CH06 Debug

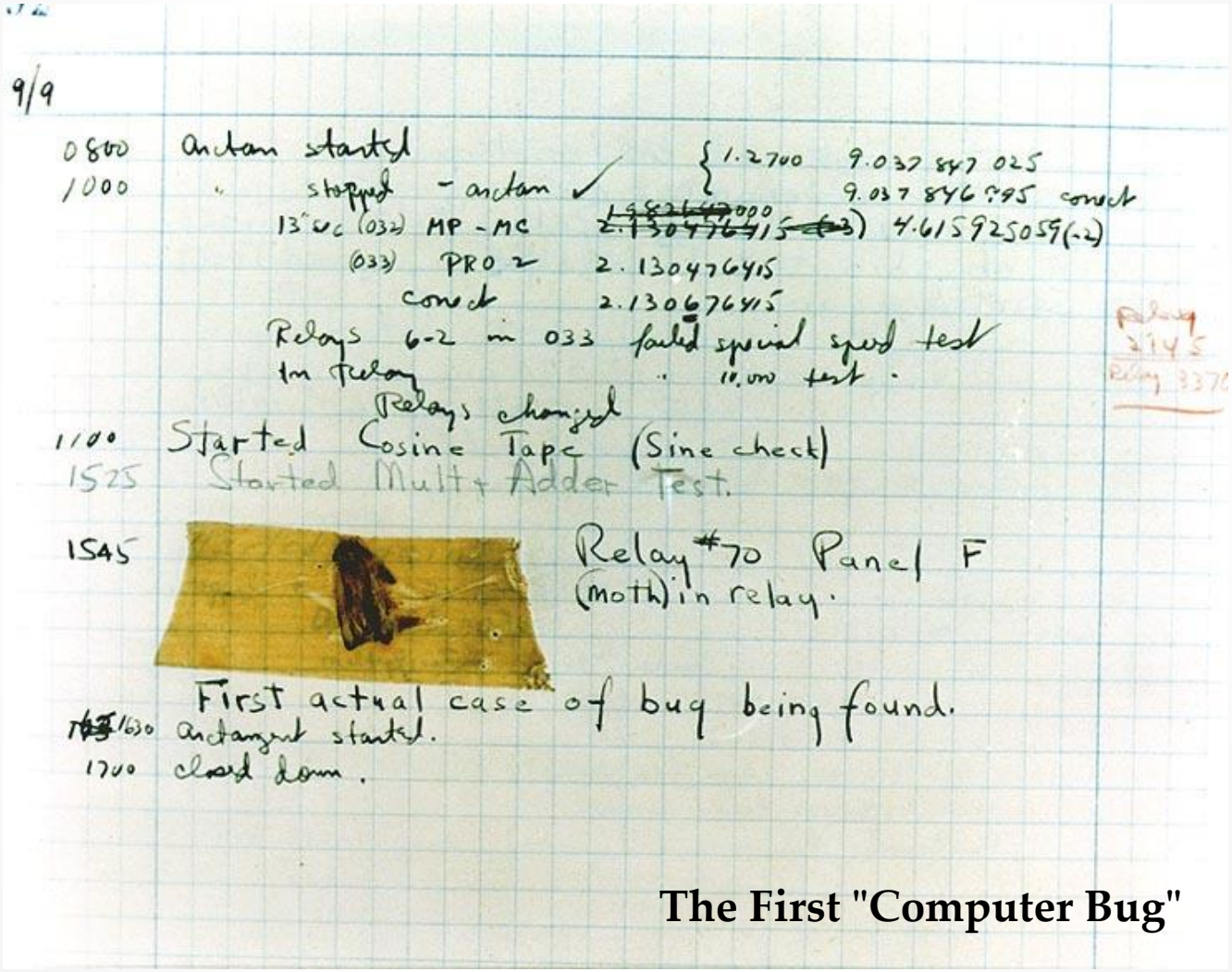
除錯 (Welcome to Hell)

(這不是Bug，這是功能)

主講人：陳靖德

2014/12/03

什麼是Bug？



The First "Computer Bug"

來源：http://commons.wikimedia.org/wiki/File:H96566k.jpg

Bug 種類

- 語法錯誤
- 執行階段錯誤
- 邏輯錯誤

語法錯誤

- 編譯器會報錯的

```
1  #include <stdio.h>
2
3  int main() {
4      printf("Hello, world!!\n")
5      return 0;
6  }
7
```

錯誤: 必須是 ';'

- 這是最好處理的！！！

[題外話]警告(warning)

```
1  #include <stdio.h>
2
3  int main() {
4      int x;
5      if (x == 0) {
6          printf("%d", 20);
7      }
8      return 0;
9  }
10
```

