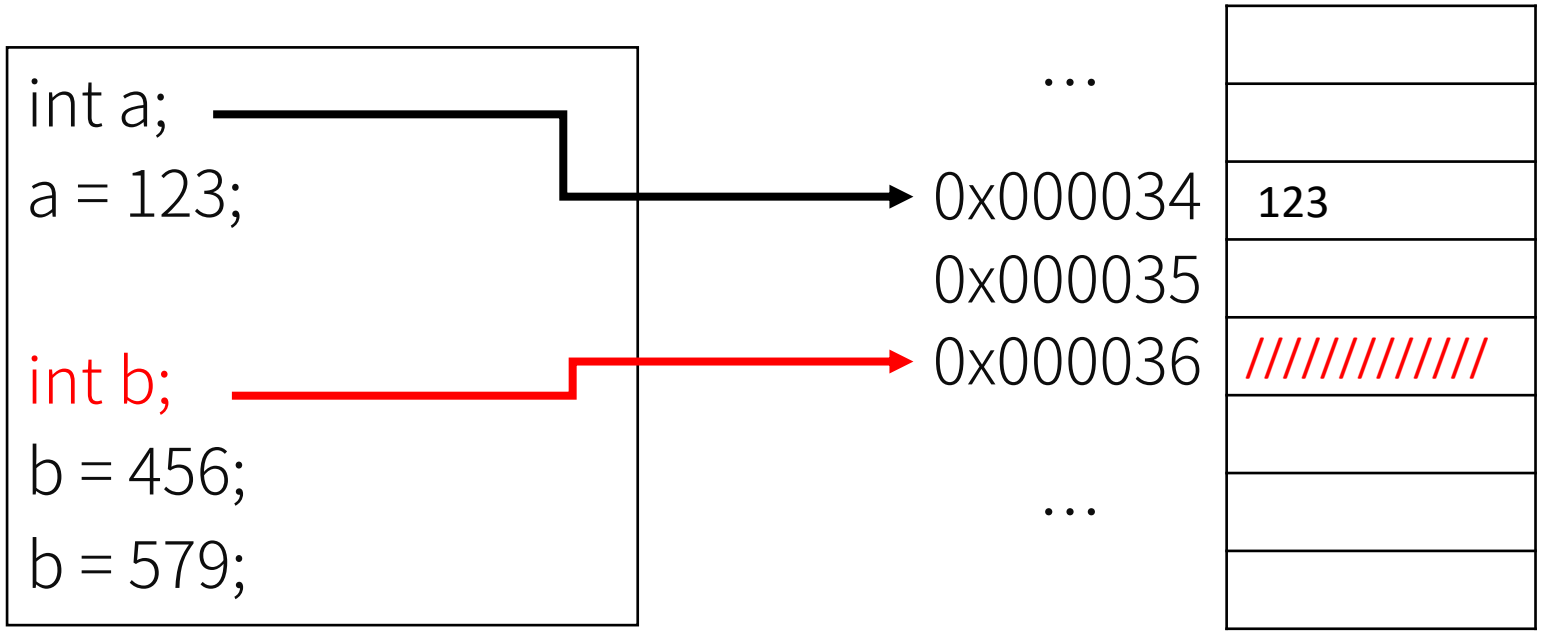
變數會存在電腦的記憶體裡



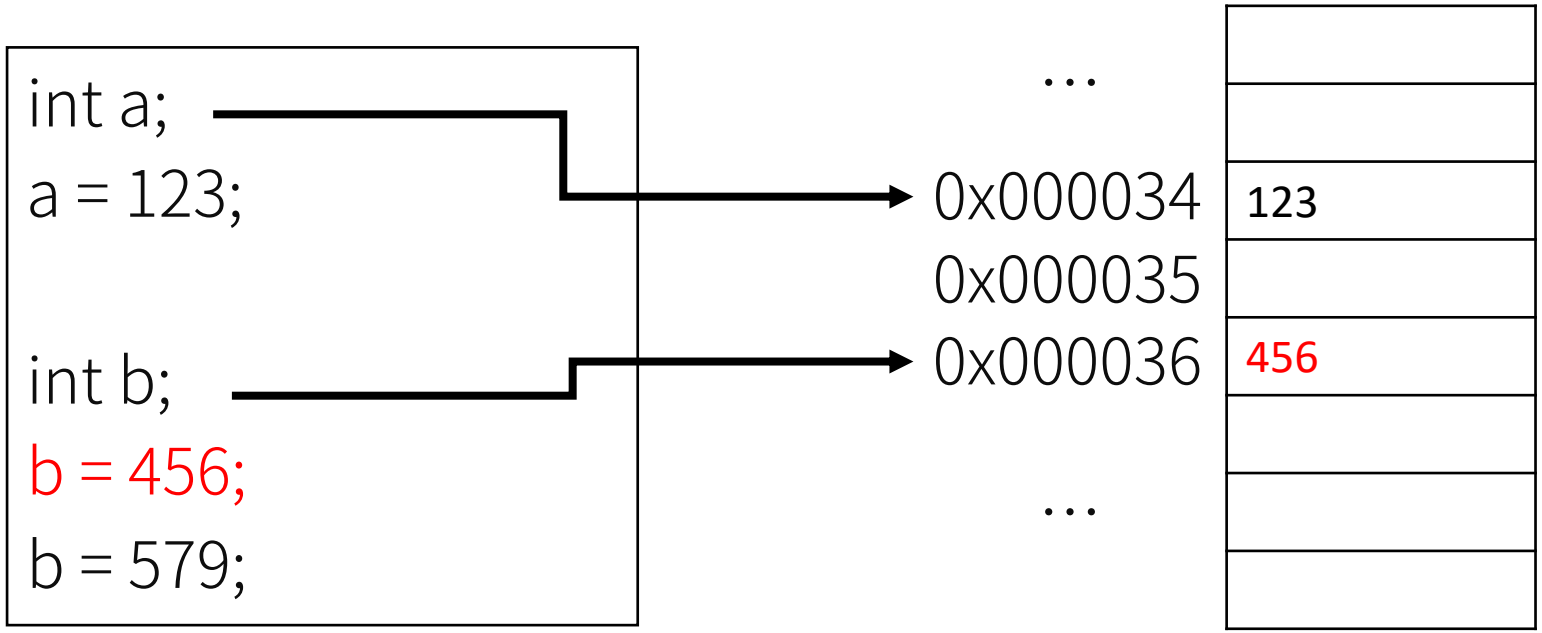
變數會存在電腦的記憶體裡



