S&T Batch-1 C Programming Lab 3

Dev C Debugger

?? Feb 2025 6:50pm

Bring along a thumb drive to the lab.
This document is produced by Hu Jialun.

Introduction

Unzip Lab3.zip sent to you by email attachment.

In this session you are going to learn the debugging skill. After you have corrected all the grammatical mistakes in your program and you get a successful compilation, it only means that the syntax of your program is correct. It does not necessarily mean that the logic of your program is correct.

If your program produces the wrong results, how do you find out the mistake? My advice to you is to use the debugger to help you. A debugger enables you to execute your program step by step, make a pause and think for a while, and inspect the values, i.e., to observe the side effect in order to figure out whether the changed values make sense. In this way it is easier for you to correct the logic in your program.

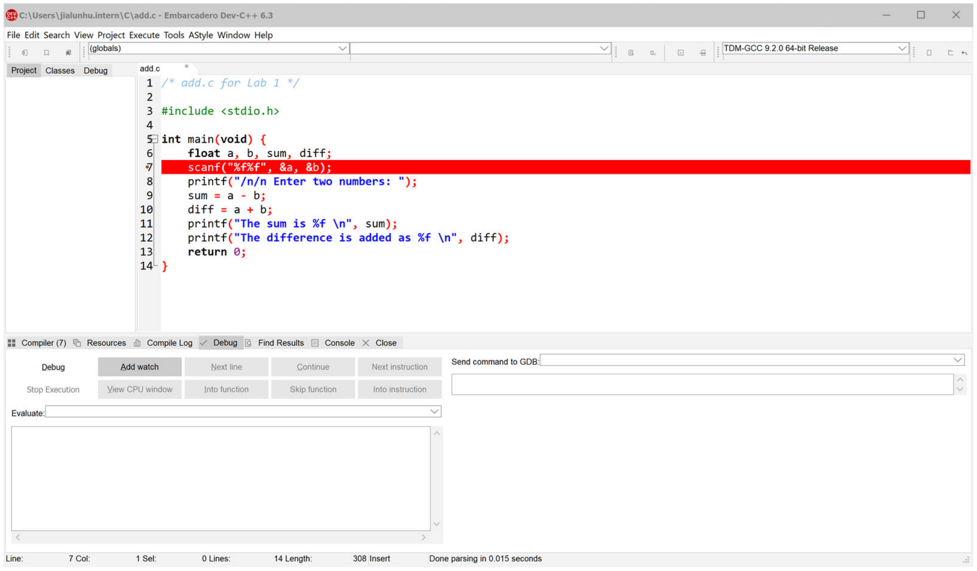
Techniques

Switch to the Debug panel on both the left and bottom panes.
The following tools are provided by the debugger. The illustrations below use the add.c snippet from Lab 1. These debugger concepts apply to most programming languages.

Breakpoint

The program normally runs to its completion without interference, and in order to inspect it amid its execution, we need to somehow pause it and interrupt the normal flow.
A breakpoint pauses program execution whenever this line is reached.

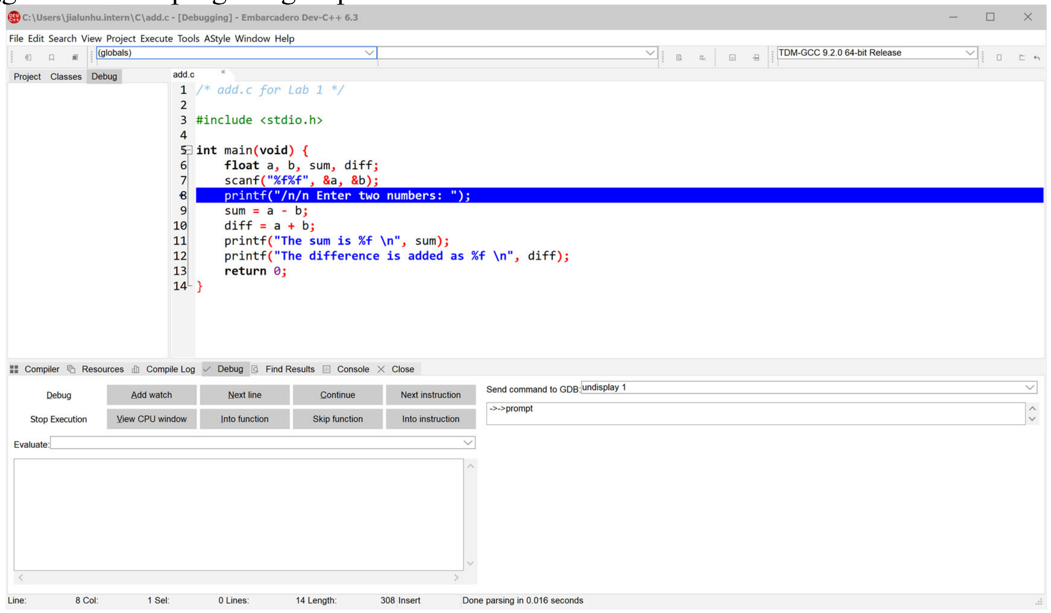
Set one by clicking on the line number. The line will turn red and you should see a red circle in front on the line number.