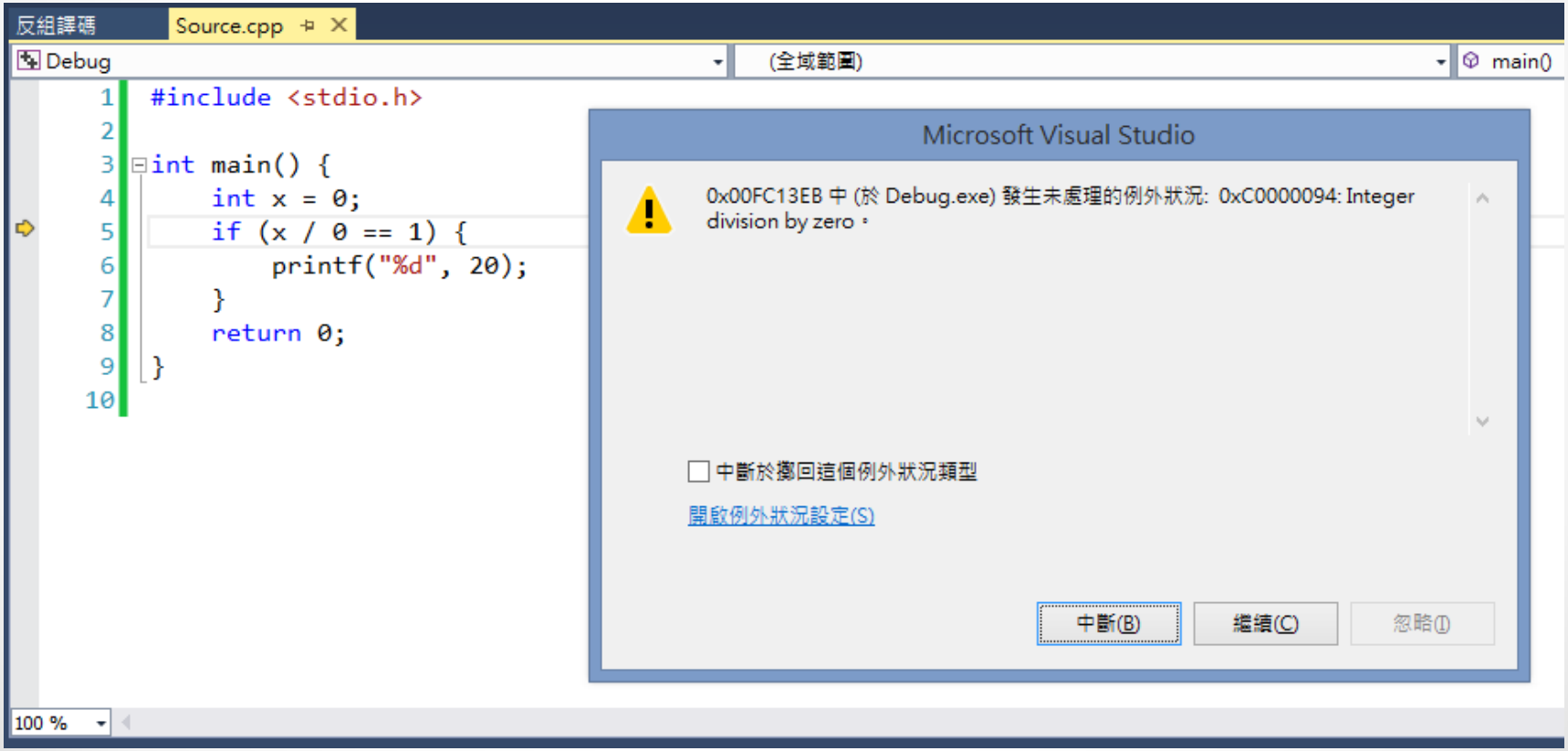
輸出

顯示輸出來源(S): 建置

```
1>------ 已開始建置:  專案: Debug, 組態: Debug Win32 ------
1>  Source.cpp
1>c:\users\          \desktop\debug\debug\source.cpp(5): warning C4700: 使用了未初始化的區域變數 'x'
1>  Debug.vcxproj -> C:\Users\danny50610\Desktop\Debug\Debug\Debug.exe
========== 建置:  1 成功、0 失敗、0 最新、0 略過 ==========
```