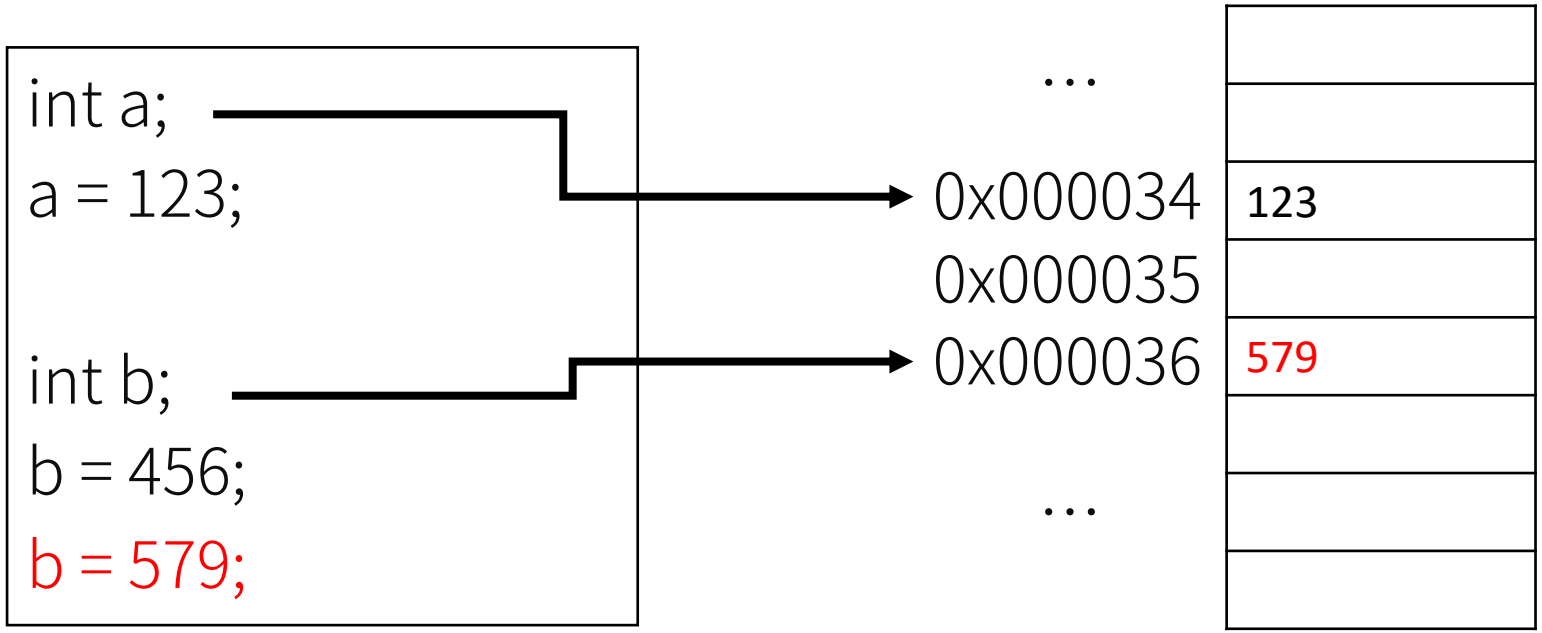
變數會存在電腦的記憶體裡



變數會存在電腦的記憶體裡

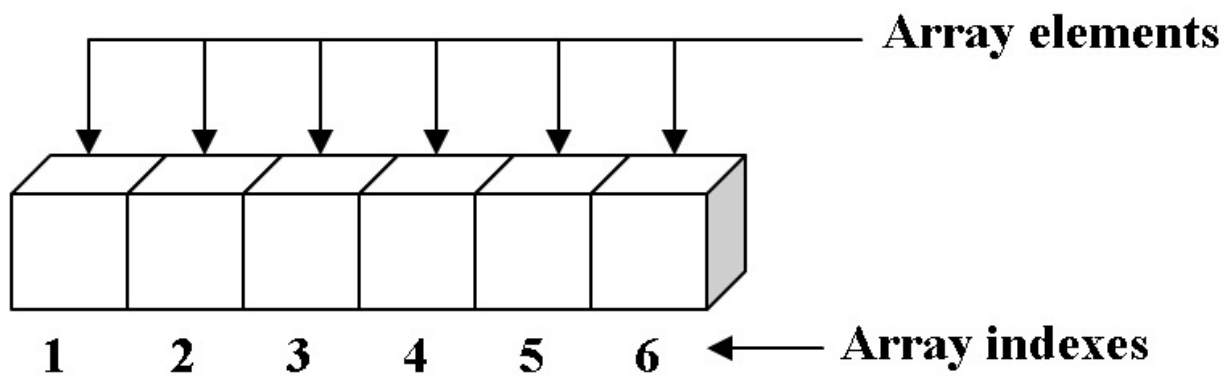


變數會存在電腦的記憶體裡



陣列（Array）

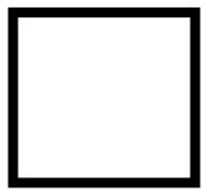
由相同型態的元素所組成的有序串列，會佔用連續性的記憶體空間。



One-dimensional array with six elements

變數與陣列

int a;



