



02 陣列、向量與結構

Fundamental Computer Programming- C++ Lab (II)

元智大學 | C++ 程式設計實習 (II)

張維元

課程投影片：[請從元智個人 Portal 下載](#)

Outline

- 01 物件導向與開發環境
- 02 陣列、向量與結構
- 03 類別與物件
- 04 物件導向程式設計: 多載
- 05 物件導向程式設計: 繼承
- 06 樣板
- 07 C++ 進階用法

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。同時經營資料科學家的工作日常 Facebook 粉專 與 Instagram 社群，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。同時經營資料科學家的工作日常 Facebook 粉專 與 Instagram 社群，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

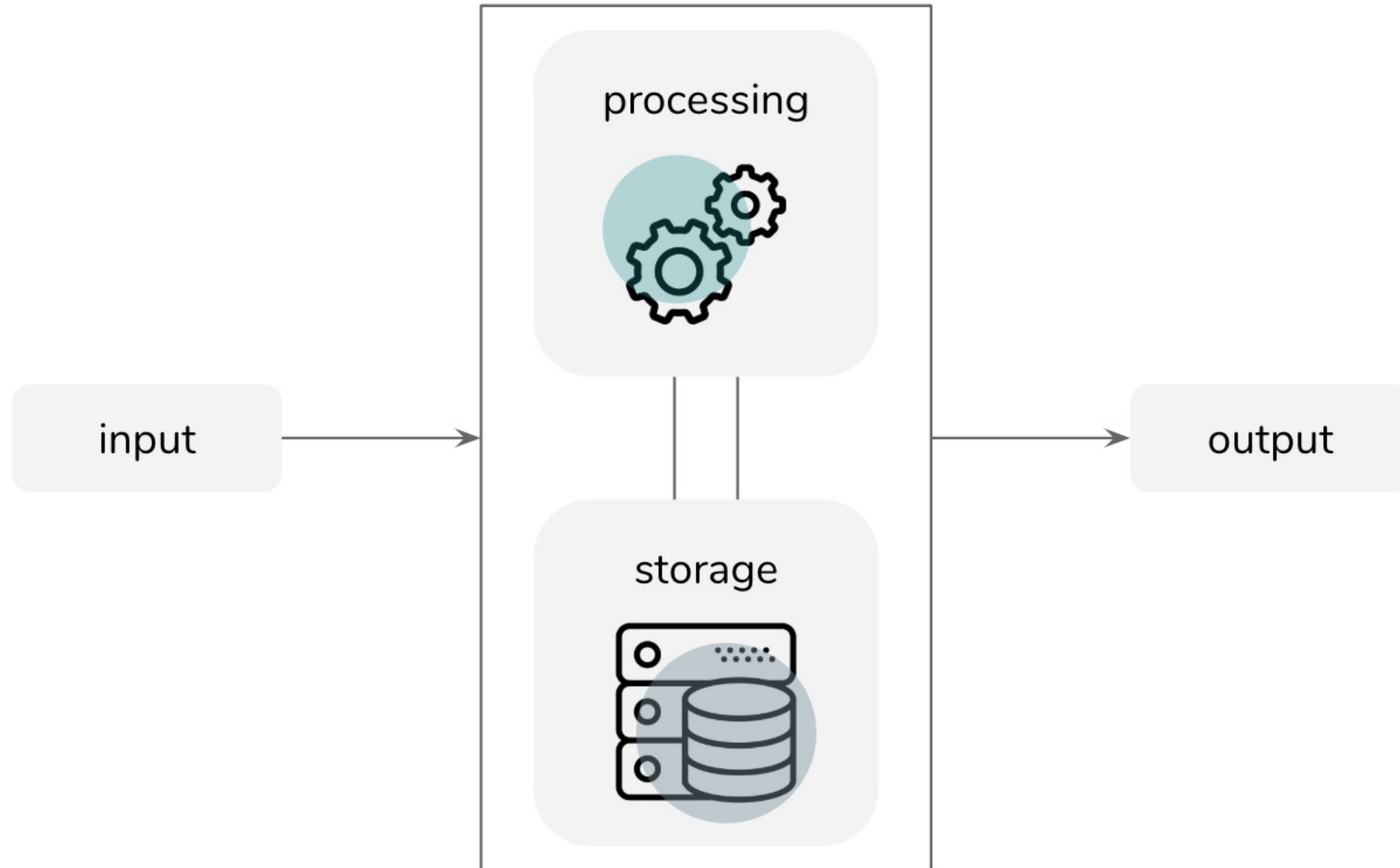


物件導向程式設計

物件導向程式設計（Object-oriented programming，OOP）是種具有物件概念的程式設計典範，同時也是一種程式開發的抽象方針。它可能包含資料、屬性、程式碼與方法。物件則指的是類別（class）的實例。它將物件作為程式的基本單元，將程式和資料封裝其中，以提高軟體的重用性、靈活性和擴充性。

把物件作為程式最小的單位，模擬實體世界的運作



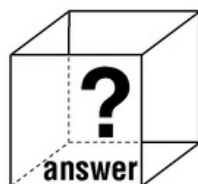


變數 (Variables)

C++ 在使用變數之前，必須先告訴電腦「我要使用變數」。電腦會幫在記憶體中準備一個空間用來儲存變數，稱之為變數宣告。

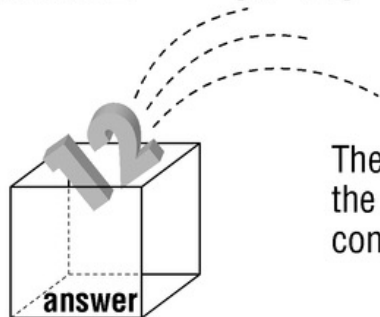
變數宣告

A `int answer;`



The variable `answer` has not been assigned a value. So we put a “?” in it to indicate that it’s in an unknown state.

B `answer = (1+2) * 4;`

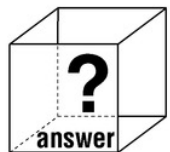


The variable `answer` is assigned the value of the expression $(1+2) * 4$. The box is shown containing the value 12.

變數定義

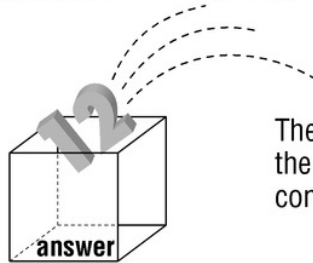
C++ 是一種強型別變數的程式語言

A `int answer;`



The variable `answer` has not been assigned a value. So we put a “?” in it to indicate that it’s in an unknown state.

B `answer = (1+2) * 4;`



The variable `answer` is assigned the value of the expression $(1+2) * 4$. The box is shown containing the value 12.

型態	中文意思	英文字義
<code>int</code>	整數	Integer
<code>float</code>	浮點數(小數)	floating point
<code>char</code>	字元(半形字)	Character
<code>string</code>	字串(文句)	String
<code>bool</code>	布林(是非)	boolean

→ 變數本身會綁定型態，相同型態的變數佔用空間是一樣

強型別變數

變數的型態

型態	意義	英文原意	可儲存的資料
int	整數	Integer	100, -5, 1024 ...
float	浮點數 (小數)	floating point	3.14, 6.9, -8.7 ...
double	雙精度浮點數	Double floating point	3.14, 6.9, -8.7 ...
bool	布林	Boolean	true, false
char	字元 (半形字)	Character	'a', 'e', '1', '-', ...
string	字串 (文句)	String	"Hello", "Hello World" ...

→ 不同的形態所佔用的空間不同，可表示的範圍也不同

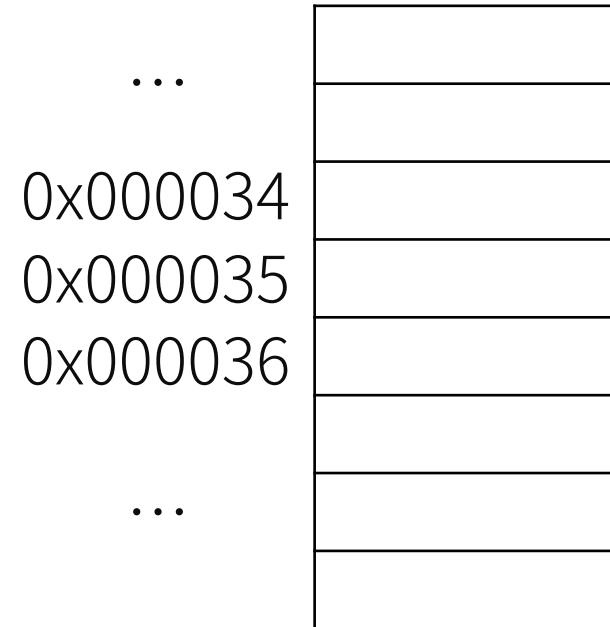
不同的形態所佔用的空間不同

→ 可表示的範圍也不同

Type	Width	Min Value	Max Value
signed char	8 bits	-128	127
unsigned char	8 bits	0	255
short	16 bits	-32768	32767
unsigned short	16 bits	0	65535
int	32 bits	-2147483648	2147483647
unsigned int	32 bits	0	4294967295
long	32 bits	-2147483648	2147483647
unsigned long	32 bits	0	4294967295
long long	64 bits	-9223372036854775808	9223372036854775807
unsigned long long	64 bits	0	18446744073709551615

變數會存在電腦的記憶體裡

```
int a;  
a = 123;  
  
int b;  
b = 456;  
b = 579;
```

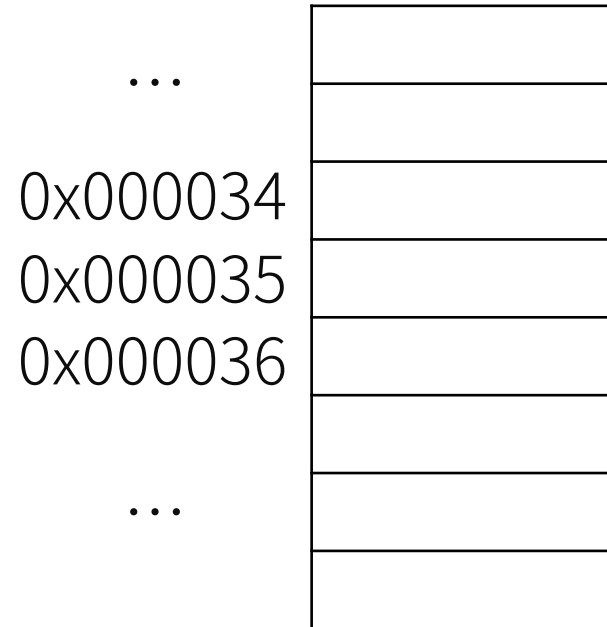


變數會存在電腦的記憶體裡

循序的
(Sequential)