C:\Users\jialunhu.intern\C\add.c - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
TDM-GCC 9.2.0 64-bit Release
Project Classes Debug
add.c
Compiler (7) Resources Compile Log Debug Find Results Console Close
Debug
Add watch
Next line
Continue
Next instruction
Stop Execution
View CPU window
Into function
Skip function
Into instruction
Send command to GDB:
Evaluate:

```
/* add.c for Lab 1 */

#include <stdio.h>

int main(void) {
    float a, b, sum, diff;
    scanf("%f%f", &a, &b);
    printf("/n/n Enter two numbers: ");
    sum = a - b;
    diff = a + b;
    printf("The sum is %f \n", sum);
    printf("The difference is added as %f \n", diff);
    return 0;
}
```

```
Line:    7 Col:    1 Sel:    0 Lines:    14 Length:    308 Insert    Done parsing in 0.015 seconds
```

The current line where the program is paused gets highlighted in blue. In this case, after you input two numbers and press , you should see the line turning blue, because the breakpoint gets triggered and the program gets paused.


C:\Users\jialunhu.intern\C\add.c - [Debugging] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
TDM-GCC 9.2.0 64-bit Release
Project Classes Debug
add.c
Compiler Resources Compile Log Debug Find Results Console Close
Debug
Add watch
Next line
Continue
Next instruction
Stop Execution
View CPU window
Into function
Skip function
Into instruction
Evaluate:
Send command to GDB:

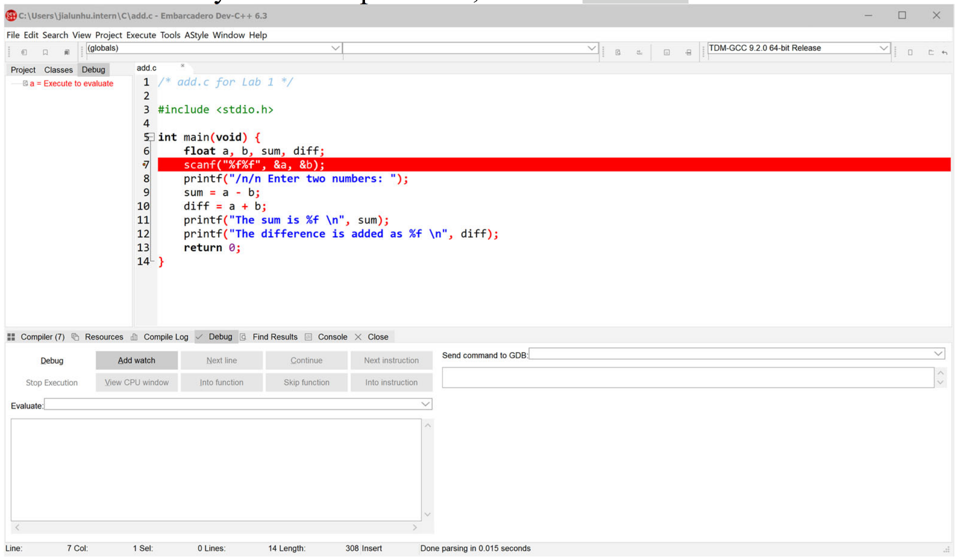
```
/* add.c for Lab 1 */

#include <stdio.h>

int main(void) {
    float a, b, sum, diff;
    scanf("%f%f", &a, &b);
    printf("/n/n Enter two numbers: ");
    sum = a - b;
    diff = a + b;
    printf("The sum is %f \n", sum);
    printf("The difference is added as %f \n", diff);
    return 0;
}
```

```
Send command to GDB: undisplay 1
->->prompt
```

```
Line: 8 Col: 1 Sel: 0 Lines: 14 Length: 308 Insert Done parsing in 0.016 seconds
```

Watch

Specify variables or expressions you want to see whenever the execution is paused.
To add a watch, use **Debug>Add watch**, and input the expression you want to watch, such as a. You can also watch any valid C expressions, such as a * 2 + 1.


