C++基礎語法 Unit-7

- function 函數
- 內建函數 vs. 自訂函數
- 全域變數 vs. 區域變數

常用內建函數

Don't reinvent the wheel!

- min(x, y)
- max(x, y)
- swap(x, y)

- #include <cmath>
  - sqrt(x) $\sqrt{x}$
  - pow(x, y) x^y
  - ceil(x) $\lceil x \rceil$
  - floor(x) $\lfloor x \rfloor$

Don’t Repeat Yourself (DRY)

```
1*1= 1  2*1= 2  3*1= 3
1*2= 2  2*2= 4  3*2= 6
1*3= 3  2*3= 6  3*3= 9
1*4= 4  2*4= 8  3*4=12
1*5= 5  2*5=10  3*5=15
1*6= 6  2*6=12  3*6=18
1*7= 7  2*7=14  3*7=21
1*8= 8  2*8=16  3*8=24
1*9= 9  2*9=18  3*9=27
----------------------
4*1= 4  5*1= 5  6*1= 6
4*2= 8  5*2=10  6*2=12
4*3=12  5*3=15  6*3=18
4*4=16  5*4=20  6*4=24
4*5=20  5*5=25  6*5=30
4*6=24  5*6=30  6*6=36
4*7=28  5*7=35  6*7=42
4*8=32  5*8=40  6*8=48
4*9=36  5*9=45  6*9=54
----------------------
7*1= 7  8*1= 8  9*1= 9
7*2=14  8*2=16  9*2=18
7*3=21  8*3=24  9*3=27
7*4=28  8*4=32  9*4=36
7*5=35  8*5=40  9*5=45
7*6=42  8*6=48  9*6=54
7*7=49  8*7=56  9*7=63
7*8=56  8*8=64  9*8=72
7*9=63  8*9=72  9*9=81
```

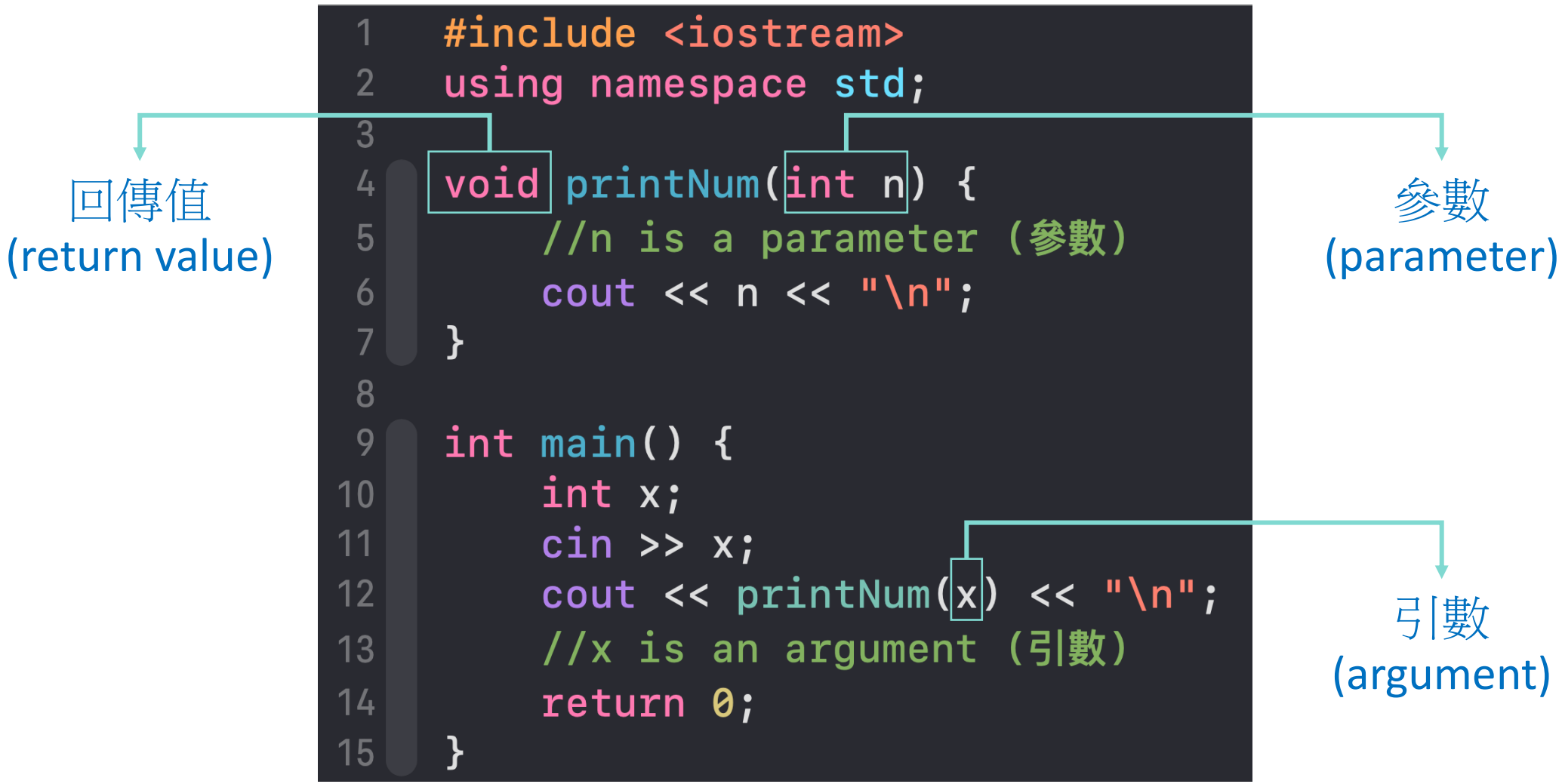
```cpp
for (int j = 1; j <= 9; j++) {
    for (int i = 1; i <= 3; i++) {
        cout << i << "*" << j << "=" << setw(2) << i * j << "  ";
    }
    cout << "\n";
}
cout << "----------------------\n";
for (int j = 1; j <= 9; j++) {
    for (int i = 4; i <= 6; i++) {
        cout << i << "*" << j << "=" << setw(2) << i * j << "  ";
    }
    cout << "\n";
}
cout << "----------------------\n";
for (int j = 1; j <= 9; j++) {
    for (int i = 7; i <= 9; i++) {
        cout << i << "*" << j << "=" << setw(2) << i * j << "  ";
    }
    cout << "\n";
}
```