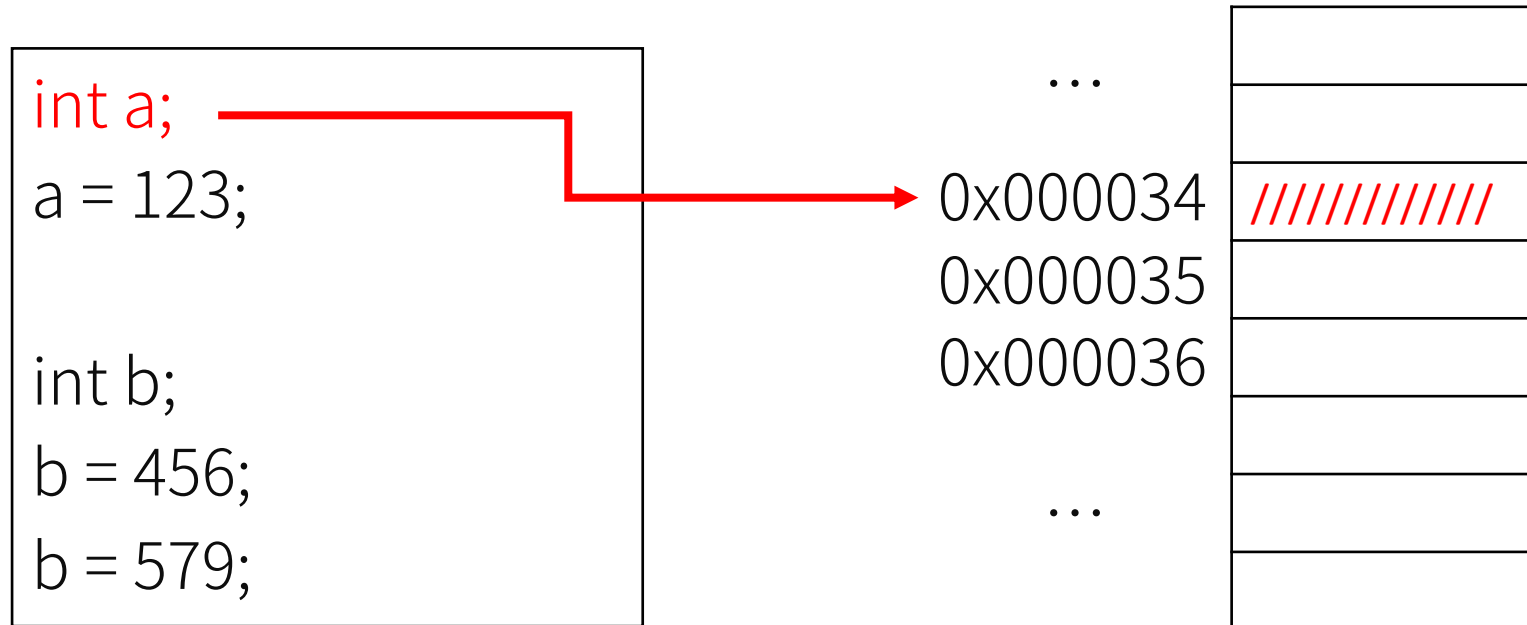


```
int a;  
a = 123;  
  
int b;  
b = 456;  
b = 579;
```

