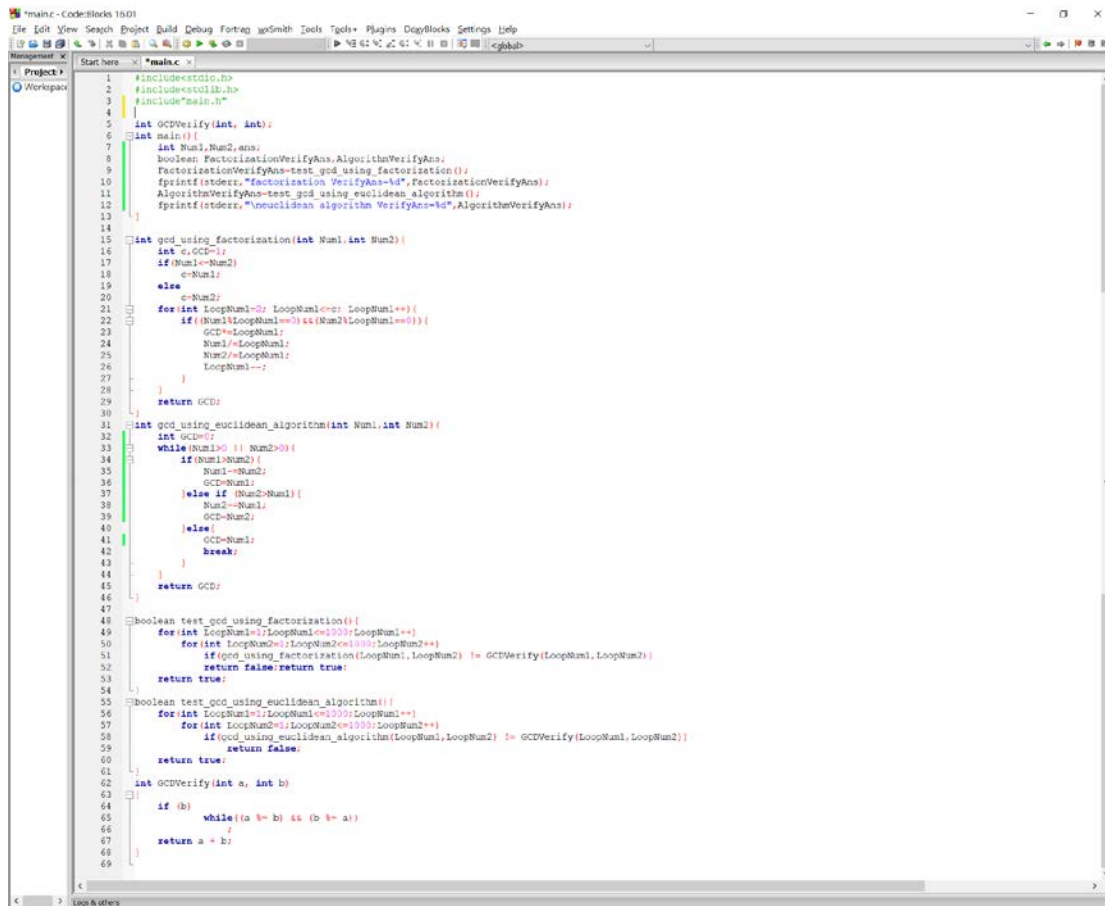


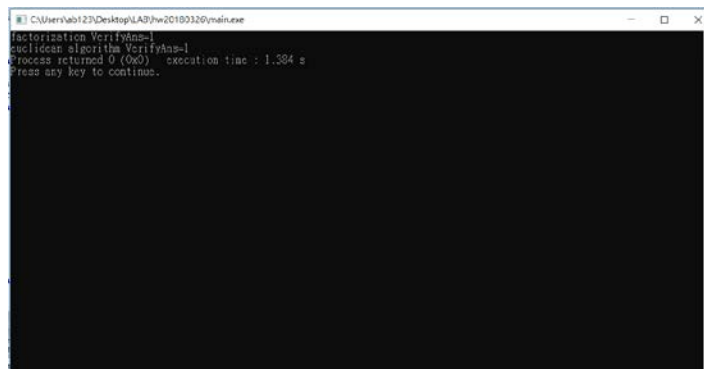
GCD LCM program verify

程式碼:V1



```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include"main.h"
4
5 int GCDVerify(int, int);
6
7 int main() {
8     int Num1, Num2, ans;
9     boolean FactorizationVerifyAns, AlgorithmVerifyAns;
10    FactorizationVerifyAns= test_gcd_using_factorization();
11    printf(stderr, "factorization VerifyAns=%d", FactorizationVerifyAns);
12    AlgorithmVerifyAns= test_gcd_using_euclidean_algorithm();
13    printf(stderr, "euclidean algorithm VerifyAns=%d", AlgorithmVerifyAns);
14
15    int gcd_using_factorization(int Num1, int Num2) {
16        int c, GCD=1;
17        if (Num1<=Num2)
18            c=Num1;
19        else
20            c=Num2;
21        for(int loopNum1=0; LoopNum1<=c; LoopNum1++){
22            if ((Num1%LoopNum1==0) && (Num2%LoopNum1==0)){
23                GCD=LoopNum1;
24                Num1/=LoopNum1;
25                Num2/=LoopNum1;
26                LoopNum1--;
27            }
28        }
29        return GCD;
30    }
31
32    int gcd_using_euclidean_algorithm(int Num1, int Num2) {
33        int GCD=0;
34        while (Num1>0 || Num2>0) {
35            if (Num1>Num2)
36                Num1-=Num2;
37            else if (Num2>Num1)
38                Num2-=Num1;
39            else
40                GCD=Num1;
41            break;
42        }
43        return GCD;
44    }
45
46    boolean test_gcd_using_factorization() {