函數的宣告 (兩種做法)

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

int func(int, int);    函數原型

int main() {
    cout << func(2, 3) << "\n";
    return 0;
}

int func(int x, int y) {
    return x + y;
}
```

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

int func(int x, int y) {
    return x + y;
}

int main() {
    cout << func(2, 3) << "\n";
    return 0;
}
```

自訂函數




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

void printNum(int n) {
    //n is a parameter (參數)
    cout << n << "\n";
}

int main() {
    int x;
    cin >> x;
    cout << printNum(x) << "\n";
    //x is an argument (引數)
    return 0;
}
```

自訂函數 / 無引數、無回傳值

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

void myPrint(void) {
    cout << "***** *****\n";
}

int main() {
    myPrint();
    return 0;
}
```

自訂函數 / 無引數、有回傳值

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

string myInput(void) {
    string s;
    cin >> s;
    return s;
}

int main() {
    cout << myInput() << "\n";
    return 0;
}
```

自訂函數 (call by value) / 有引數、無回傳值

Example 7-1

```
5  void func(int x, int y) {
6      for (int j = 1; j <= 9; j++) {
7          for (int i = x; i <= y; i++) {
8              cout << i << "*" << j << "=" << setw(2) << i * j << "   ";
9          }
10         cout << "\n";
11     }
12     for (int i = 0; i < 8 * (y - x + 1) - 2; i++) {
13         cout << "-";
14     }
15     cout << "\n";
16 }
17
18 int main() {
19     func(2, 5);
20     func(6, 9);
21     return 0;
22 }
```

自訂函數 (call by value) / 有引數、有回傳值

Example 7-1B

```
4  int myMin(int x, int y) {
5      if (x <= y) return x;
6      else return y;
7  }
8
9  int main() {
10     int a = 3, b = 5;
11     cout << myMin(a, b) << "\n";
12     return 0;
13 }
```

自訂函數 (call by reference)

Example 7-2

```
4  void mySwap(int& x, int& y) {
5      int temp = y;
6      y = x;
7      x = temp;
8      cout << "in mySwap(): " << &x << " " << &y << "\n";
9  }
10
11 int main() {
12     int a = 3, b = 5;
13     mySwap(a, b);
14     cout << a << " " << b << "\n";
15     cout << "in main(): " << &a << " " << &b << "\n";
16     return 0;
17 }
```

自訂函數 (return by reference)

Example 7-8

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

int& f(int &x, int &y) {
    if (x > y) return x;
    else return y;
}

int main() {
    int a = 3, b = 5;

    cout << a << " " << b << "\n";
    cout << f(a, b) << "\n";
    cout << a << " " << b << "\n";
    return 0;
}
```

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

int& f(int &x, int &y) {
    if (x > y) return x = 99;
    else return y = 99;
}

int main() {
    int a = 3, b = 5;

    cout << a << " " << b << "\n";
    cout << f(a, b) << "\n";
    cout << a << " " << b << "\n";
    return 0;
}
```

const

Example 7-9

