



# 除錯(debug)工具





# Debug 工具可讓你一步一步觀察程式的執行

- 我們以這個程式為例

```
#include <iostream>

using namespace std;

int main()
{
    int num = 1;

    while(num<=10)
    {
        cout << num << endl;
        num = num + 1;
    }

    return 0;
}
```

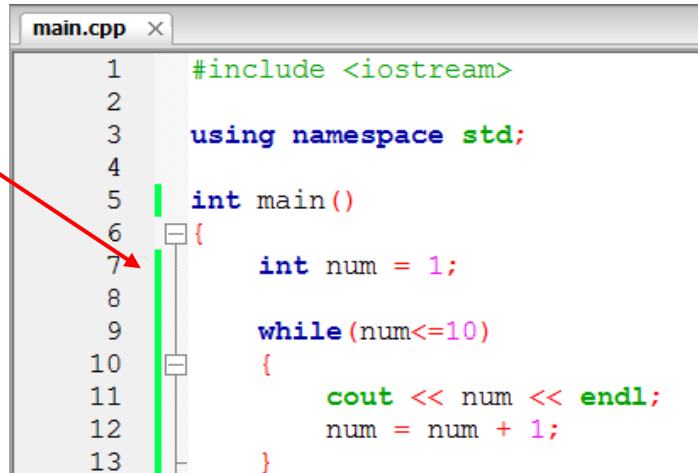
**注意：專案的完整路徑名稱中  
不可以有中文或空白**



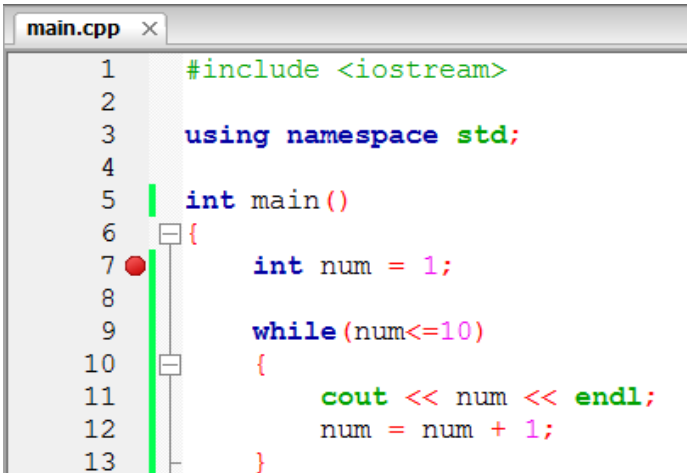


## 設定中斷點

- 在程式碼前面按一下滑鼠左鍵，會出現一個紅圈，這就是中斷點(break point)。
- 你可以依需求，設定許多個中斷點。



```
main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7  int num = 1;
8
9  while(num<=10)
10 {
11     cout << num << endl;
12     num = num + 1;
13 }
```

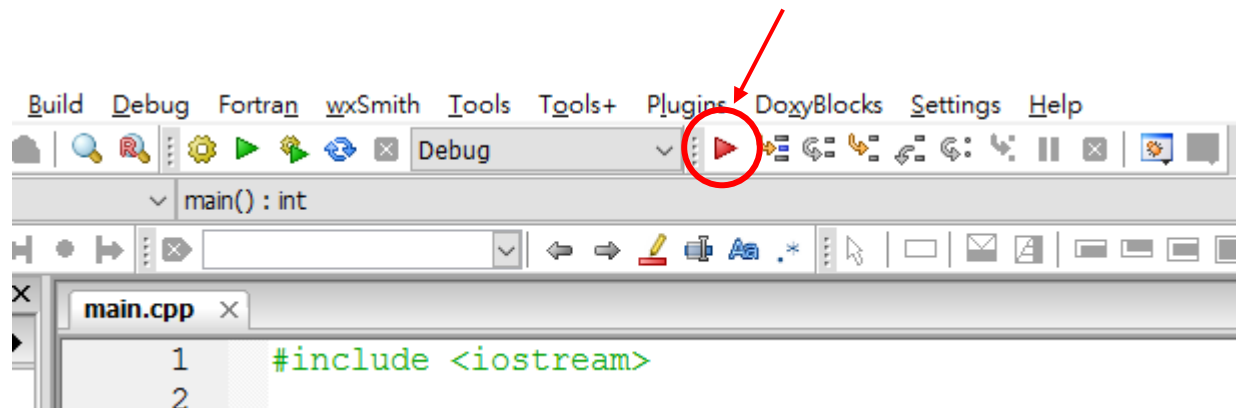


```
main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7  int num = 1;
8
9  while(num<=10)
10 {
11     cout << num << endl;
12     num = num + 1;
13 }
```