a

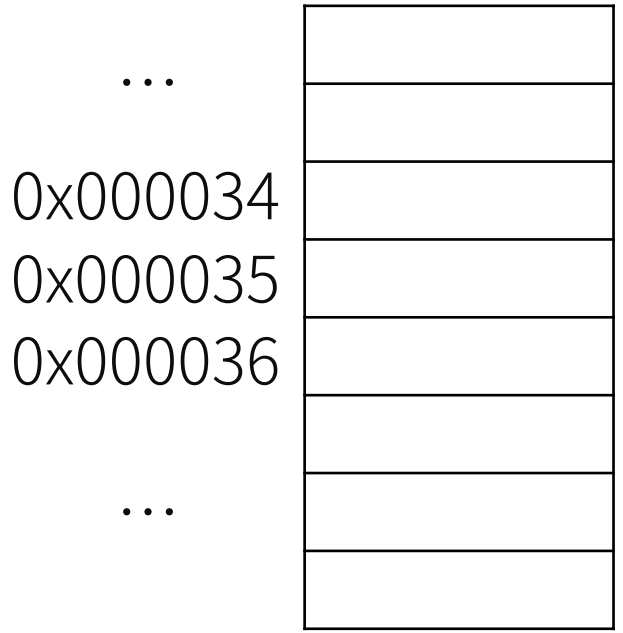
int a[5];