```
1  #include <iostream>
2  using namespace std;
3
4  int& f(const int &x, const int &y) {
5      if (x > y) return x = 99;
6      else return y = 99;
7  }
8
9  int main() {
10     int a = 3, b = 5;
11
12     cout << a << " " << b << "\n";
13     cout << f(a, b) << "\n";
14     cout << a << " " << b << "\n";
15     return 0;
16 }
```

Cannot assign to variable 'x' with const-qualified type 'const int &'

Cannot assign to variable 'y' with const-qualified type 'const int &'

自訂函數 (return by reference)

Example 7-7

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

int a[] = {0, 1, 2, 3, 4};

int f1(int i) {
    //回傳 a[i] 的值 (value)
    return a[i];
}

int& f2(int i) {
    //回傳 a[i] 的參考 (reference)
    return a[i];
}

int main() {
    cout << f1(3) << "\n";

    f2(3) = 99; //修改 a[3] 的值
    cout << a[3] << "\n";

    return 0;
}
```

變數生命週期

- 全域變數 vs. 區域變數
  Global vs. Local variables
- 存放區域變數的記憶體空間有限，約 int a[2000000];
- 陣列開太大，程式執行時會發生 Segmentation fault
- 變通做法?

Example 7-3

```cpp
#include <iostream>
using namespace std;
//global variables (全域變數)
int global = 123;

void mySwap(int& x, int& y) {
    //x, y: local variables (區域變數)
    int temp = y;
    y = x;
    x = temp;
    cout << "in mySwap(): " << global << "\n";
}

int main() {
    int a = 3, b = 5;
    mySwap(a, b);
    cout << a << " " << b << "\n";
    cout << "in main(): " << global << "\n";
    //cout << x << " " << y << "\n";
    return 0;
}
```

全域變數 vs. 區域變數

Example 7-4

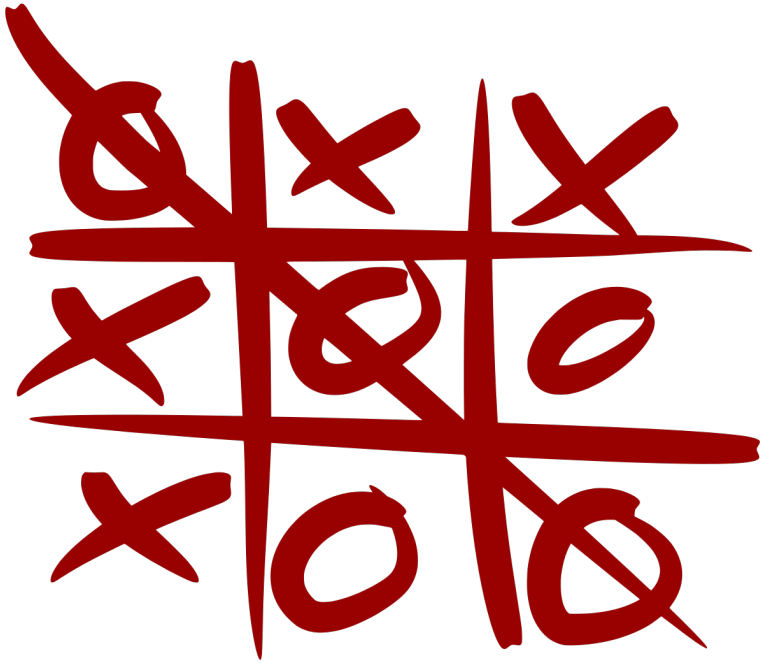
```
3  int mul = 10;
4
5  int myMul_1(int& x) {
6      return x * mul;
7  }
8
9  int myMul_2(int& x) {
10     int mul = 5;
11     return x * mul;
12 }
13
14 int main() {
15     int x = 3;
16     cout << myMul_1(x) << "\n";
17     cout << myMul_2(x) << "\n";
18     return 0;
19 }
```

利用#define，定義簡單的函數

Example 7-10

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

//利用#define，定義簡單的函數
//為了安全起見，每個變數都用括號
#define f(x) (x)*(x)
//#define f(x) x*x

int main() {
    cout << f(1) << "\n";
    cout << f(2) << "\n";
    //(2+1)*(2+1) or 2+1*2+1
    cout << f(2+1) << "\n";
    return 0;
}
```

井字遊戲 (tic-tac-toe)



兩個玩家：player-0 和 player-1
對應的符號：mark[2] = {‘O’, ‘X’}

定義函數：
printBoard()：印出棋盤
place(x, y)：判斷座標(x, y)是否合法
check()：檢查是否連成一線

https://pastebin.ubuntu.com/p/jtWQFWwsSv/