47        for(int LoopNum1=1; LoopNum1<=1000; LoopNum1++){
48            for(int LoopNum2=1; LoopNum2<=1000; LoopNum2++){
49                if (gcd_using_factorization(LoopNum1, LoopNum2) != GCDVerify(LoopNum1, LoopNum2))
50                    return false; return true;
51            }
52        }
53        return true;
54    }
55
56    boolean test_gcd_using_euclidean_algorithm() {
57        for(int LoopNum1=1; LoopNum1<=1000; LoopNum1++){
58            for(int LoopNum2=1; LoopNum2<=1000; LoopNum2++){
59                if (gcd_using_euclidean_algorithm(LoopNum1, LoopNum2) != GCDVerify(LoopNum1, LoopNum2))
60                    return false;
61            }
62        }
63        return true;
64    }
65
66    int GCDVerify(int a, int b) {
67        if (b)
68            while ((a % b) != 0) b = a % b;
69        return a / b;
70    }
```

執行結果:V1



```
C:\Users\lab123\Desktop\LAB\hw20100320\main.exe
factorization VerifyAns=1
euclidean algorithm VerifyAns=1
Process returned 0 (0x0)   execution time : 1.384 s
Press any key to continue.
```

遇到的問題:使用%來求餘數進行輾轉相除法行運算時發生 crash

問題原因:運算時發生 1/0 的運算導致程式 crash

```
BufNum1 = 1
BufNum2 = 0

27 }
28 return GCD;
29 }
30 int gcd_using_euclidean_algorithm(int Num1
31 int BufNum1, BufNum2, GCD=0;
32 BufNum1=Num1;
33 BufNum2=Num2;
34 while(BufNum1>0 || BufNum2>0){
35     if(BufNum1>BufNum2){
36         BufNum1%=BufNum2;
37         GCD=BufNum1;
38     }
39     else if (BufNum2>BufNum1){
40         BufNum2%=BufNum1;
41         GCD=BufNum2;
```