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

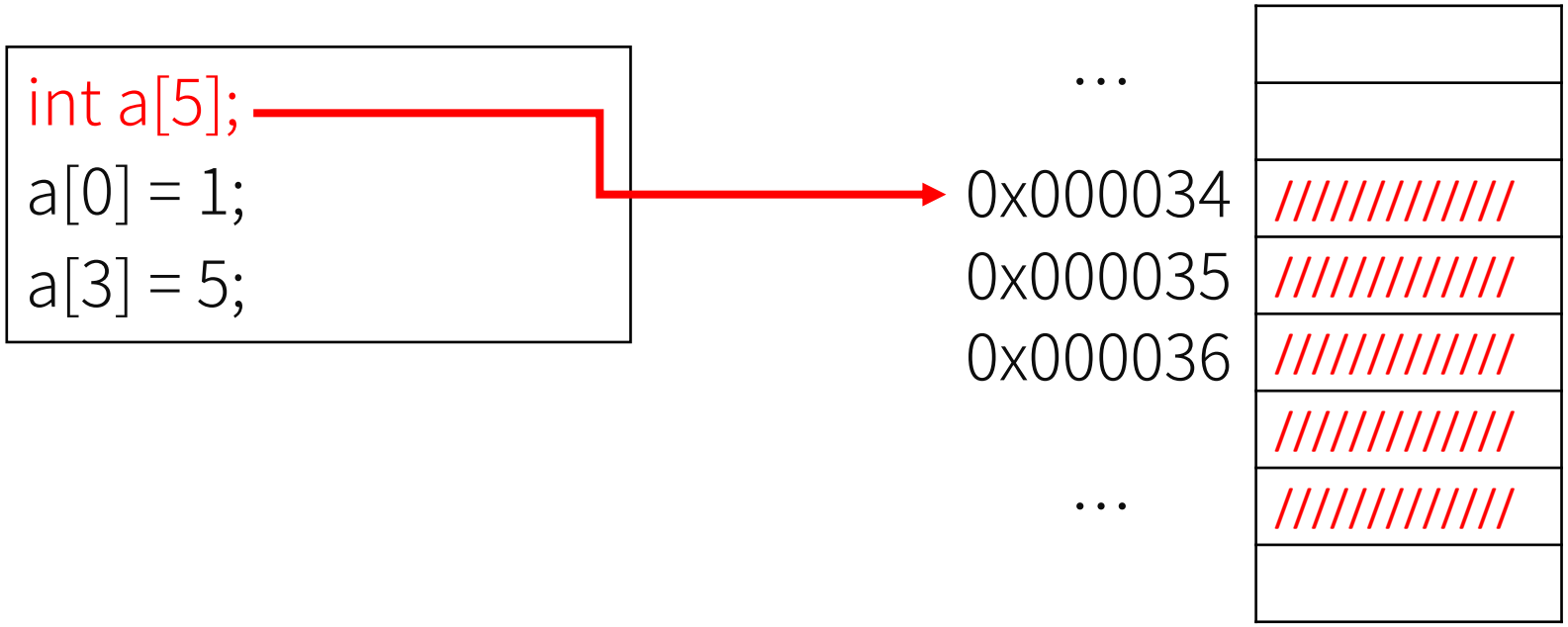
→ 陣列的長度要預先定義

陣列會佔用連續的記憶體空間

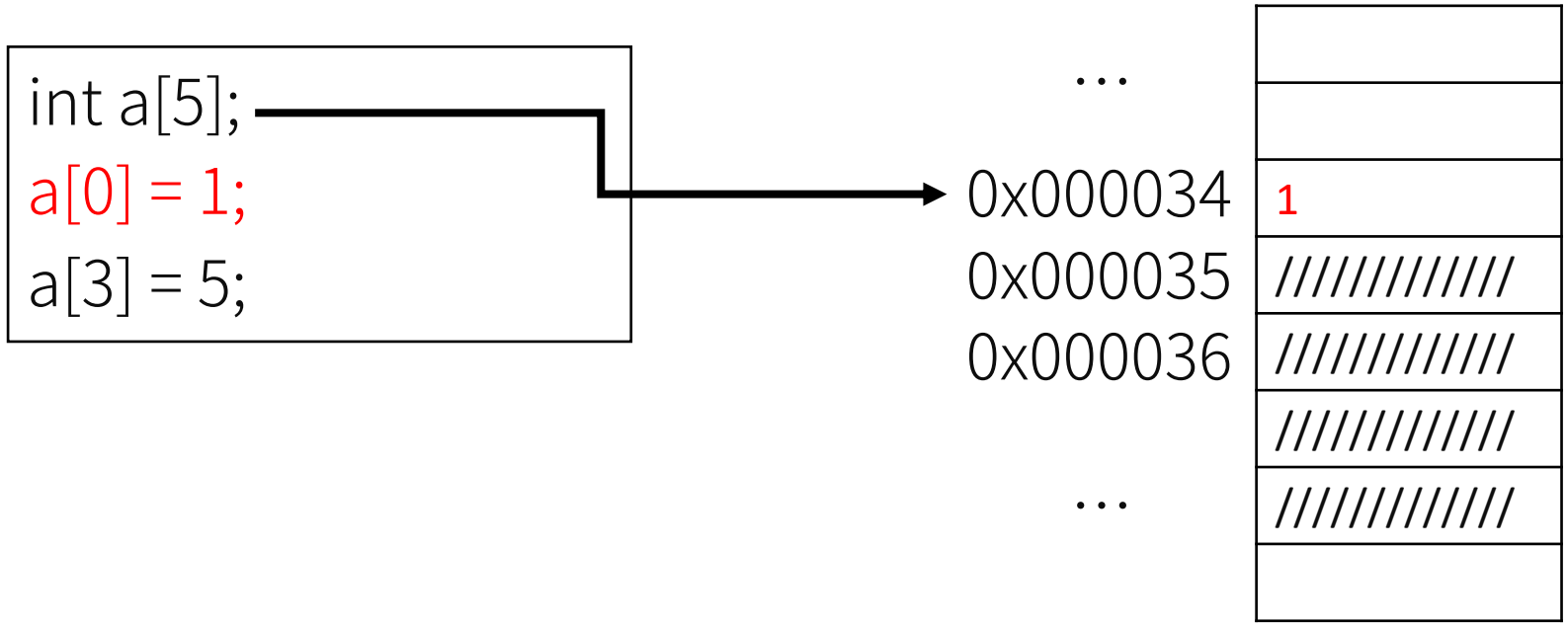
```
int a[5];
a[0] = 1;
a[3] = 5;
```



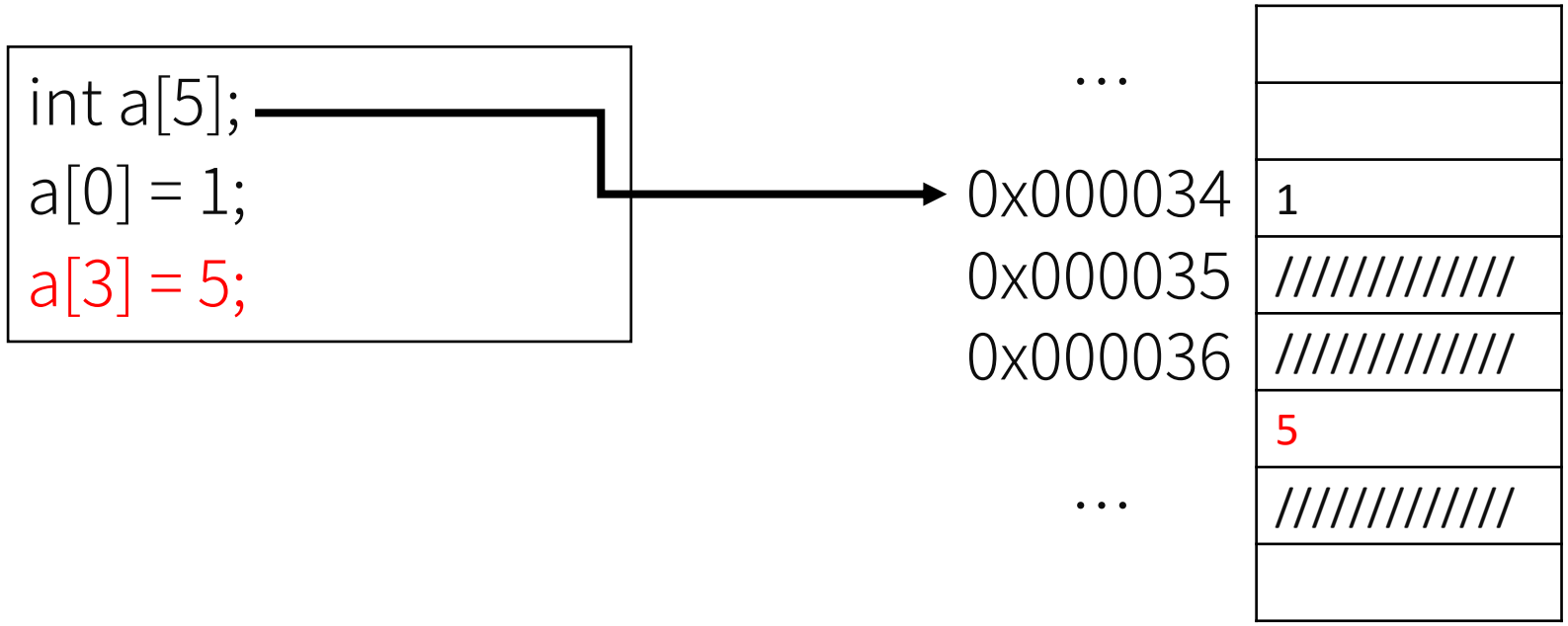
陣列會佔用連續的記憶體空間



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



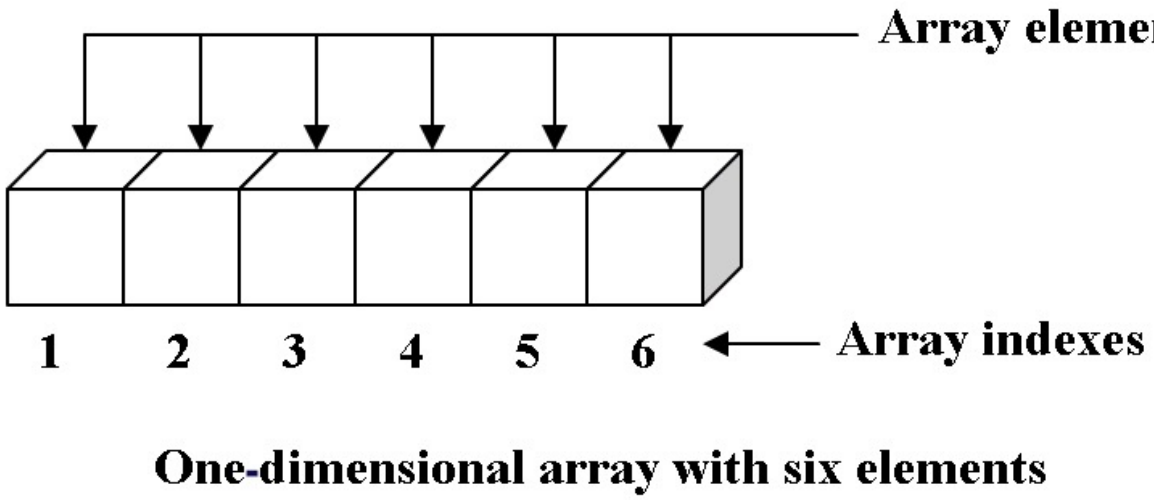
陣列是由連續的位置所組成的容器

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

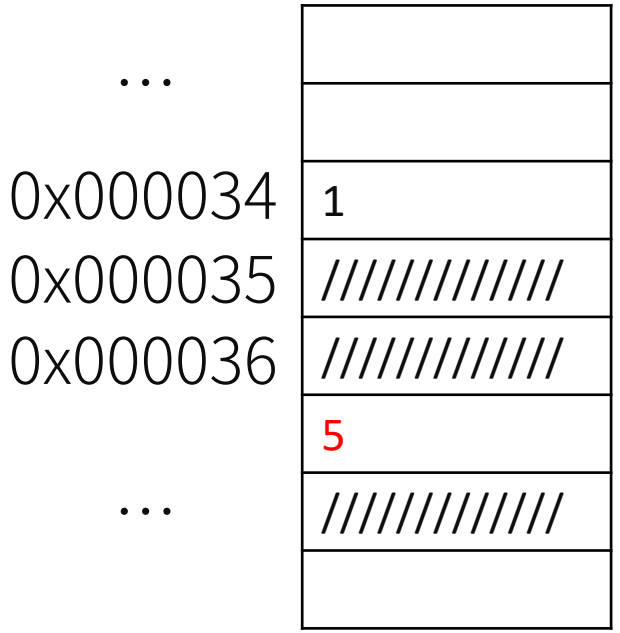
- 以 { } 語法宣告陣列，以 [ ] 語法來取值與賦值