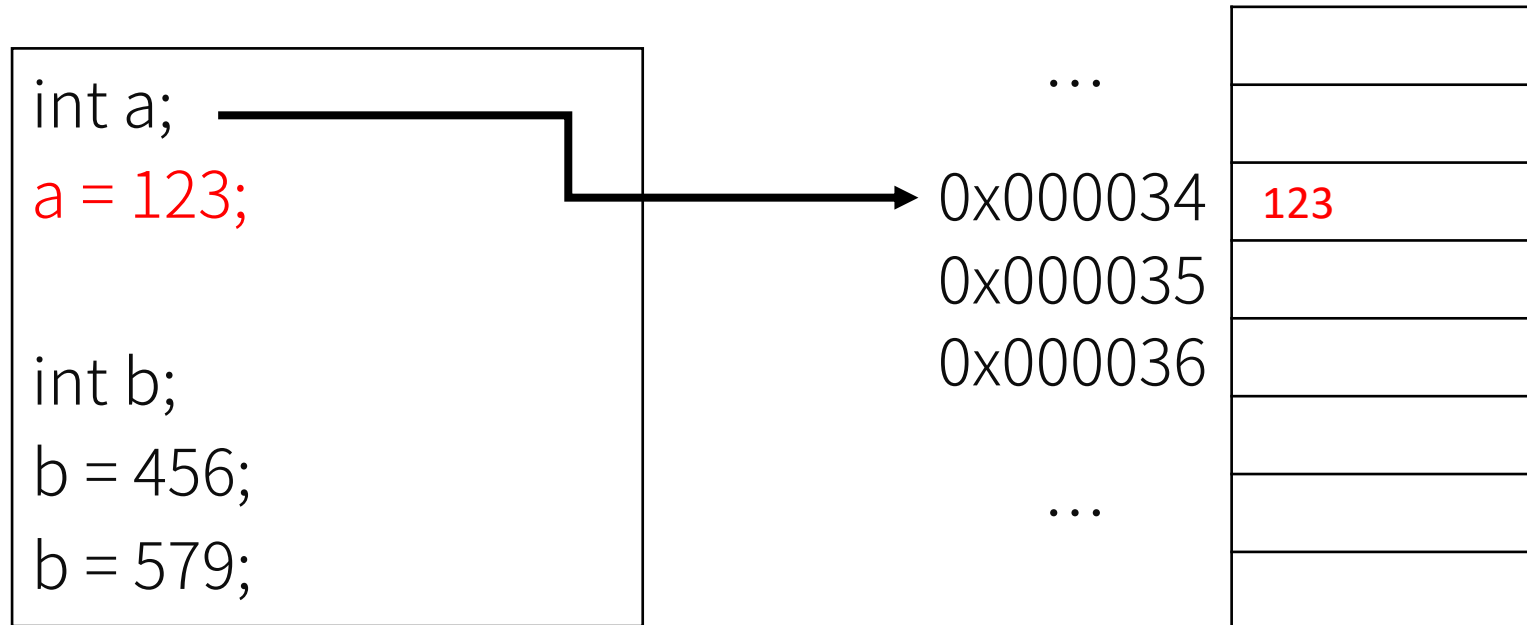


由上而下、由左而右、由內而外

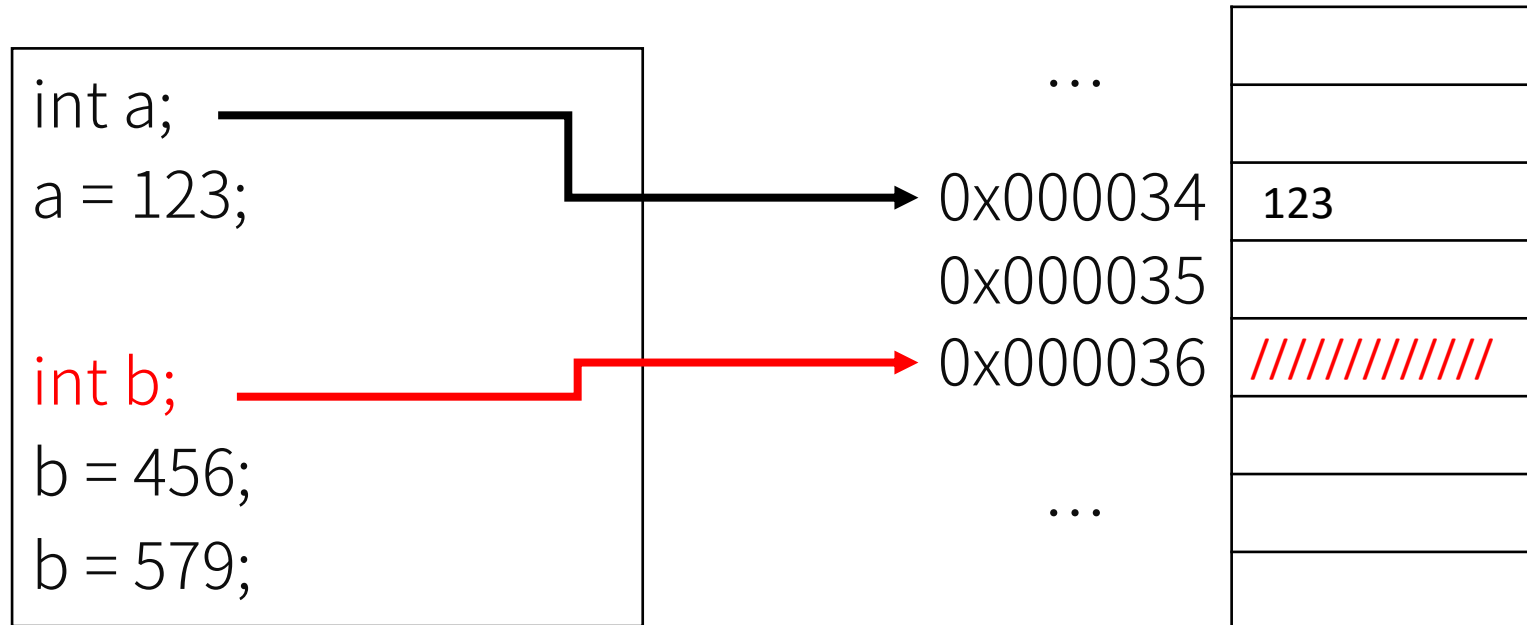
變數會存在電腦的記憶體裡



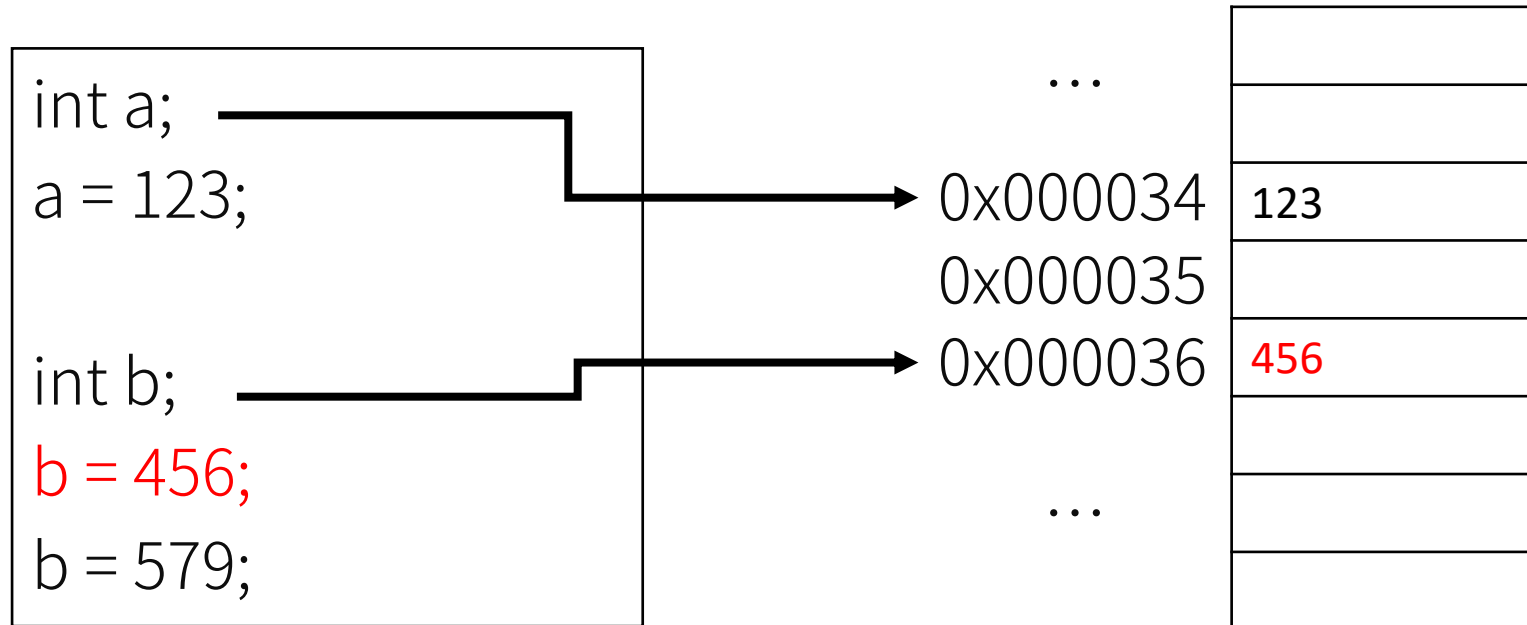
變數會存在電腦的記憶體裡



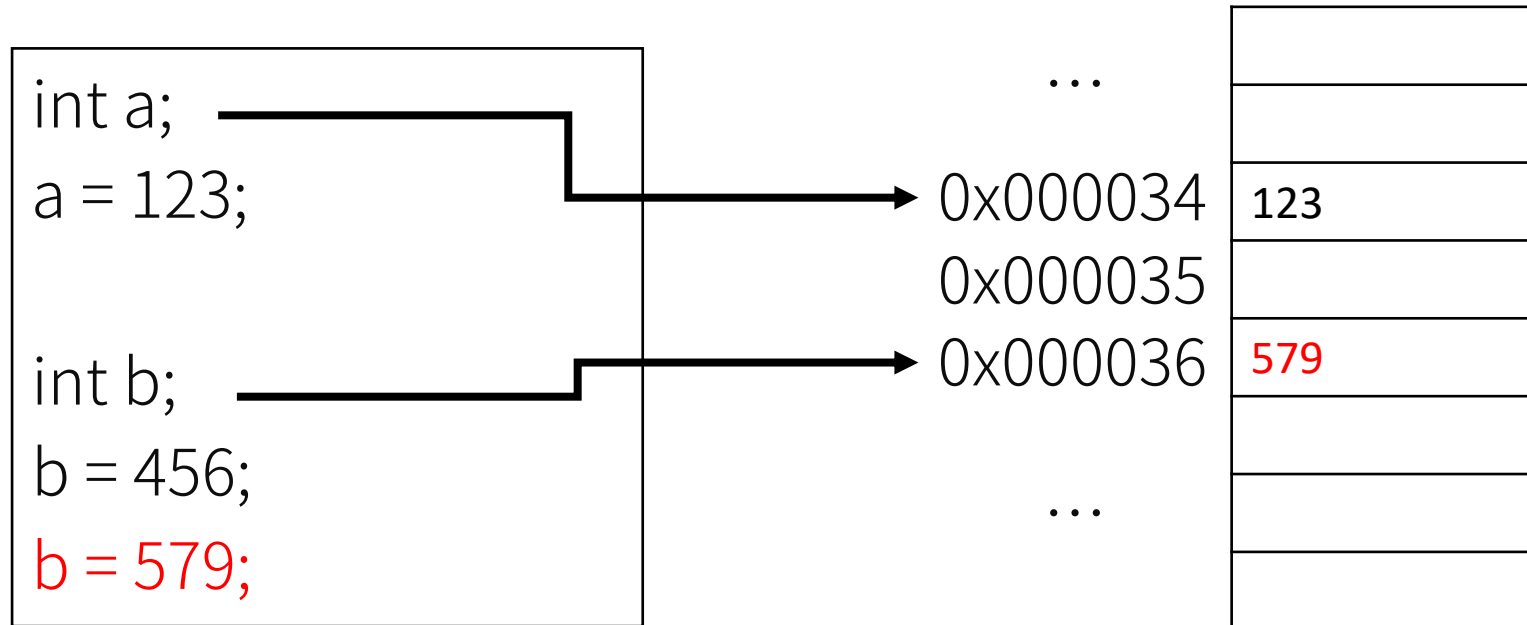
變數會存在電腦的記憶體裡



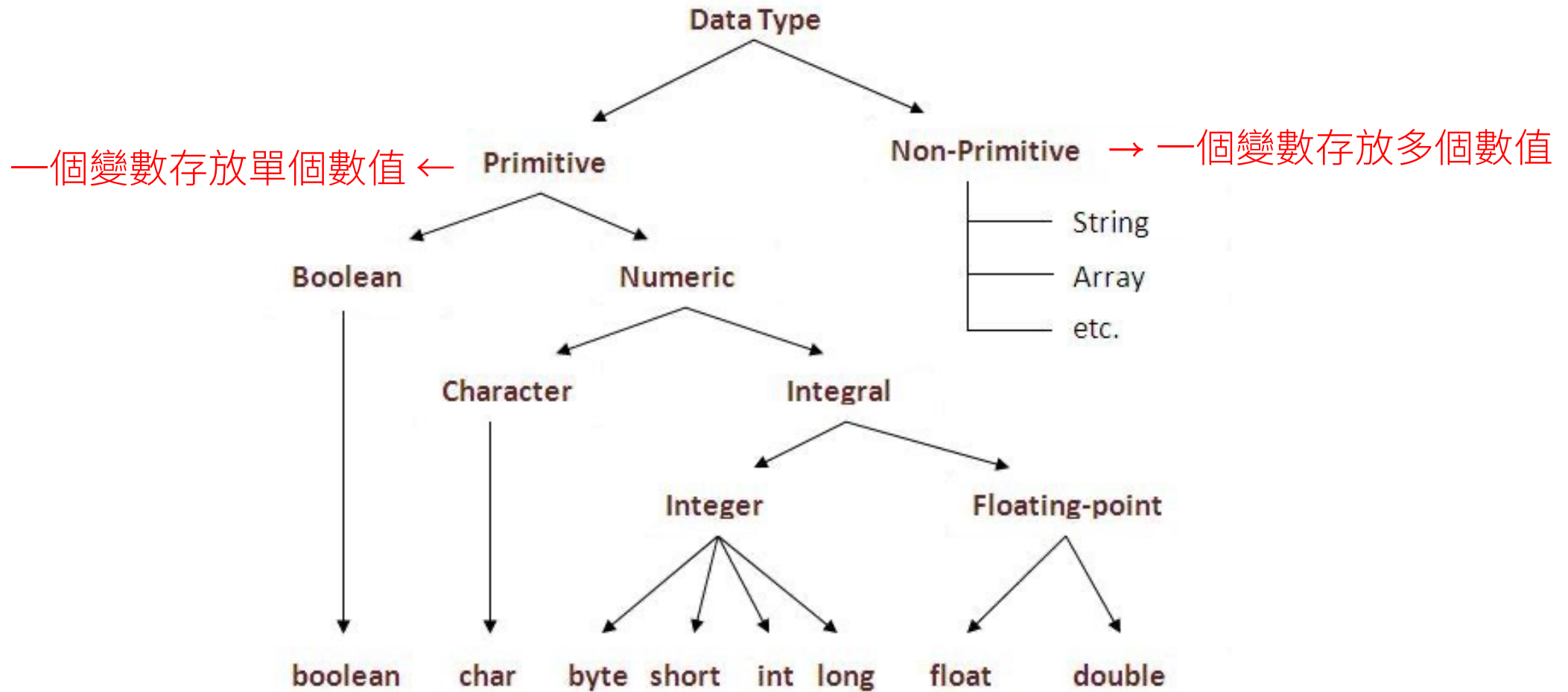
變數會存在電腦的記憶體裡



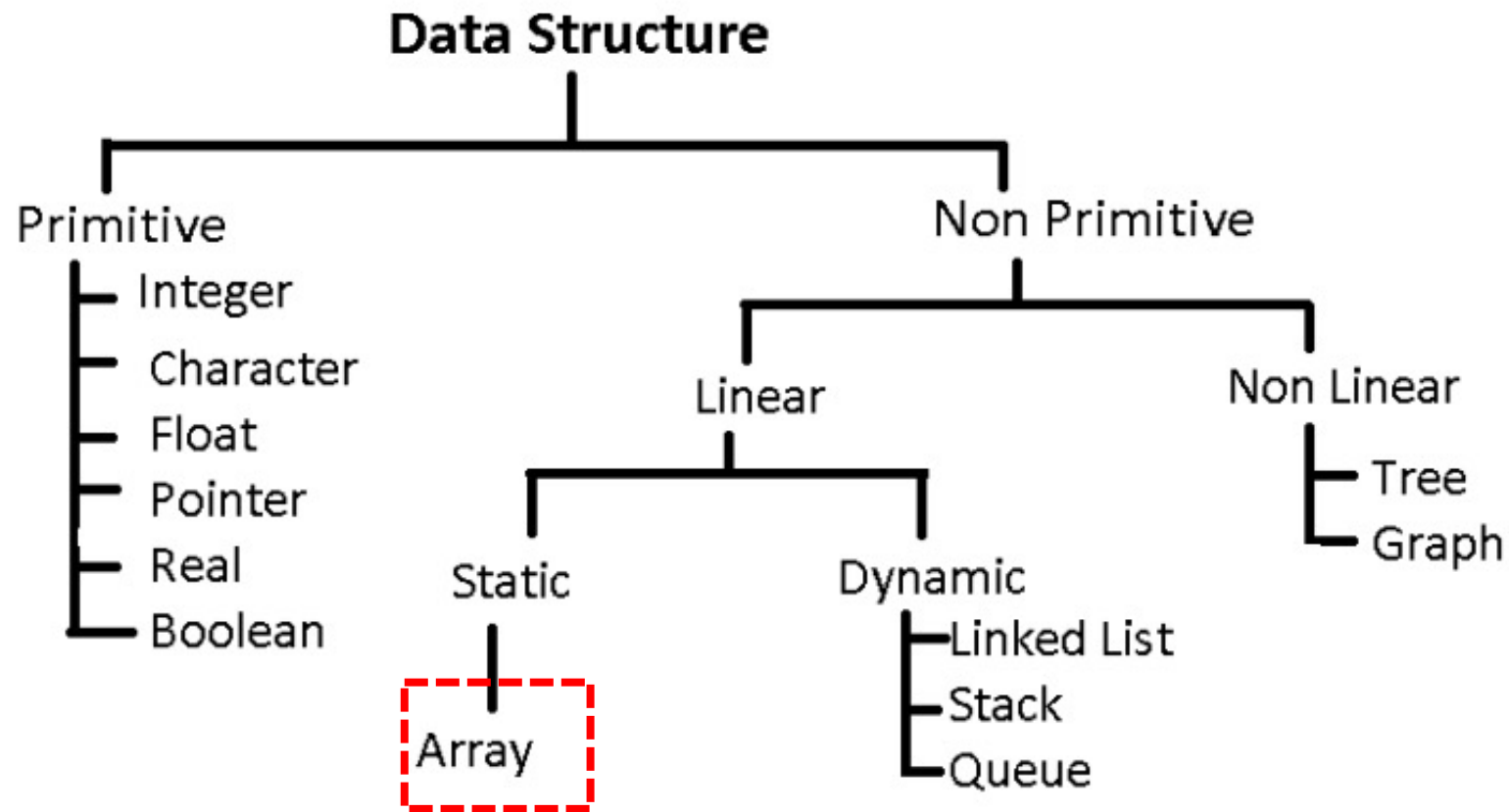
變數會存在電腦的記憶體裡



不同的資料型態

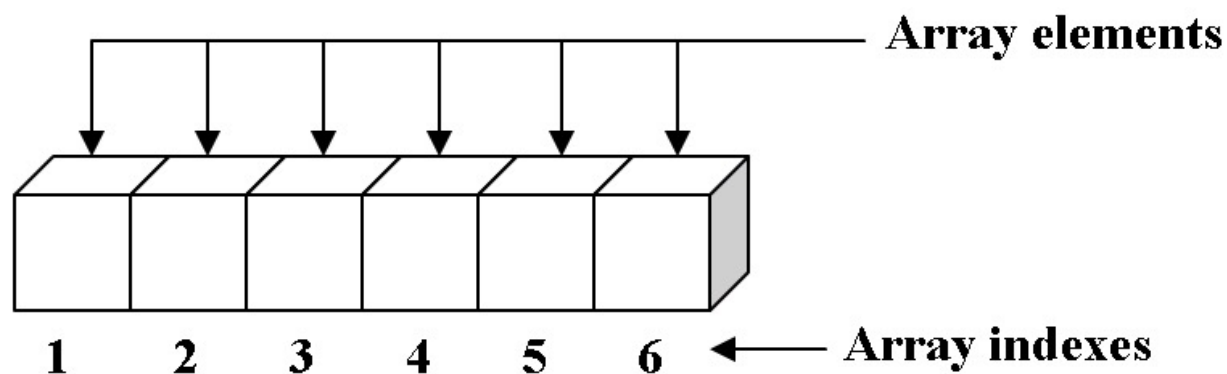


陣列是一種靜態且線性的資料結構



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

→ 陣列的長度要預先定義

int a;



a

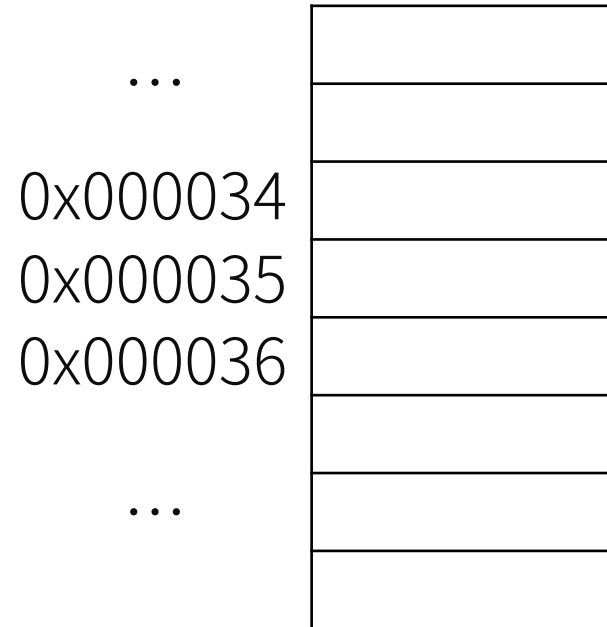
int a[5];