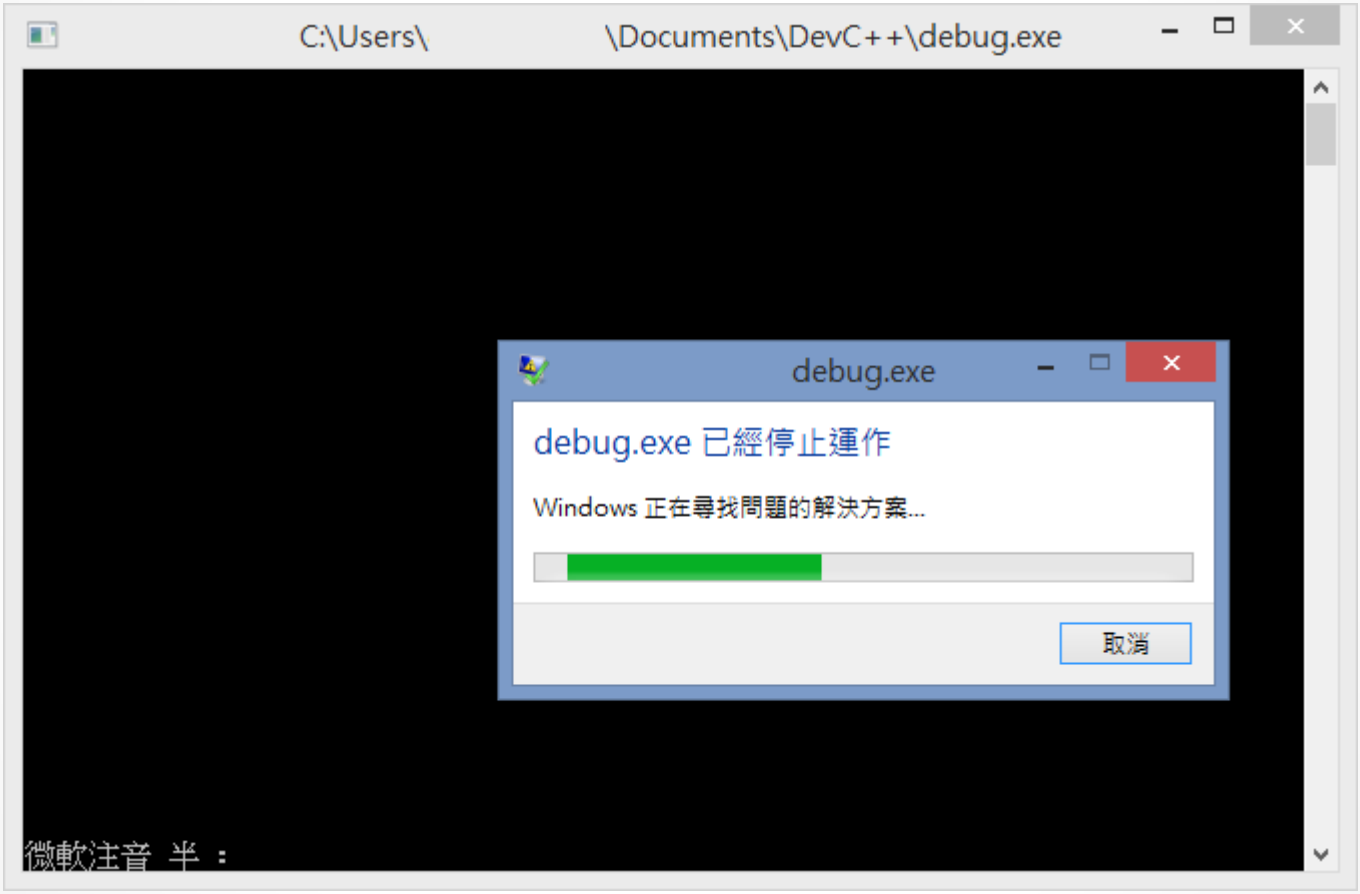
執行階段錯誤

- 程式執行時，發生錯誤(像是整數除以0、存取非法記憶體位置……)



我知道範例很蠢…

執行階段錯誤(續)


C:\Users\ \Documents\DevC++\debug.exe
debug.exe
debug.exe 已經停止運作
Windows 正在尋找問題的解決方案...
取消
微軟注音 半 :

執行階段錯誤(續)

debug.cpp

```
1  #include <stdio.h>
2
3  int main() {
4      int x = 0;
5      if (x / 0 == 1) printf("hello");
6      return 0;
7  }
```

*請用Debug模式

邏輯錯誤

- 程式沒當掉，但行為不是想要的。最難處理

印出0到10的整數，錯在哪裡？

```c
#include <stdio.h>

int main() {
    int i;
    for (i = 0; i = 10; i++) {
        printf("%d ", i);
    }
    return 0;
}
```

如何除錯

沒有固定的方法，這是需要學習的

重點是…

多動腦筋，多做思考

拿起紙筆

常用原則

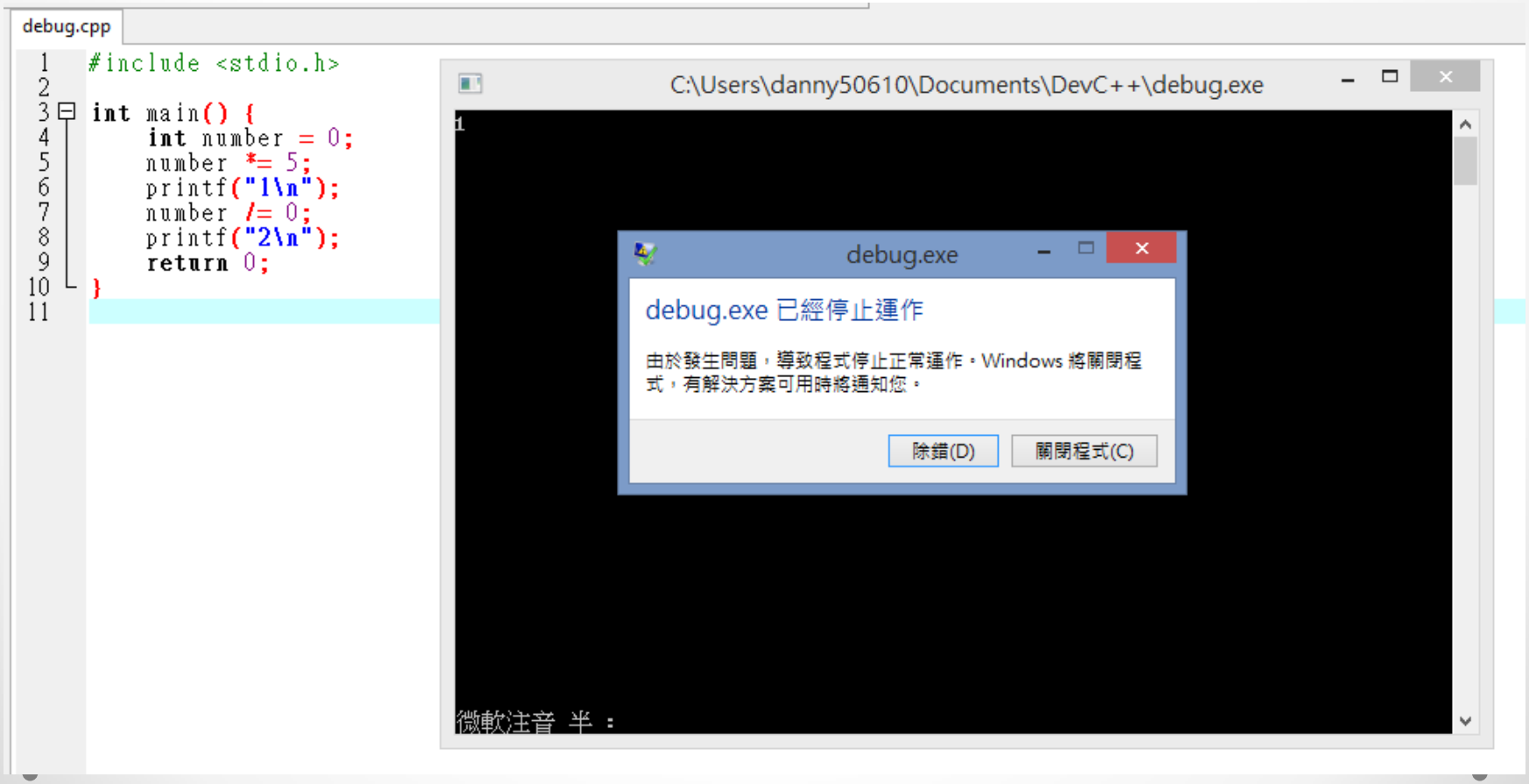
- 了解程式碼
- 尋找可以再次發生錯誤的資料或操作
- 定位錯誤發生的地方
- 小心地修正，避免產生其他Bug
- 驗證假設