```
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 }
```

#動手實作

- #練習：想一下，如果陣列是由使用者輸入的，你怎麼知道最後一位是哪一個？

```
1  #include<iostream>
2  #define MAXN 10
3  using namespace std;
4
5  int a[MAXN];
6  int main(){
7    int x, n = 0;
8    while(cin >> x){
9      a[n++] = x;
10   }
11
12   return 0;
13 }
```

寫入的時候同時也要記錄位置

```
1  #include<iostream>
2  #define MAXN 10
3  using namespace std;
4
5  int a[MAXN];
6  int main(){
7
8    int x, n = 0;
9    while(cin >> x){
10     a[n++] = x; → n 代表有填上數值的個數
11   }
12
13   for(int i = 0; i < n; i++){
14     cout << i << " => " << a[i] << endl;
15   }
16
17   return 0;
18 }
```

可以在一開始給初始值

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

可以在一開始給初始值

```
   #include<iostream>
 1 using namespace std;
 2
 3 int a[10] = {}; → 初始值為 0，可能會受到作業系統影響而不同
 4 int main(){
 5   for(int i = 0; i < 10; i++){
 6     a[i] = 0;                    → 手動初始為 0
 7   }
 8   for(int i = 0; i < 10; i++){
 9     cout << i << " => " << a[i] << endl;
10   }
11   return 0;
12 }
13
```