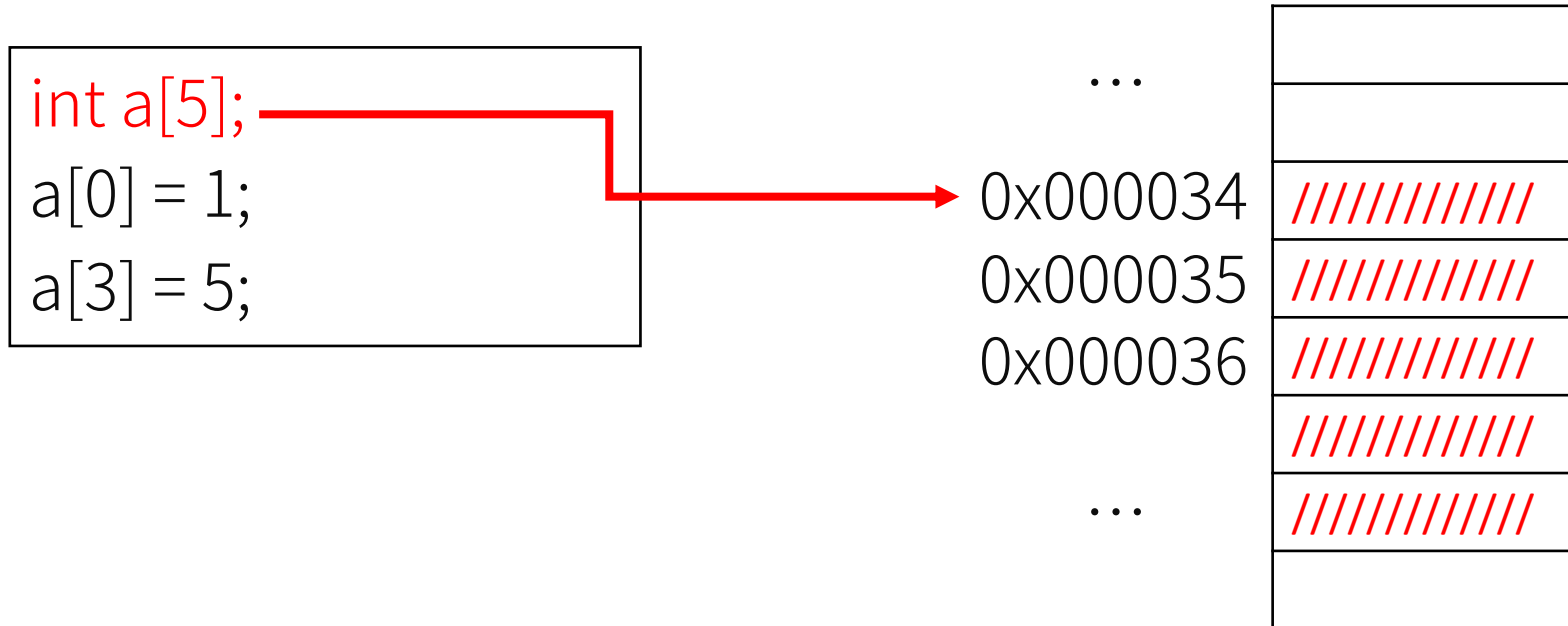
a[0] a[1] a[2] a[3] a[4]

陣列會佔用連續的記憶體空間

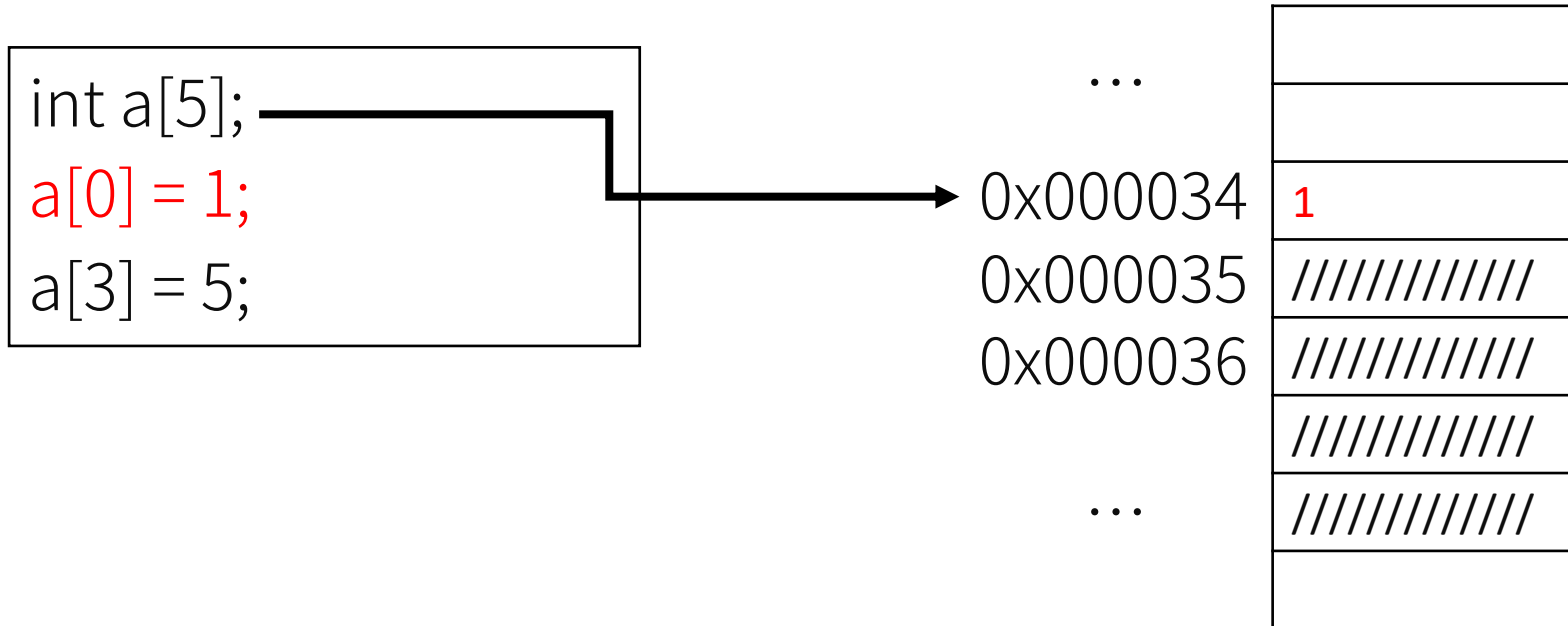
```
int a[5];  
a[0] = 1;  
a[3] = 5;
```