  (例如：我覺得這個if會進去，結果事實上一次都沒有)

可再現性

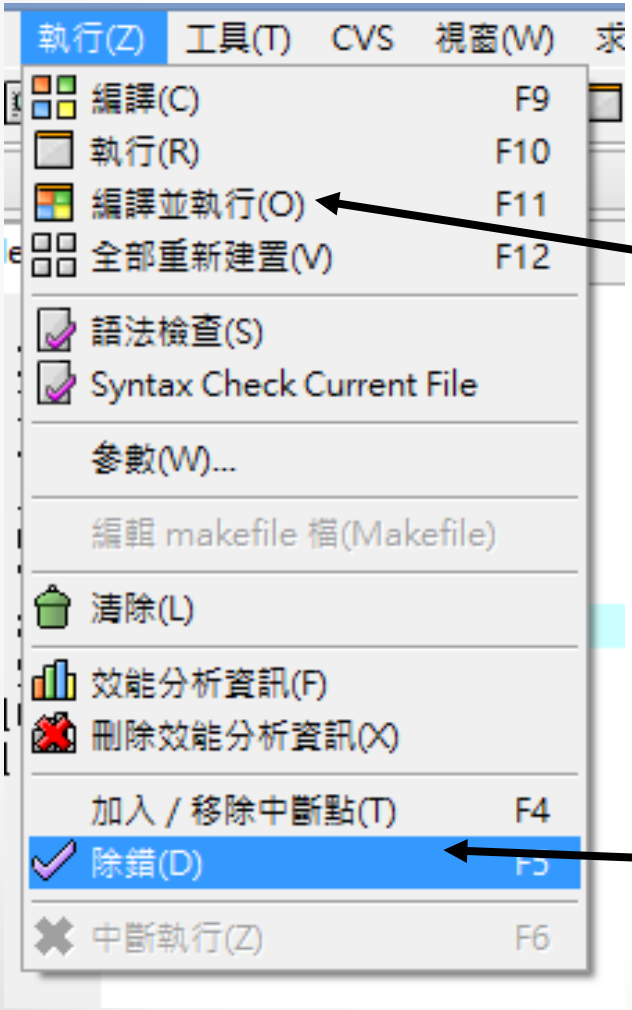
常用技巧(一)