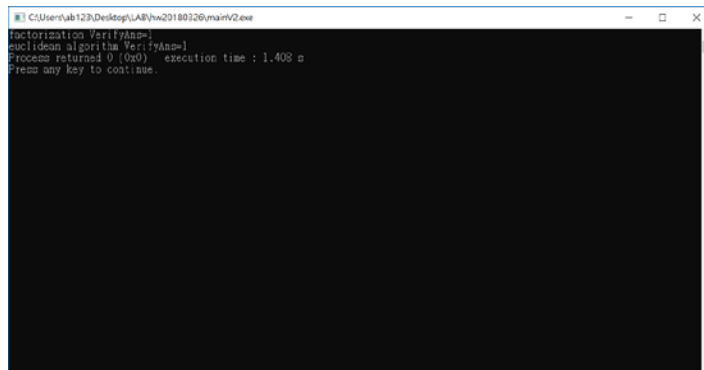
修改後版本

程式碼 V2:

```
mainV2.c - Code::Blocks 16.01
File Edit View Search Project Build Debug fortran wxSmith Tools Plugins DoxyBlocks Settings Help
Start here: mainV2.c
Project: mainV2.c
Workspace:

1 #include<stdio.h>
2 #include<stdlib.h>
3 #include"main.h"
4
5 int GCDVerify(int, int);
6
7 int main()
8 {
9     int Num1, Num2, ans;
10     boolean FactorizationVerifyAns, AlgorithmVerifyAns;
11     FactorizationVerifyAns=test_gcd_using_factorization();
12     printf(stderr, "Factorization VerifyAns=%d", FactorizationVerifyAns);
13     AlgorithmVerifyAns=test_gcd_using_euclidean_algorithm();
14     printf(stderr, "Euclidean algorithm VerifyAns=%d", AlgorithmVerifyAns);
15 }
16
17 int gcd_using_factorization(int Num1, int Num2)
18 {
19     int c, GCD=1;
20     if(Num1==Num2)
21         c=Num1;
22     else
23         c=Num2;
24     for(int LoopNum1=2; LoopNum1<=c; LoopNum1++){
25         if((Num1%LoopNum1==0)&&(Num2%LoopNum1==0)){
26             GCD*=LoopNum1;
27             Num1/=LoopNum1;
28             Num2/=LoopNum1;
29             LoopNum1--;
30         }
31     }
32     return GCD;
33 }
34
35 int gcd_using_euclidean_algorithm(int Num1, int Num2)
36 {
37     int GCD;
38     while(Num1>0 || Num2>0){
39         if(Num1>Num2 && Num2%Num1!=0){
40             Num1=Num2;
41             GCD=Num1;
42             else if (Num2>Num1 && Num2%Num1!=0){
43                 Num2=Num1;
44                 GCD=Num2;
45             }
46             else{
47                 if(Num2>Num1)
48                     GCD=Num1;
49                 else
50                     GCD=Num2;
51                 break;
52             }
53         }
54     }
55     return GCD;
56 }
57
58 boolean test_gcd_using_factorization()
59 {
60     for(int LoopNum1=1; LoopNum1<=100; LoopNum1++)
61         for(int LoopNum2=1; LoopNum2<=100; LoopNum2++)
62             if(gcd_using_factorization(LoopNum1, LoopNum2) != GCDVerify(LoopNum1, LoopNum2))
63                 return false;
64     return true;
65 }
66
67 boolean test_gcd_using_euclidean_algorithm()
68 {
69     for(int LoopNum1=1; LoopNum1<=100; LoopNum1++)
70         for(int LoopNum2=1; LoopNum2<=100; LoopNum2++)
71             if(gcd_using_euclidean_algorithm(LoopNum1, LoopNum2) != GCDVerify(LoopNum1, LoopNum2))
72                 return false;
73     return true;
74 }
75
76 int GCDVerify(int a, int b)
77 {
78     if (b)
79         while((a % b) && (b % a))
80             ;
81     return a * b;
82 }
```

修改後執行結果



```
C:\Users\ak129\Desktop\AB\hw20180326\main\2.exe
Verification Verifying...
euclidean algorithm Verifying...
Process returned 0 (0x0)   execution time : 1.408 s
Press any key to continue.
```

心得:

這次製作的程式是 GCD & LCM 的驗證程式以及.h 檔的應用，在.h 檔的使用上目前沒有遇到什麼問題，但是在程式運算上卻遇到了一些小錯誤。原本在打輾轉相除法的運算的時候是想使用取餘數的方式做運算，可是因為遇到 crash 的問題後來改成用遞減的方式，最後才回頭研究如何修正原本會 crash 的版本，這次主要還是犯了邏輯上面的錯誤，而且修正後反而看起來更繁瑣，且執行時間也沒有比較快，這點必須再修正。而測試程式的部分我使用維基百科上面的程式做參考組，如果兩程式執行結果相同就會回傳 1 反之回傳 0，測試範圍為 1 至 1000。