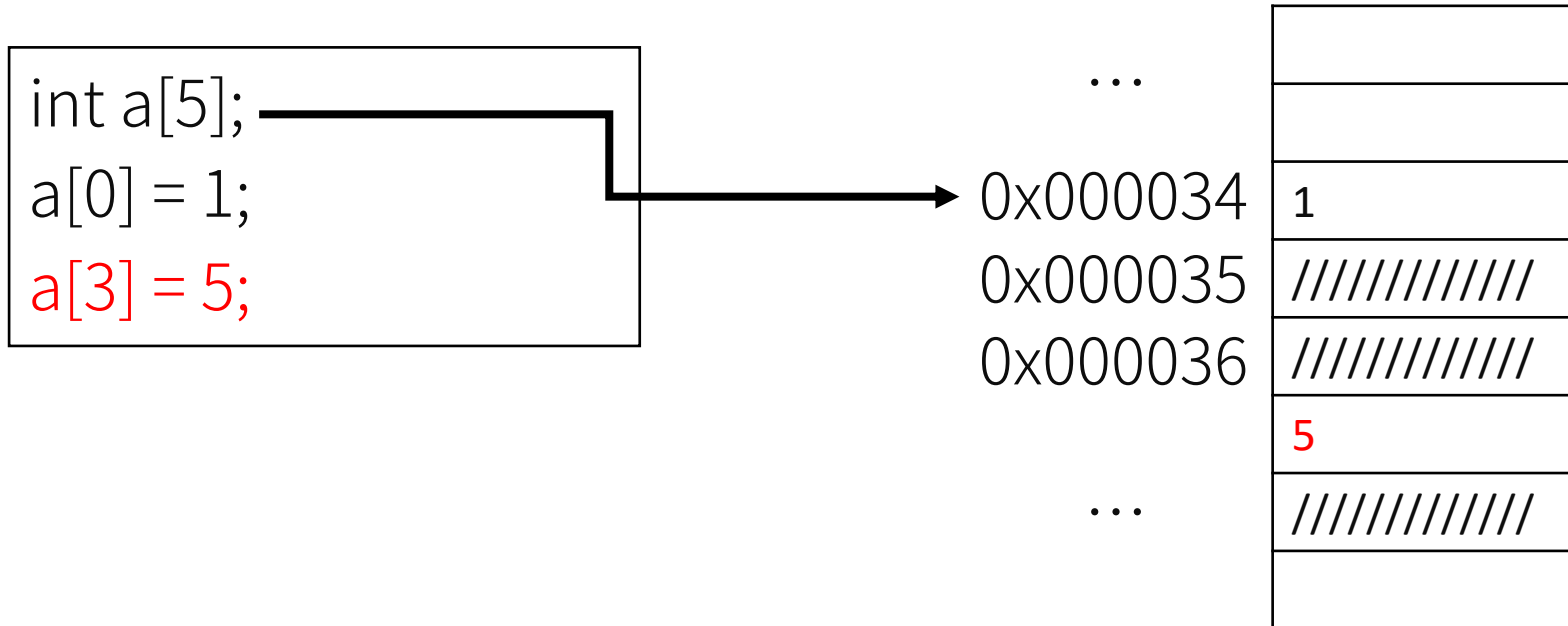
陣列會佔用連續的記憶體空間



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



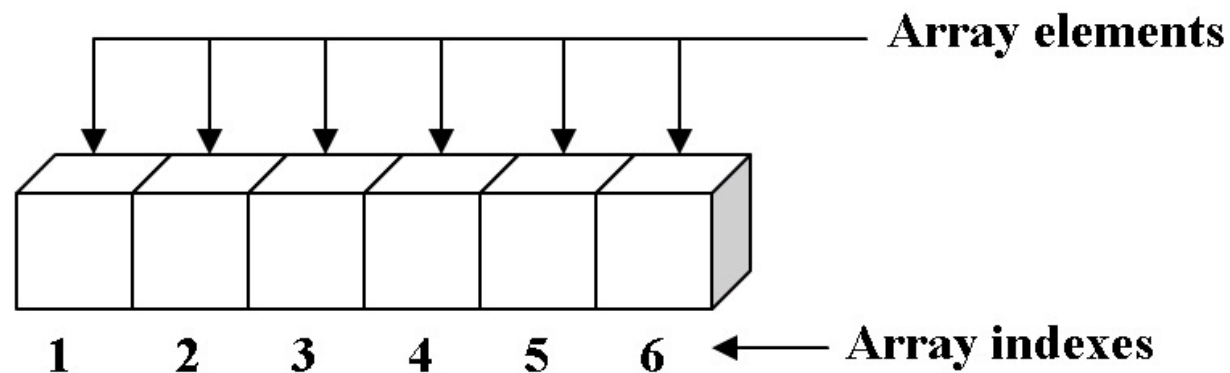
陣列是由連續的位置所組成的容器

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

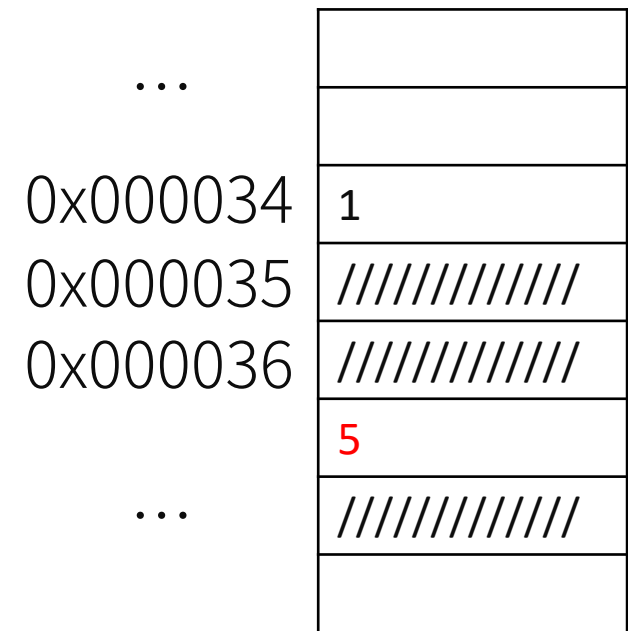
■以 $\{\dots\}$ 語法來表示宣告陣列，以 $[\dots]$ 語法取值或賦值 (Setter & Getter)

```
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

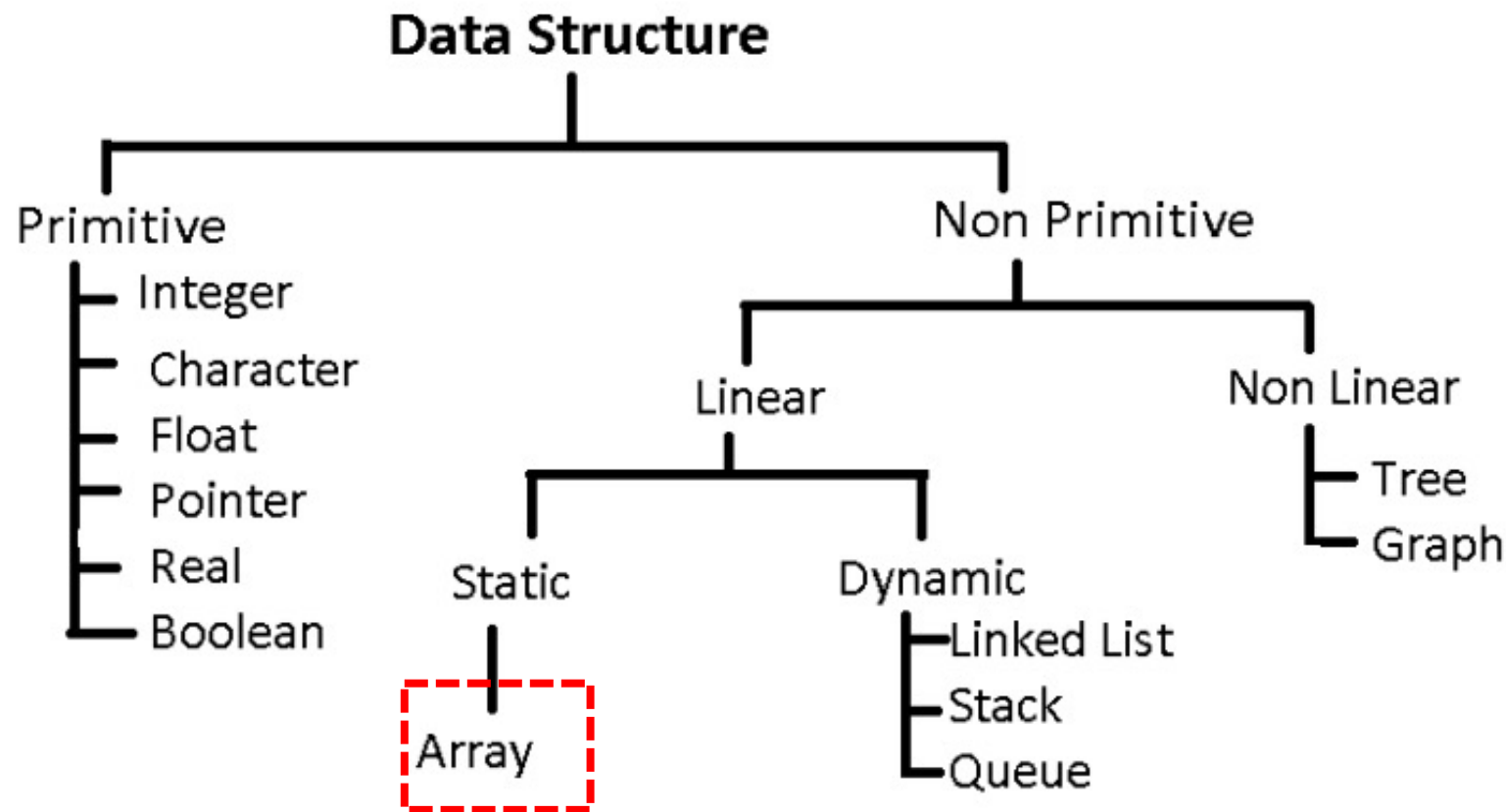


→ 使用的時候是一個容器，本質上是存在連續個記憶體位置上

陣列是一種 靜態 且 線性 的資料結構

→ 大小固定

→ 連續的



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]); → 手動算出陣列大小的方法
11
12    cout << "a = " << a << endl; → 陣列變數代表在記憶體中開始的位置
13    cout << "a[0] = " << a[0] << endl;
14    cout << "a[MAXN-1] = " << a[MAXN-1] << endl; → 陣列的索引會從 0 ~ n-1
15    cout << "a[n-1] = " << a[n-1] << endl;          元素有 n 個
16    return 0;
17 }
```

寫入的時候同時也要記錄位置

```
1 #include<iostream>
2 #define MAXN 10 → MAXN 表示陣列佔用的大小
3 using namespace std;
4
5 int a[MAXN];
6 int main(){
7
8     int x, n = 0; → n 代表有填上數值的個數
9     while(cin >> x){
10         a[n++] = x; → 一個一個填入輸入到陣列中
11     }
12     for(int i = 0; i < n; i++){
13         cout << i << " => " << a[i] << endl;
14     }
15     return 0;
16 }
```

→ 準備一個「比較大的」陣列，讓使用者輸入加入到陣列中

一維陣列與二維陣列

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

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

```
int a[4][5] = {  
    {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]

一維陣列

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

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

二維陣列當成方陣

```
1 #include<iostream>
2 using namespace std;
3 int a[4][4] = {
4     {10, 11, 12, 13},
5     {14, 15, 16, 17},
6     {18, 19, 20, 21},
7     {22, 23, 24, 25}
8 };
9 int main(){
10     int n = sizeof(a)/sizeof(a[0]);
11     int m = sizeof(a[0])/sizeof(a[0][0]);
12     for(int i = 0; i < n; i++){
13         for(int j = 0; j < m; j++){
14             cout << a[i][j] << "\t";
15         }
16         cout << endl;
17     }
18     return 0;
19 }
```

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

寫入的時候同時也要記錄位置

```
1 #include<iostream>
2 #define MAXN 10 → MAXN 表示陣列佔用的大小
3 using namespace std;
4
5 int a[MAXN];
6 int main(){
7
8     int x, n = 0; → n 代表有填上數值的個數
9     while(cin >> x){
10         a[n++] = x; → 一個一個填入輸入到陣列中
11     }
12     for(int i = 0; i < n; i++){
13         cout << i << " => " << a[i] << endl;
14     }
15     return 0;
16 }
```

→ 準備一個「比較大的」？陣列，讓使用者輸入加入到陣列中

動態宣告陣列

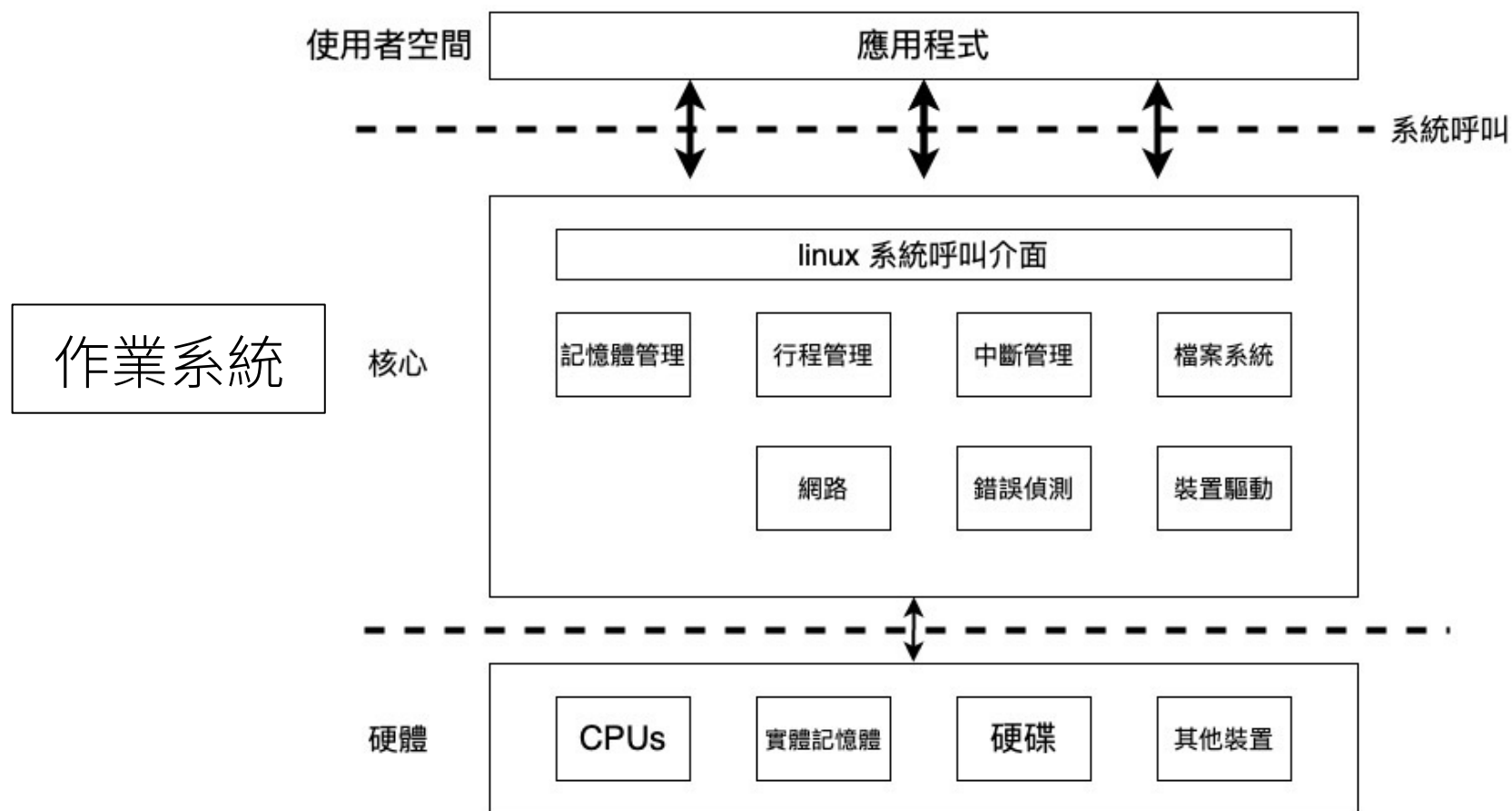
當無法預知陣列的大小時，而是由使用者輸入後才做決定的話。這時候可以使用動態宣告陣列的技巧，根據程式運行的過程才向記憶體要空間。