- 在適當的地方printf一些東西，來定位錯誤發生的地方



Debug Tool

- 這裡先介紹Dev C++的基本除錯

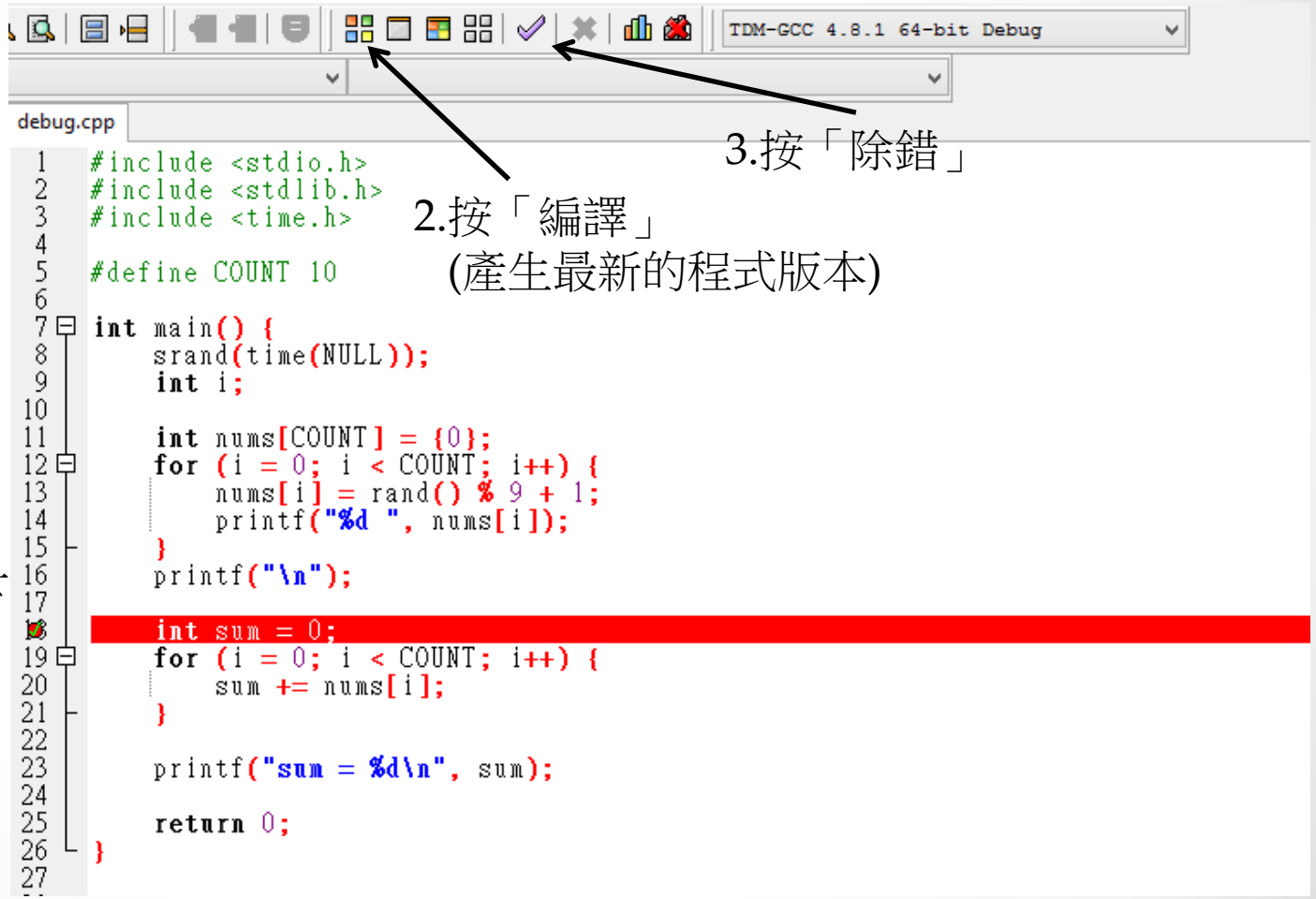


這是平常用的，程式當掉就當掉了

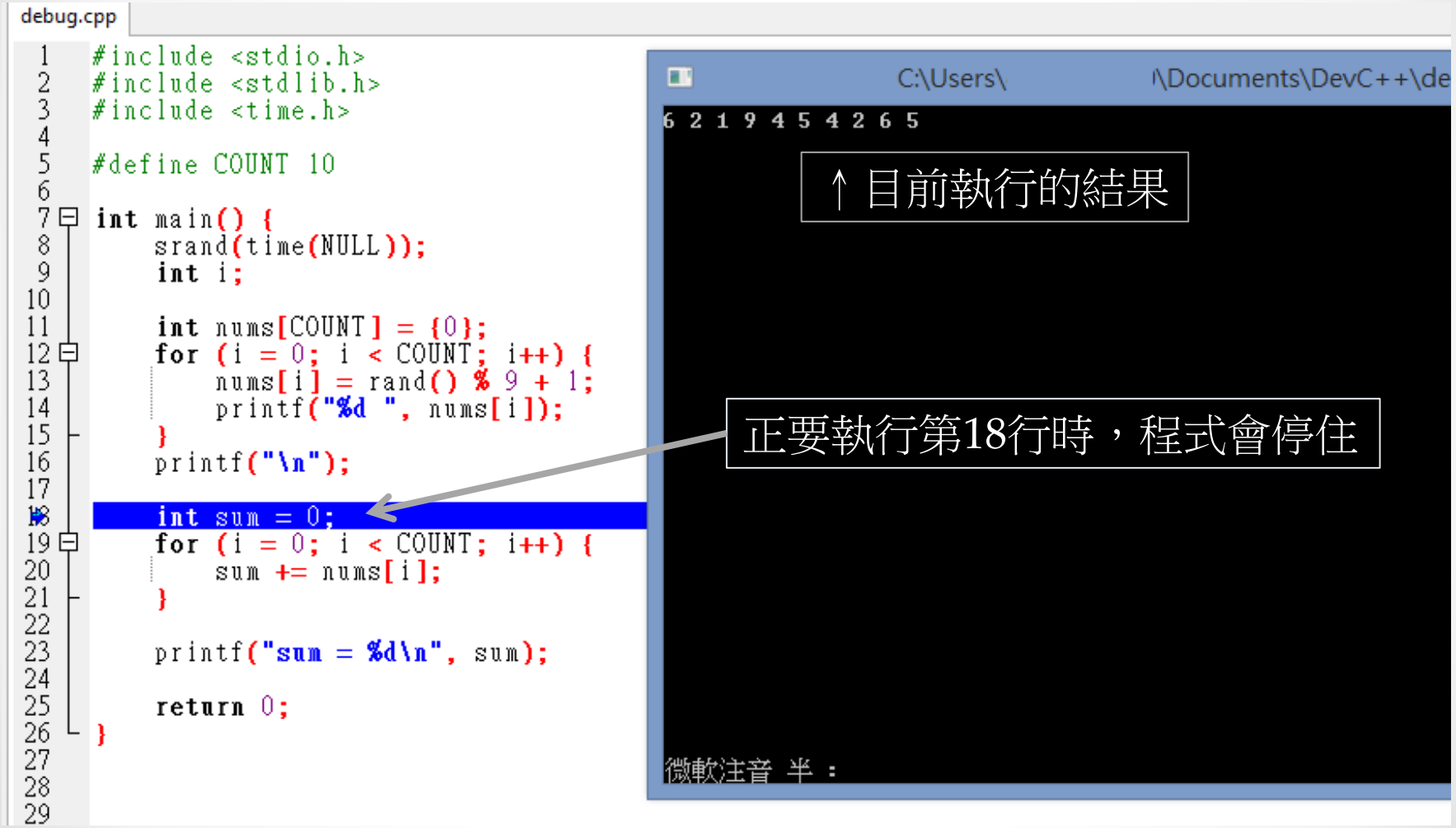