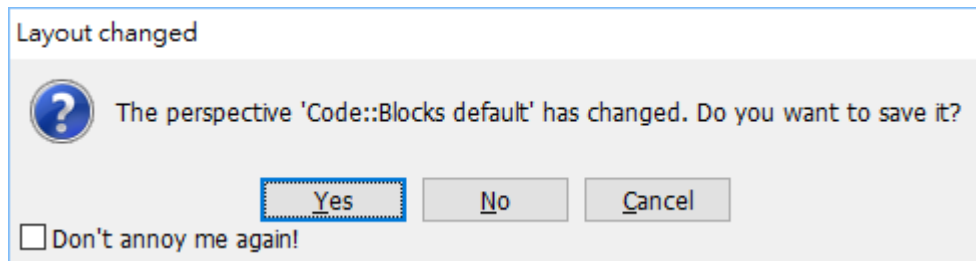
## 開始除錯

- 按下紅色的 Play 鈕，程式會執行到第一個遇到的中斷點





- 如果看到這個訊息，選 [Yes] 即可。
- 它是問你要不要儲存目前的視窗配置。





- 你現在應該會看到這樣的畫面

6 {  
7 int num = 1;  
8

Watches (new)  
Function argur  
Locals  
num 4309710

main.cpp  
1 #include <iostream>  
2  
3 using namespace std;  
4  
5 int main()  
6 {  
7 int num = 1;  
8  
9 while (num <= 10)  
10 {  
11 cout << num << endl;

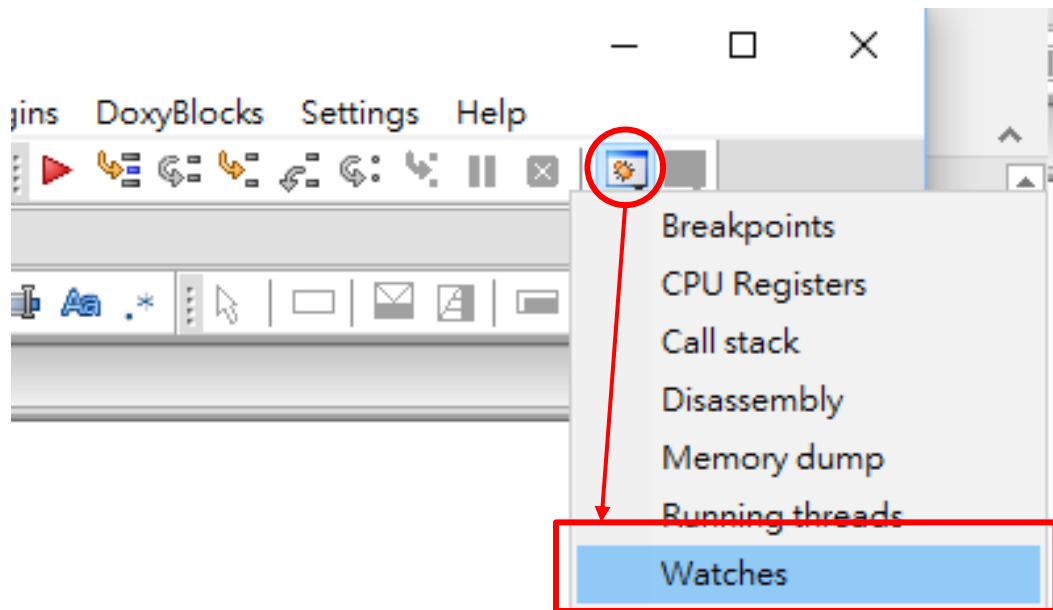
Debugger  
Setting breakpoints  
Debugger name and version: GNU gdb (GDB) 7.6.1  
Child process PID: 12840  
At D:\projects\test\main.cpp:7  
Command:

Memory  
Address: 0x0 Bytes: 32 Go  
(e.g. 0x401060, or &variable, or \$\$eax)  
Cannot access memory at address 0x0

這裡可以看到目前的變數值

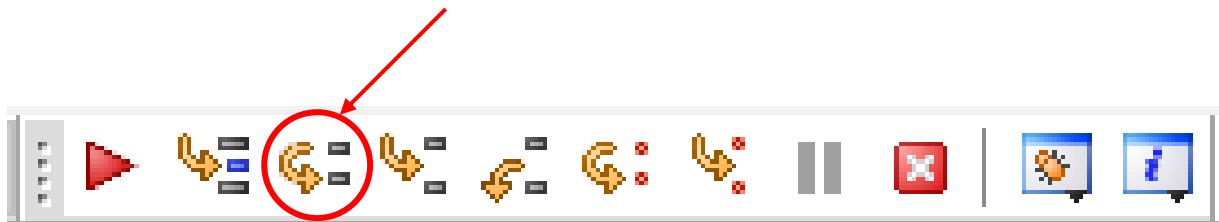
黃色箭頭表示：下一行要執行的程式

- 如果看不到左下角那個 Watch 視窗，可以從這裡把它打開



# 一行一行執行

- 這個按鈕每按一下執行一行，你可以看到程式往哪裡執行，也可以看到各變數數值的變化。







# 觀察重點

- num 的值在哪裡增加？
- console 畫面的輸出
- num 在程式執行中最大值為何？