```
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     }
```

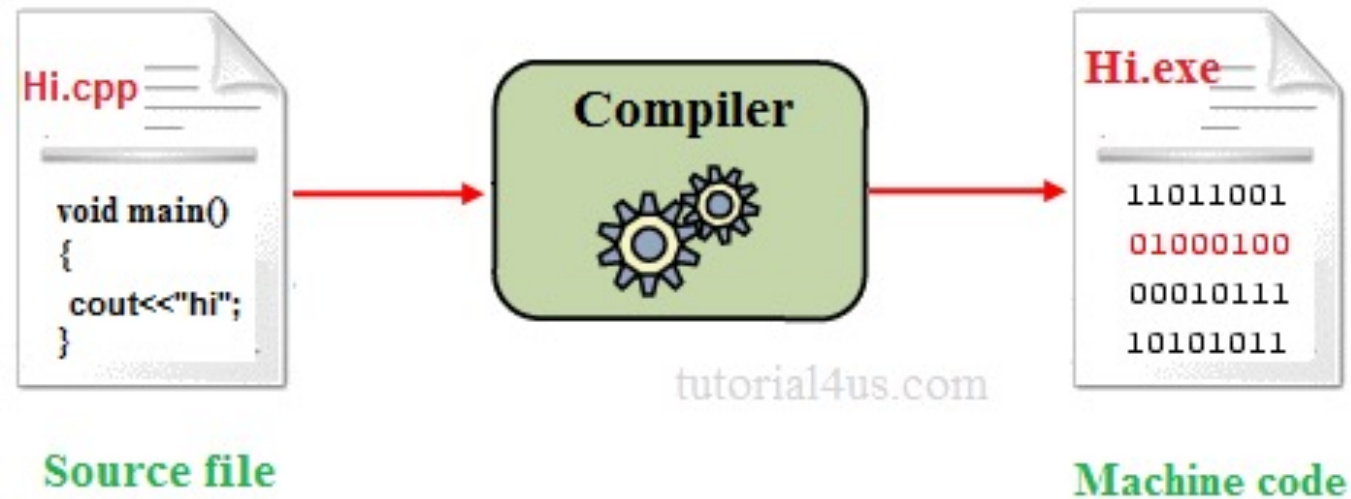
```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int n;
6     cin >> n;
7     int *a = new int[n];
8     // int *a;
9     // a = new int[n];
10
11     for(int i = 0; i < n; i++){
12         cout << i << " => " << a[i] << endl;
13     }
14 }
```

變數會存在電腦的記憶體裡

→ 宣告變數的當下，就會向記憶體索取一段空間



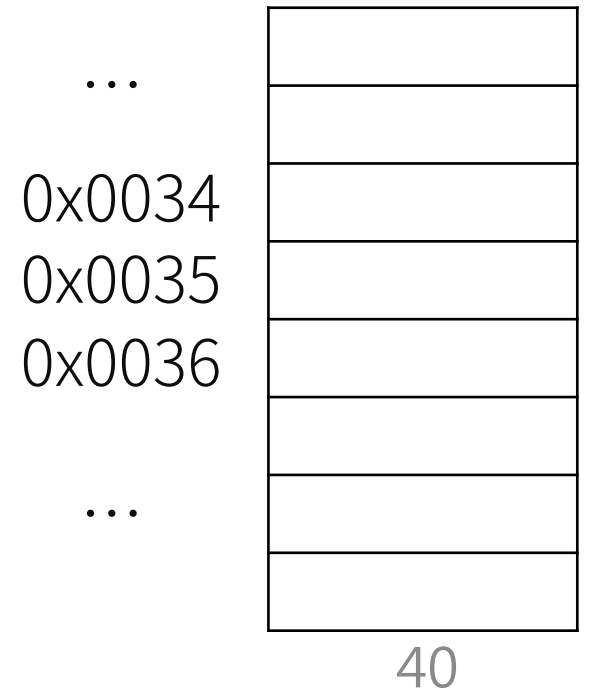
C++ 是一種編譯式的程式語言



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。

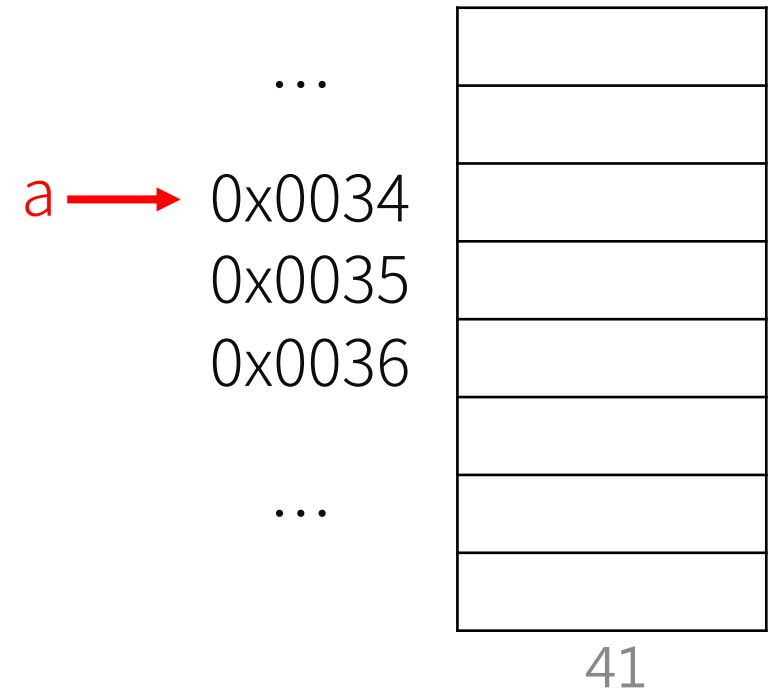
```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int n;
6     cin >> n;
7     int *a;
8     a = new int[n];
9
10    for(int i = 0; i < n; i++){
11        cout << i << " => " << a[i] << endl;
12    }
13 }
```



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。

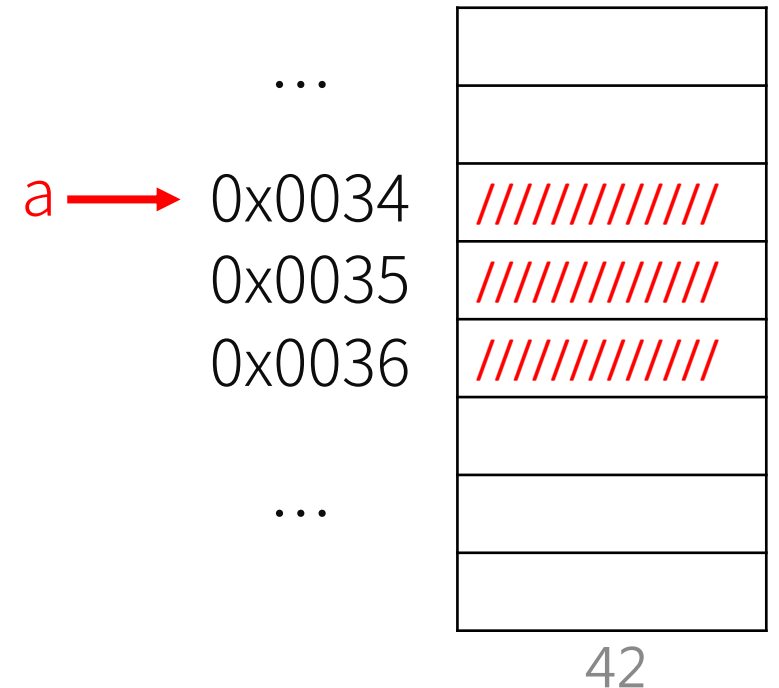
```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int n;
6     cin >> n;
7     int *a; → 宣告 a 指標，指向一個記憶體位置
8     a = new int[n];
9
10    for(int i = 0; i < n; i++){
11        cout << i << " => " << a[i] << endl;
12    }
13 }
```



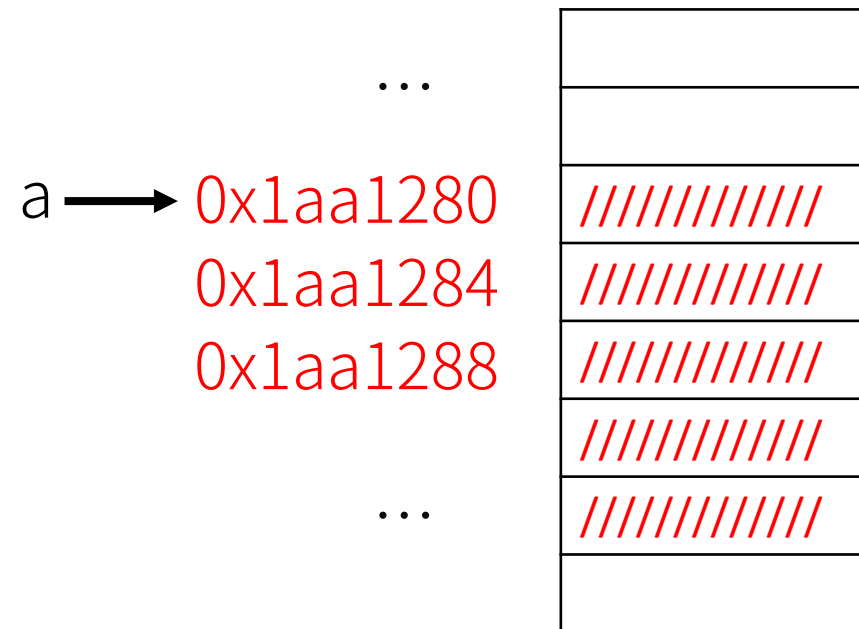
動態宣告陣列

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。

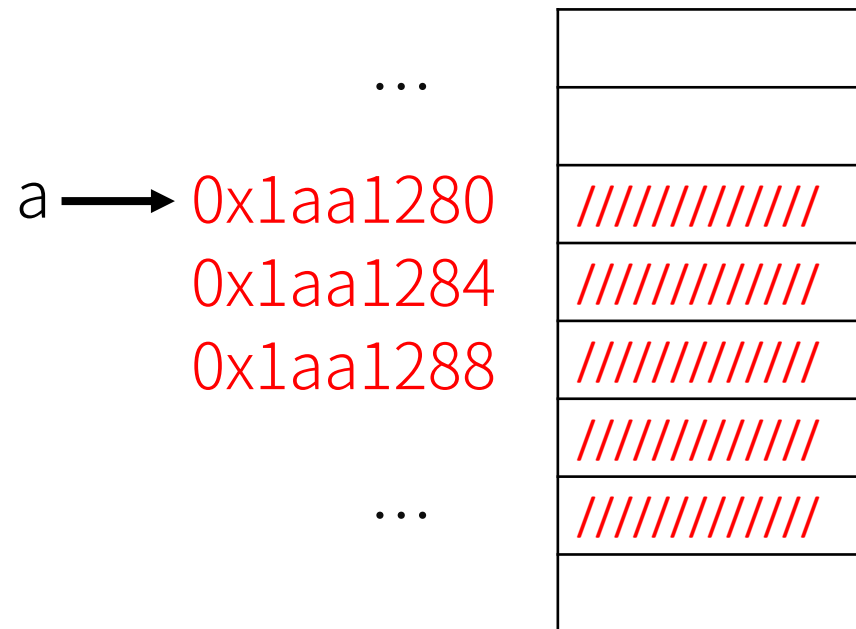
```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int n;
6     cin >> n;
7     int *a;
8     a = new int[n]; → 從 a 位置上，新增 n 個大小
9
10    for(int i = 0; i < n; i++){
11        cout << i << " => " << a[i] << endl;
12    }
13 }
```



動態宣告陣列實際宣告出來的大小

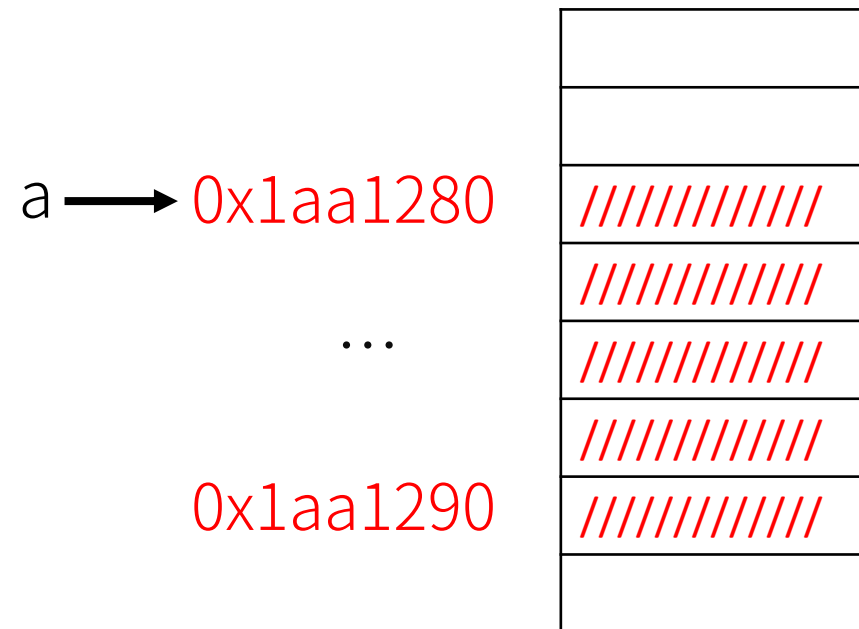


動態宣告陣列實際宣告出來的大小

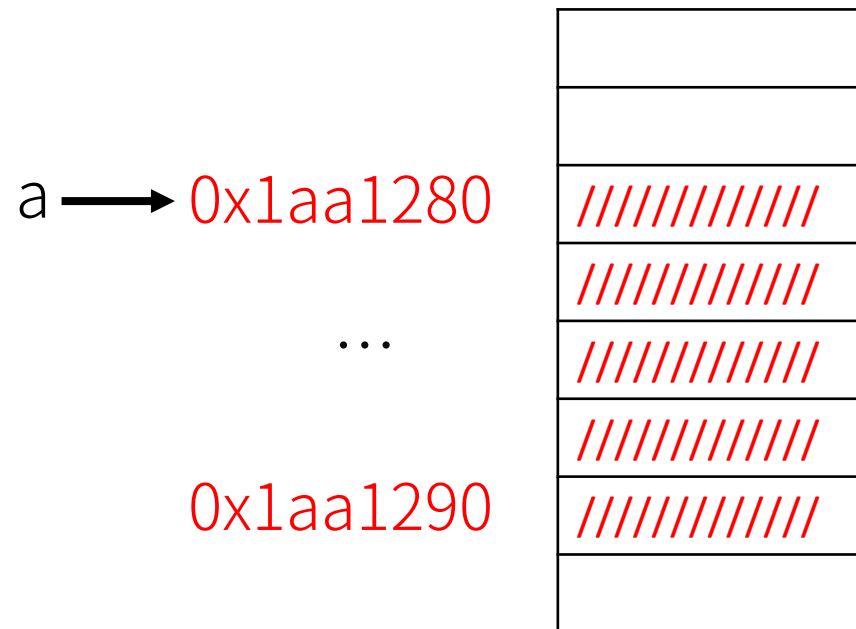


→ 記憶體一格的大小差距為 4（因為佔用 4 Bytes）

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



→ 記憶體位址會採用 16 進位的計算，五格的差距是 16

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    }
12    cout << endl;
13 }
```