接下來介紹的功能
程式要在「除錯」模式執行
除錯前再點

中斷點

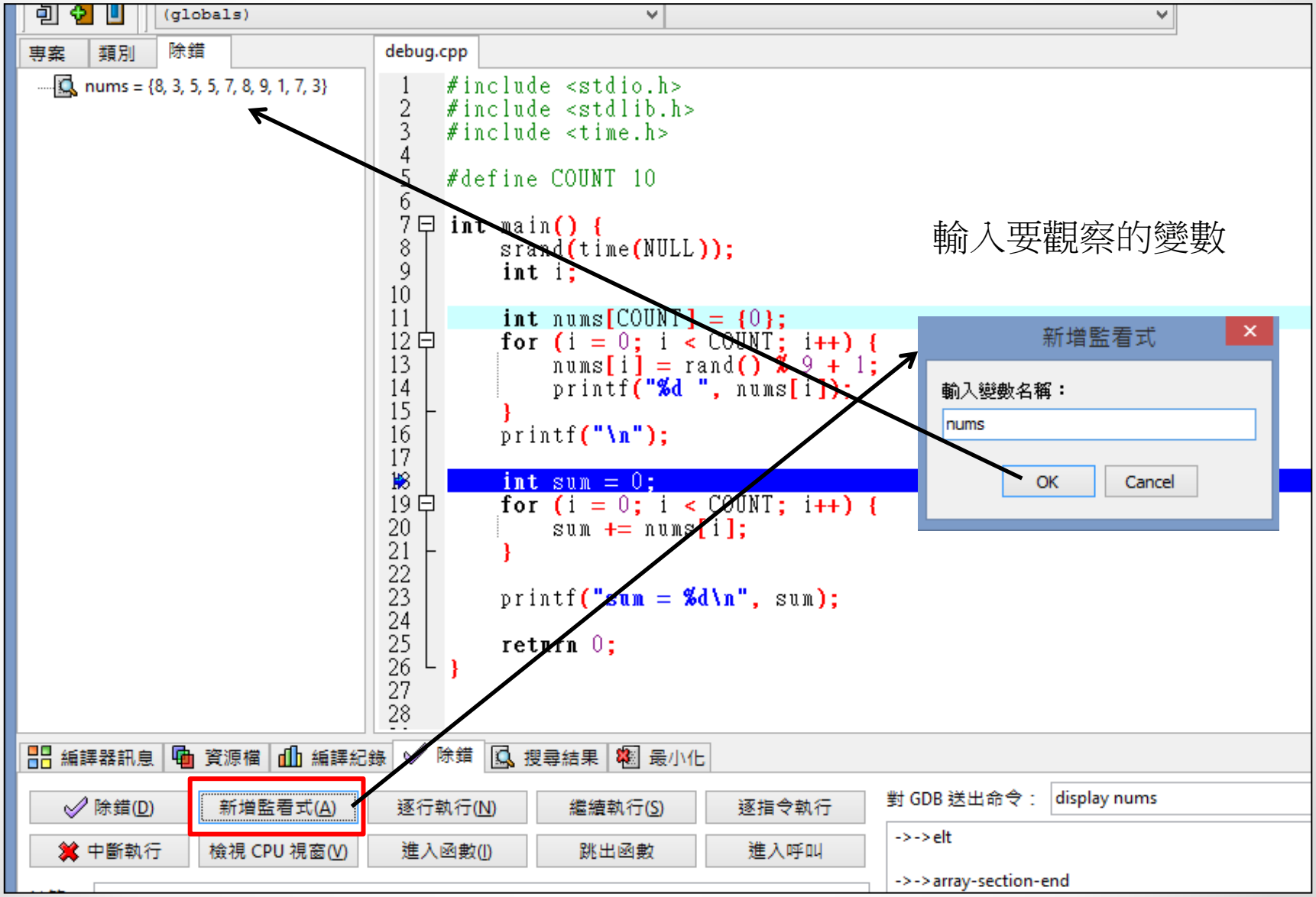
1.在行號的地方點一下
設定中斷點



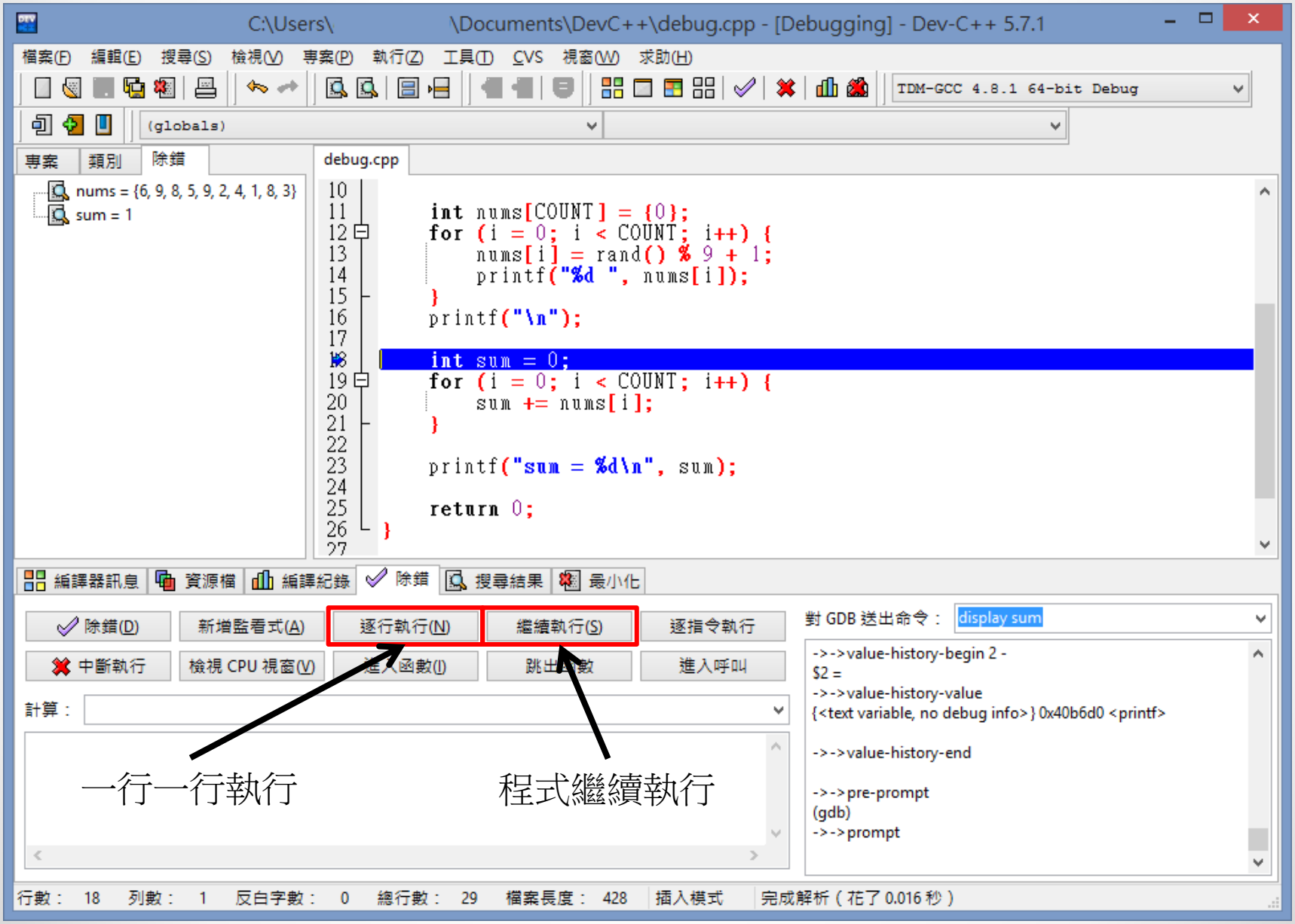
中斷點(續)


