

# IM1003: Programming Design, Spring 2017

## Lab 04

Jhih-Bang Hsieh  
bigelephant29

**National Taiwan University**

# Outline

- HW2 Code Review
- Array
- Function
- Practice

## HW2-3

```

1  #include <iostream>
2  using namespace std;
3  int main ()
4  {
5      int num = 0, k = 0, a = 0 ;
6      cin >> num >> k ;
7      if (k == 1)
8          { //盡量不要過度縮排，一個 block 一個縮排。
9              for (int i = 1 ; i <= num ; i++ )
10                 {
11                     if (num % i == 0 )
12                         cout << (a == 0 ? "" : " ") << i ;
13                     a = 1 ;
14                 }
15             }
16      if (k == 2)
17          {
18              // 省略
19          }
20      return 0 ;
21  }

```

## HW2-3

```
1  for(t; t <= n; t++)
2  {
3
4  }
5
6  for(; t <= n; t++)
7  {
8
9  }
10
11 for(;;)
12 {
13     //infinity loop
14 }
```

## HW2-3

```
1  if(number % i == 0) // 建議加上大括號，增加可讀性。
2      for(int j = 2; j<=i; j++)
3      {
4          if(i % j == 0 && i != j)
5              break;
6
7          if(i == j)
8          {
9              if(k != 1)
10                 cout << " ";
11                 cout << i;
12                 k++;
13             }
14     }
```

## HW2-4

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {    //請注意縮排對齊
5      int num = 0, sum = 0;
6      while(cin >> num)
7      {
8          //縮排距離請保持一致
9          for(int i=2; i <= num; i++)
10         {
11             while(num % i == 0)
12             {
13                 if(num % (i * i)==0)
14                 {
15                     //中間省略
16                 }
17             }
18         }
19         cout << "\n";
20     } //這行與下一行應該要對齊
21     return 0;
22 }
23
24 }
```

# Array

- 陣列分成靜態陣列、動態陣列。
- 前者在編譯時期決定大小，存在 stack 裡面。
- 後者在執行時期決定大小，存在 heap 裡面。

# Static & Dynamic Array

- 靜態陣列速度很快。
- 動態陣列大小彈性，控制記憶體用量較容易。
- 非全域靜態陣列受 stack size 限制。



# Declare a static array

```
1  int my_array[256];
```

- 這樣宣告了一個名字叫做 `my_array` 的靜態陣列，長度為 256。
- index: `my_array[0]` - `my_array[255]`
- 大小無法改變，宣告時無法用變數指定大小。

# Declare a dynamic array

```
1  int n = 256;  
2  int *my_array = new int[n];  
3  
4  delete [] my_array;
```

- 這樣宣告了一個名字叫做 `my_array` 的指標。
- 該指標指向一段長度為 256 的動態陣列。
- index: `my_array[0]` - `my_array[255]`
- 大小無法改變，但是宣告時可以用變數指定大小。
- 陣列使用完畢之後，須自行指定釋放記憶體（delete）。

# Notice

```
1  int arr[] = {}; // wrong
2
3  int n;
4  cin >> n;
5  int arr[n]; // wrong
```

- 兩種寫法都是錯的。
- 靜態陣列須在編譯時期決定大小，所以不能讓使用者輸入時決定（執行時期）。

# Notice

```
1  #include<iostream>
2  using namespace std;
3  int main() {
4      int arr[100000000];
5      for(int i = 0 ; i < 100000000 ; i++)
6          arr[i] = i;
7      cout << arr[99999999] << endl;
8      return 0;
9  }
```

- 非全域的靜態陣列存在 stack 裡面，受到 stack size 限制。
- stack size 通常不大，所以開一個很大的靜態陣列在 function 內是很危險的。
- 將這段程式碼抓到電腦上跑跑看，應該會出問題才對！
- 可以在編譯參數中將 stack 調大避免這個問題。
- 用動態陣列才是最好的方法！

# Function