釋放動態記憶體配置空間

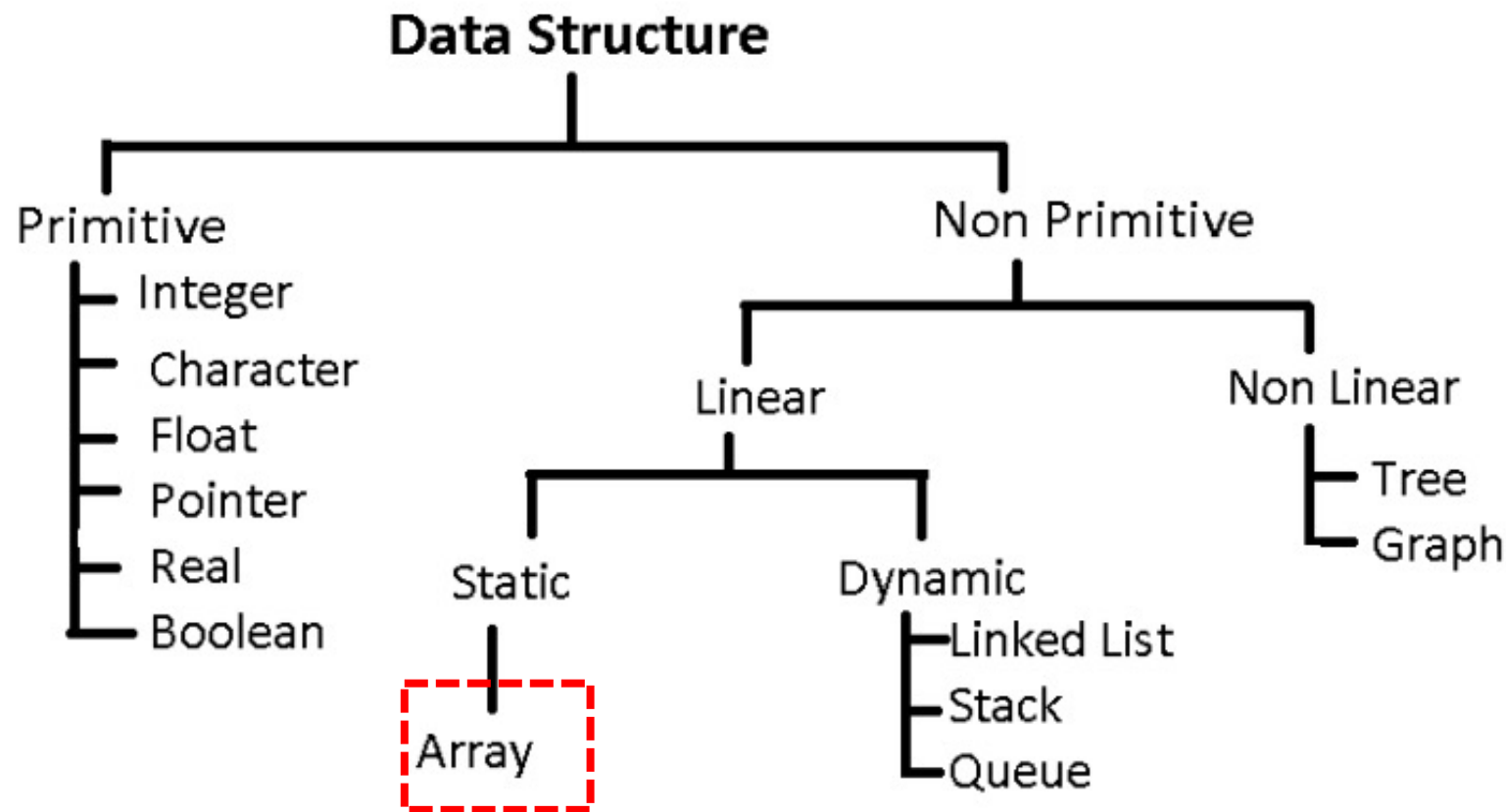
```
1 int *ptr;  
2 ptr = new int[10];  
3  
4 // 釋放動態記憶體配置空間  
5 delete [] ptr;
```

```
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
```

陣列是一種 靜態 且 線性 的資料結構

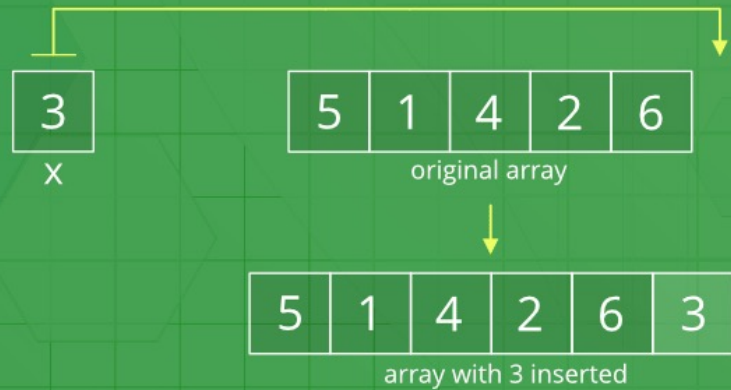
→ 大小固定

→ 連續的

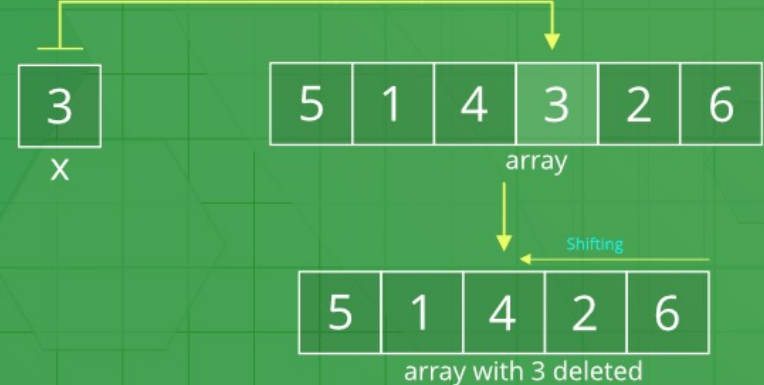


Array 佔用連續性的記憶體空間

Insert Operation in Unsorted Array



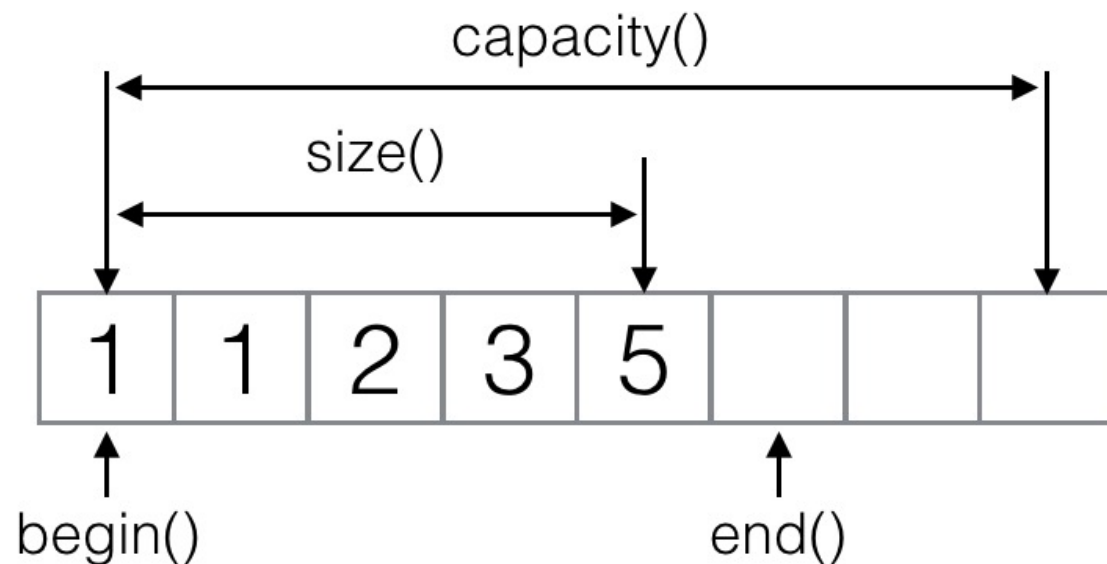
Delete Operation in Unsorted Array



→ 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。



```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     // initialize
6     int a[4] = {1, 2, 3};
7
8     // push
9     a[3] = 4;
10
11    // insert
12    a[3] = a[2];
13    a[2] = a[1];
14    a[1] = 999;
15
16    int n = sizeof(a) / sizeof(a[0]);
17    for(int i = 0; i < n; i++)
18        cout << a[i] << endl;
19 }