debug.cpp
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define COUNT 10
int main() {
srand(time(NULL));
int i;
int nums[COUNT] = {0};
for (i = 0; i < COUNT; i++) {
nums[i] = rand() % 9 + 1;
printf("%d ", nums[i]);
}
printf("\n");
int sum = 0;
for (i = 0; i < COUNT; i++) {
sum += nums[i];
}
printf("sum = %d\n", sum);
return 0;
}
C:\Users\ \Documents\DevC++\de
6 2 1 9 4 5 4 2 6 5
↑目前執行的結果
正要執行第18行時，程式會停住
微軟注音 半 :

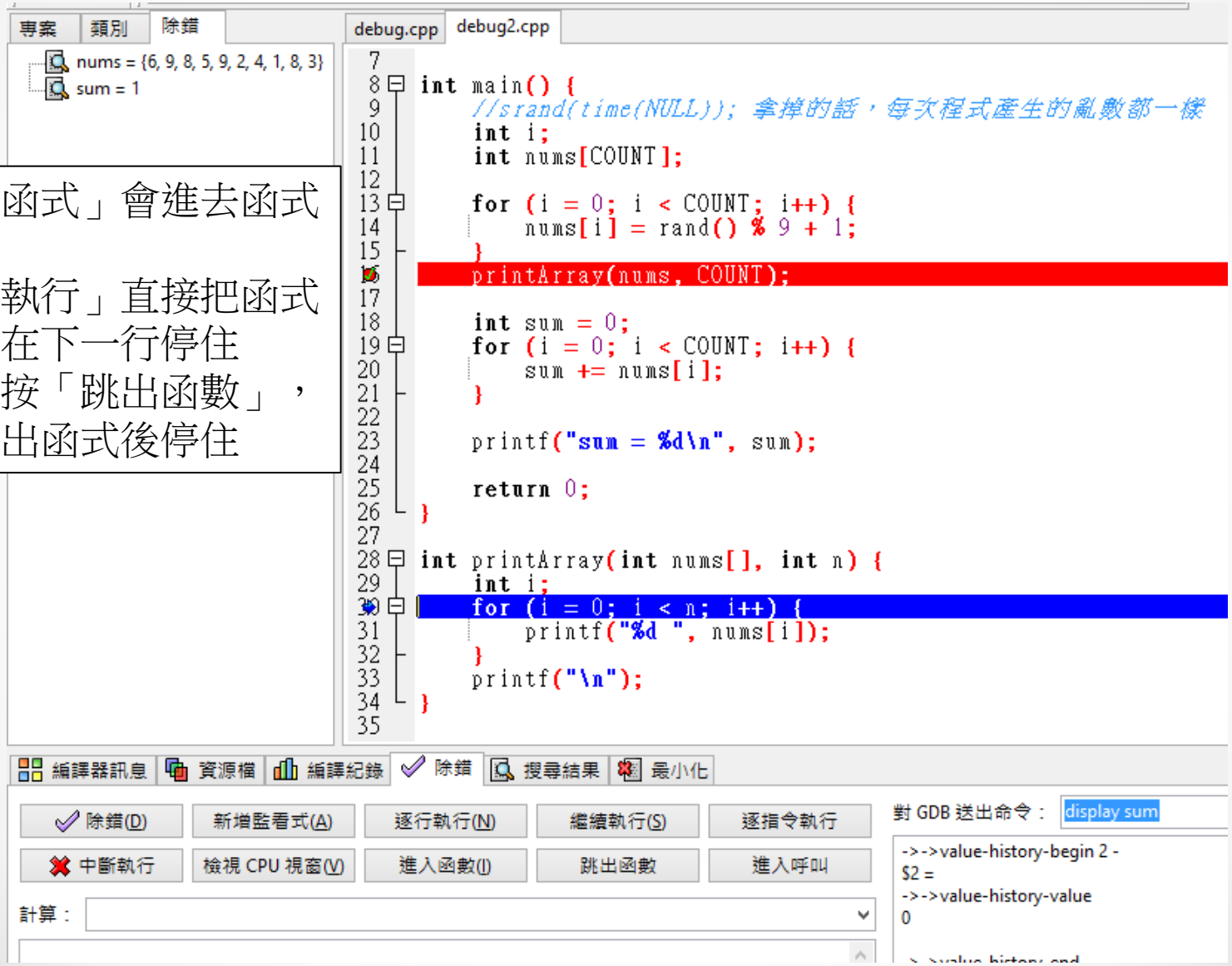
監看式


(globals)
專案
類別
除錯
nums = {8, 3, 5, 5, 7, 8, 9, 1, 7, 3}
debug.cpp
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define COUNT 10
int main() {
srand(time(NULL));
int i;
int nums[COUNT] = {0};
for (i = 0; i < COUNT; i++) {
nums[i] = rand() % 9 + 1;
printf("%d ", nums[i]);
}
printf("\n");
int sum = 0;
for (i = 0; i < COUNT; i++) {
sum += nums[i];
}
printf("sum = %d\n", sum);
return 0;
}
輸入要觀察的變數
新增監看式
輸入變數名稱：
nums
OK
Cancel
編譯器訊息
資源檔
編譯紀錄
除錯
搜尋結果
最小化
除錯(D)
新增監看式(A)
逐行執行(N)
繼續執行(S)
逐指令執行
中斷執行
檢視 CPU 視窗(V)
進入函數(I)
跳出函數
進入呼叫
對 GDB 送出命令：
display nums
->->elt
->->array-section-end