可以在一開始給初始值

```
1  #include<iostream>
2  using namespace std;
3
4  int a[10] = {1, 2, 3}; → 設定前三個為 1, 2, 3，其餘為 0
5  int main(){
6
7    for(int i = 0; i < 10; i++){
8      cout << i << " => " << a[i] << endl;
9    }
10   return 0;
11 }
12
13
```

可以省略宣告陣列長度

```
1  #include<iostream>
2  using namespace std;
3
4  int a[] = {1, 2, 3};   → 定義一個設定 [1, 2, 3] 長度為 3 的陣列
5  int main(){
6
7    for(int i = 0; i < 10; i++){
8      cout << i << " => " << a[i] << endl;
9    }
10   return 0;
11 }
12
13
```

利用所佔大小計算陣列長度

```
1  #include<iostream>
2  using namespace std;
3
4  int a[] = {1, 2, 3};
5  int main(){
6
7    int n = sizeof(a)/sizeof(a[0]); → 陣列長度 = 陣列容量 / 一個的容量
8    for(int i = 0; i < n; i++){
9      cout << i << " => " << a[i] << endl;
10   }
11   return 0;
12 }
13
```

一維陣列與二維陣列

i	0	1	2	3	4	5	6	7	8	9	10
A[i]	1	2	3	4	5	6	7	8	9	10	11

x	0				1			
y	0	1	2	3	0	1	2	3
A[x][y]	10	11	12	13	14	15	16	17

x / y	0	1	2	3
0	10	11	12	13
1	14	15	16	17
2	18	19	20	21
3	22	23	24	25

一維陣列與二維陣列

i	0	1	2	3	4	5	6	7	8	9	10
A[i]	1	2	3	4	5	6	7	8	9	10	11

```
1
2
3
4
5
6
7
8
9
10
11
12
13
```

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

int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11};
int main(){

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

一維陣列與二維陣列

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

x	0				1			
y	0	1	2	3	0	1	2	3
A[x][y]	10	11	12	13	14	15	16	17

一維陣列與二維陣列

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

→ a[0]

→ a[1]

x	0				1			
y	0	1	2	3	0	1	2	3
A[x][y]	10	11	12	13	14	15	16	17

→ a[0]

a[0][0], a[0][1], a[0][2], a[0][3]

一維陣列與二維陣列

```
1  #include<iostream>
2  using namespace std;
3  int a[2][4] = {
4    {10, 11, 12, 13},
5    {14, 15, 16, 17}
6  };
7  int main(){
8    int m = 2; →x 的長度
9    int n = 4; →y 的長度
10
11   for(int i = 0; i < m; i++){
12     cout << i << " => " << endl;
13     for(int j = 0; j < n; j++){
14       cout << "\t" << j << " => " << a[i][j] << endl;
15     }
16   }
17   return 0;
18 }
```

x	0				1			
y	0	1	2	3	0	1	2	3
A[x][y]	10	11	12	13	14	15	16	17

一維陣列與二維陣列

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

x	0				1			
y	0	1	2	3	0	1	2	3
A[x][y]	10	11	12	13	14	15	16	17

→ x 的長度

→ y 的長度

二維陣列當成方陣

```
#include<iostream>
using namespace std;
int a[4][4] = {
  {10, 11, 12, 13},
  {14, 15, 16, 17},
  {18, 19, 20, 21},
  {22, 23, 24, 25}
};
int main(){

  int n = sizeof(a)/sizeof(a[0]);
  int m = sizeof(a[0])/sizeof(a[0][0]);

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

x / y	0	1	2	3
0	10	11	12	13
1	14	15	16	17
2	18	19	20	21
3	22	23	24	25

Thanks for listening.

元智大學 | C++ 程式設計實習
Wei-Yuan Chang