```

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     // initialize
7     vector<int> v{ 1, 2, 3 };
8
9     // push
10    v.push_back(4);
11
12    // insert
13    v.insert(v.begin()+1, 999);
14
15    for(int i = 0; i < v.size(); i++)
16        cout << v[i] << endl;
17 }
18
19

```



```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     // initialize
6     int a[4] = {1, 2, 3};
7
8     // push
9     a[3] = 4;
10
11    // insert
12    a[3] = a[2];
13    a[2] = a[1];
14    a[1] = 999;
15
16    int n = sizeof(a) / sizeof(a[0]);
17    for(int i = 0; i < n; i++)
18        cout << a[i] << endl;
19 }

```

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     // initialize
7     vector<int> v{ 1, 2, 3 };
8
9     // push
10    v.push_back(4);
11
12    // insert
13    v.insert(v.begin()+1, 999);
14
15    for(int i = 0; i < v.size(); i++)
16        cout << v[i] << endl;
17 }
18
19

```

```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     // initialize
6     int a[4] = {1, 2, 3};
7
8     // push
9     a[3] = 4;
10
11    // insert
12    a[3] = a[2];
13    a[2] = a[1];
14    a[1] = 999;
15
16    int n = sizeof(a) / sizeof(a[0]);
17    for(int i = 0; i < n; i++)
18        cout << a[i] << endl;
19 }

```

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     // initialize
7     vector<int> v{ 1, 2, 3 };
8
9     // push
10    v.push_back(4);
11
12    // insert
13    v.insert(v.begin()+1, 999);
14
15    for(int i = 0; i < v.size(); i++)
16        cout << v[i] << endl;
17 }
18
19

```

```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     // initialize
6     int a[4] = {1, 2, 3};
7
8     // push
9     a[3] = 4;
10
11     // insert
12     a[3] = a[2];
13     a[2] = a[1];
14     a[1] = 999;
15
16     int n = sizeof(a) / sizeof(a[0]);
17     for(int i = 0; i < n; i++)
18         cout << a[i] << endl;
19 }

```

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     // initialize
7     vector<int> v{ 1, 2, 3 };
8
9     // push
10    v.push_back(4);
11
12    // insert
13    v.insert(v.begin()+1, 999);
14
15    for(int i = 0; i < v.size(); i++)
16        cout << v[i] << endl;
17 }
18
19

```

```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     // initialize
6     int a[4] = {1, 2, 3};
7
8     // push
9     a[3] = 4;
10
11    // insert
12    a[3] = a[2];
13    a[2] = a[1];
14    a[1] = 999;
15
16    int n = sizeof(a) / sizeof(a[0]);
17    for(int i = 0; i < n; i++)
18        cout << a[i] << endl;
19 }

```

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
6     // initialize
7     vector<int> v{ 1, 2, 3 };
8
9     // push
10    v.push_back(4);
11
12    // insert
13    v.insert(v.begin()+1, 999);
14
15    int n = v.size();
16    for(int i = 0; i < n; i++)
17        cout << v[i] << endl;
18 }
19

```



#思考一下

如果我們想要紀錄每一個的數字分別
被輸入幾次的話，可以怎麼做？

陣列與物件的使用情境

index:	0	1	2	3	4	5	6	7	8	9	
{	0	0	0	0	0	0	0	0	0	0	}

→ 把 Index 當成出現數字，Value 當成出現次數



#思考一下

那如果輸入的數值是字元怎麼辦？

陣列索引一定是連續的數字

index:	0	1	2	3	4	5	6	7	8	9
	0	0	0	0	0	0	0	0	0	0

→ 把 Index 當成出現數字，Value 當成出現次數

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

```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int A[200] = {0};
6     A[int('A')] = 999;
7     A[int("z"[0])] = 888;
8
9     for(int i = 65; i <= 122; i++){
10         if(A[i] > 0){
11             cout << char(i) << " => ";
12             cout << A[i] << endl;
13         }
14     }
15 }

```

```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int A[100] = {0};
6     A[int('A')-65] = 999;
7     A[int("z"[0])-65] = 888;
8
9     for(int i = 0; i <= 57; i++){
10         if(A[i] > 0){
11             cout << char(i+65) << " => ";
12             cout << A[i] << endl;
13         }
14     }
15 }

```

→ 可能會多餘的索引沒有被使用到的空間浪費

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。第二個稱為該关键字的值 (value)。

1120217	Nikhilesh
1120236	Navneet
1120250	Vikas
1120255	Doodrah

Keys

values

Map

key:	A	B	...	Y	Z	...	a	b	...	z	
{	0	0	0	0	0	0	0	0	0	0	}

→ Map 可以自定義 key-value 的組合

```

1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int A[100] = {0};
6     A[int('A')-65] = 999;
7     A[int("z"[0])-65] = 888;
8
9     for(int i = 0; i <= 57; i++){
10         if(A[i] > 0){
11             cout << char(i+65) << " => ";
12             cout << A[i] << endl;
13         }
14     }
15 }

```

```

1
2 #include <iostream>
3 #include <map>
4 using namespace std;
5
6 int main() {
7     map<char, int> m;
8     m['A'] = 999;
9     m['z'] = 888;
10
11     map<char,int>::iterator it;
12     for(it = m.begin(); it != m.end();
13         ++it){
14         cout << it->first << " => ";
15         cout << it->second << endl;
16     }
17 }
18

```

結構 (Struct)

C/C++ 而結構 (Struct) 提供在一個變數內自定義儲存的資料屬性。

```
1  
2  
3 struct Book {  
4     string title;  
5     int price;  
6 };  
7  
8  
9
```

```
1 int main( ){  
2  
3     Book book1;  
4  
5     book1.title = "C++ How to Program";  
6     book1.price = 636;  
7  
8     cout << book1.title << endl;  
9     cout << book1.price << endl;  
10    return 0;  
11  
12 }
```

→ Struct 是一種可以自定義格式型態

結構 (Struct)

C/C++ 而結構 (Struct) 提供在一個變數內自定義儲存的資料屬性。

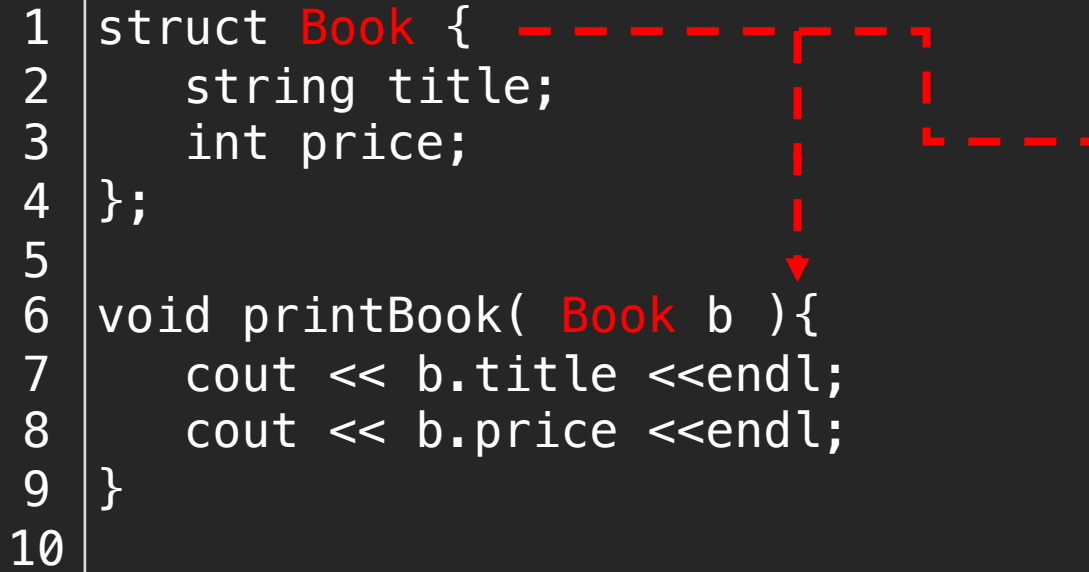
```
1  
2  
3 struct Book {  
4     string title;  
5     int price;  
6 };  
7  
8  
9
```

```
1 int main( ){  
2  
3     Book book1;  
4  
5     book1.title = "C++ How to Program";  
6     book1.price = 636;  
7  
8     cout << book1.title << endl;  
9     cout << book1.price << endl;  
10    return 0;  
11  
12 }
```

→ 定義了一個叫做「Book」的變數型態

Struct 就像是自定義的「變數」型態

```
1 struct Book {  
2     string title;  
3     int price;  
4 };  
5  
6 void printBook( Book b ){  
7     cout << b.title << endl;  
8     cout << b.price << endl;  
9 }  
10
```



```
1 int main( ){  
2  
3     Book book1;  
4  
5     book1.title = "C++ How to Program";  
6     book1.price = 636;  
7  
8     printBook( book1 );  
9     return 0;  
10 }
```

→ 宣告的物件實體也可以被當成參數傳入 function

成員函式 (member function)

```
1 struct Book {  
2  
3     string title;  
4     int price;  
5     void f(){  
6         cout << "I'm a book";  
7     }  
8  
9 };  
10
```

```
1  
2  
3 int main( ){  
4  
5     Book book1;  
6     book1.f();  
7  
8     return 0;  
9 }  
10
```

→ Struct 中也能夠定義函式，稱為成員函式

成員函式 (member function)

```
1 struct Book {  
2  
3     string title;  
4     int price; ← 成員變數  
5     void f(); ← 成員函式  
6         cout << "I'm a book";  
7     }  
8  
9 };  
10
```

```
1  
2  
3 int main( ){  
4  
5     Book book1;  
6     book1.f();  
7  
8     return 0;  
9 }  
10
```

Struct 另一種初始化方法

```
1
2 int main( ){
3
4     Book book1;
5     book1.title = "C++ How to Program";
6     book1.price = 636;
7
8     return 0;
9 }
10
```

```
1
2 int main( ){
3
4     Book book1 = {
5         .title = "C++ How to Program",
6         .price = 656
7     };
8
9     return 0;
10 }
```

Struct 就像是自定義的「變數」型態

```
1  
2  
3 struct Book {  
4     int id;  
5     void f(int i){  
6         cout << "I'm a book #";  
7     }  
8 };  
9  
10
```

```
1  
2 int main( ){  
3     Book books[10];  
4     for(int i = 0; i < 10; i++){  
5         books[i].id = i;  
6         books[i].f(books[i].id);  
7     }  
8     return 0;  
9 }  
10
```

→ 宣告的實體也可以被當成元素組成一個陣列

Struct 就像是自定義的「變數」型態

```
1
2
3 struct Book {
4     int id;
5     void f(int i){
6         cout << "I'm a book #";
7     }
8 };
9
10
```

```
1 int main( ){
2
3     Book books[10];
4     for(int i = 0; i < 10; i++){
5         books[i].id = i;
6         books[i].f(books[i].id);
7     }
8     return 0;
9 }
10
```

→ 用 [...] 取得元素，用 . 取得成員

Thanks for listening.

元智大學 | C++ 程式設計實習

Wei-Yuan Chang