逐行執行、繼續執行


C:\Users\ \Documents\DevC++\debug.cpp - [Debugging] - Dev-C++ 5.7.1
檔案(F) 編輯(E) 搜尋(S) 檢視(V) 專案(P) 執行(Z) 工具(T) CVS 視窗(W) 求助(H)
TDM-GCC 4.8.1 64-bit Debug
(globals)
專案 類別 除錯
nums = {6, 9, 8, 5, 9, 2, 4, 1, 8, 3}
sum = 1
debug.cpp
10 int nums[COUNT] = {0};
11 for (i = 0; i < COUNT; i++) {
12 nums[i] = rand() % 9 + 1;
13 printf("%d ", nums[i]);
14 }
15 printf("\n");
16
17 int sum = 0;
18 for (i = 0; i < COUNT; i++) {
19 sum += nums[i];
20 }
21
22 printf("sum = %d\n", sum);
23
24 return 0;
25 }
26
編譯器訊息 資源檔 編譯紀錄 除錯 搜尋結果 最小化
除錯(D) 新增監看式(A) 逐行執行(N) 繼續執行(S) 逐指令執行
中斷執行 檢視 CPU 視窗(V) 進入函數(I) 跳出函數 進入呼叫
計算：
對 GDB 送出命令： display sum
->->value-history-begin 2 -
$2 =
->->value-history-value
{<text variable, no debug info>} 0x40b6d0 <printf>
->->value-history-end
->->pre-prompt
(gdb)
->->prompt
一行一行執行
程式繼續執行
行數： 18 列數： 1 反白字數： 0 總行數： 29 檔案長度： 428 插入模式 完成解析（花了 0.016 秒）

進入函式、跳出函式



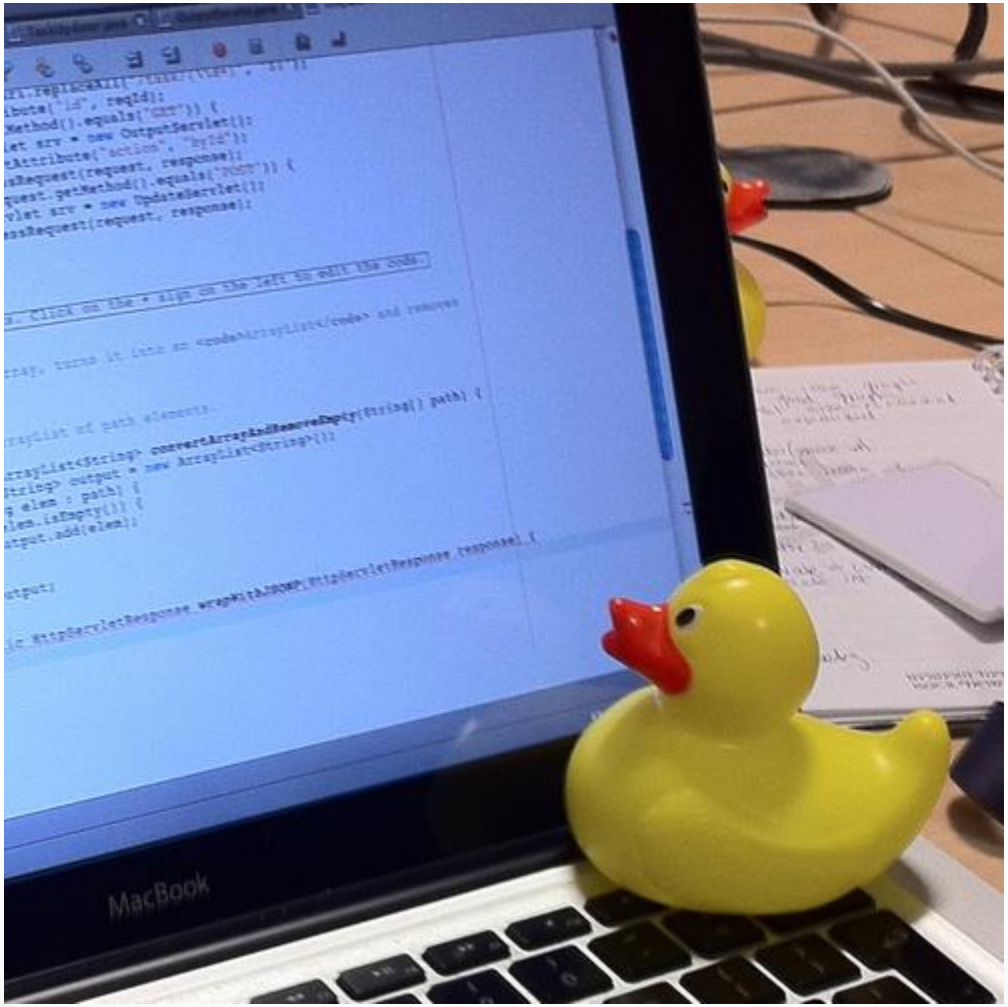
按「進入函式」會進去函式停住
按「逐行執行」直接把函式執行完，在下一行停住
在函式裡按「跳出函數」，會等到跳出函式後停住

預防勝於治療

- 多考慮各種情況(不要走火入魔)
- 變數立即初始化(int x = 0;)
- 每寫完一小段就測試
- ……

找人幫助

Rubber duck debugging



找人幫助

- 大家都是很有愛心的，有時間就會指點(基本上)
- 給予足夠的資訊
  - 程式碼
  - 能夠產生錯誤的操作
  - 錯誤的結果、正確的結果(不然別人不知道你要甚麼)
  - 錯誤訊息(哪一行當掉、程式或編譯器丟出的訊息)
  - 環境資訊(函式庫版本、程式語言、編譯器...)

延伸主題

- bug tracker
  - Bug 回報範例：https://github.com/Azanor/thaumcraft/issues/1041
- 單元測試
- 版本控制系統(git…)
- assert

參考書目、資料

- 編成創藝：編寫出卓越的程式碼(Code Craft：The Practice of Writing Excellend code), Pete Goodliffe / 蔡學庸 譯, 2009
- http://en.wikipedia.org/wiki/Rubber_duck_debugging