C:\Users\jialunhu.intern\C\add.c - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
TDM-GCC 9.2.0 64-bit Release
Project Classes Debug
add.c
Compiler (7) Resources Compile Log Debug Find Results Console Close
Debug
Add watch
Next line
Continue
Next instruction
Stop Execution
View CPU window
Into function
Skip function
Into instruction
Evaluate:
Send command to GDB:

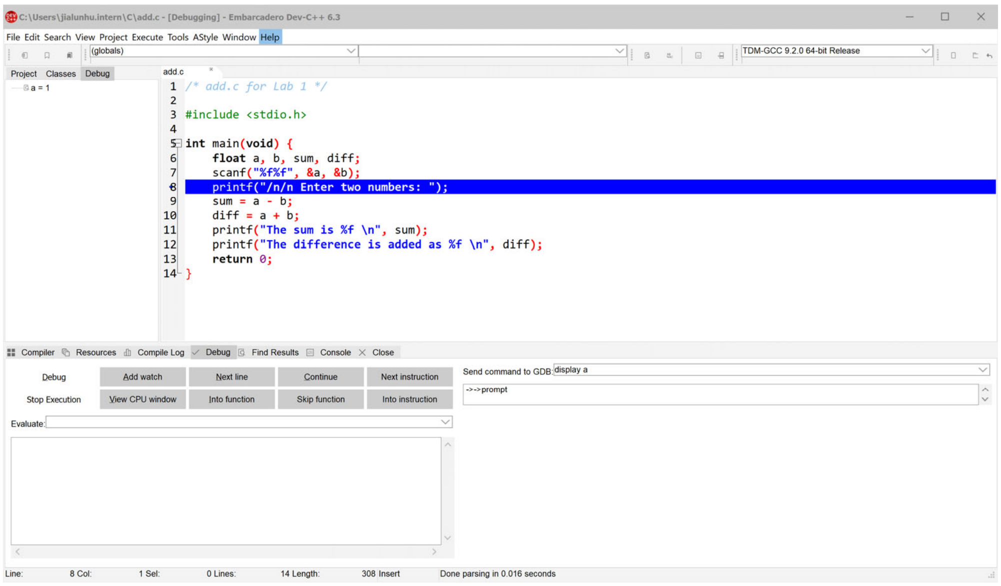
```
a = Execute to evaluate
```

```
/* add.c for Lab 1 */

#include <stdio.h>

int main(void) {
    float a, b, sum, diff;
    scanf("%f%f", &a, &b);
    printf("/n/n Enter two numbers: ");
    sum = a - b;
    diff = a + b;
    printf("The sum is %f \n", sum);
    printf("The difference is added as %f \n", diff);
    return 0;
}
```

```
Line: 7 Col: 1 Sel: 0 Lines: 14 Length: 308 Insert Done parsing in 0.015 seconds
```

The left Debug pane will show the expression(s) you are watching and their respective values. Before you start executing the program in the debugger, the watches are shown as "Execute to evaluate", since there are no values to be shown yet.

During execution, whenever the program is paused, the watched expressions in the left panel will be updated to reflect the current value.


C:\Users\jialunhu.intern\C\add.c - [Debugging] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
TDM-GCC 9.2.0 64-bit Release
Project Classes Debug
add.c
Compiler Resources Compile Log Debug Find Results Console Close
Debug
Add watch
Next line
Continue
Next instruction
Stop Execution
View CPU window
Into function
Skip function
Into instruction
Evaluate:

a = 1

```c
/* add.c for Lab 1 */

#include <stdio.h>

int main(void) {
    float a, b, sum, diff;
    scanf("%f%f", &a, &b);
    printf("/n/n Enter two numbers: ");
    sum = a - b;
    diff = a + b;
    printf("The sum is %f \n", sum);
    printf("The difference is added as %f \n", diff);
    return 0;
}
```

Send command to GDB: display a

```
->->prompt
```

Line: 8 Col: 1 Sel: 0 Lines: 14 Length: 308 Insert Done parsing in 0.016 seconds

Next line (aka step over)

Execute one line in the current function and pause again.

Skip function (aka step out)

Execute until the end of the current innermost function and pause again.

Into function (aka step in)

Execute one line and pause again. If the line calls another function, jump inside the callee instead of jump over the invocation. This is useful when you want to trace into a function you wrote.

Note

Use step over instead of step in with library functions such as printf, since we are normally not interested in the internals of those, and their implementations are not available in source form on the lab computers anyways.

If you accidentally wander into that territory, use step out to come back to your own code.

Continue

Continue execution until the program gets paused again or terminates.

Note

The debugger can only manipulate a program when the program is paused or has not started running. When the program is running its own course, you are not able to configure the debugger options (e.g. add a new breakpoint), and information such as watches are not updated.

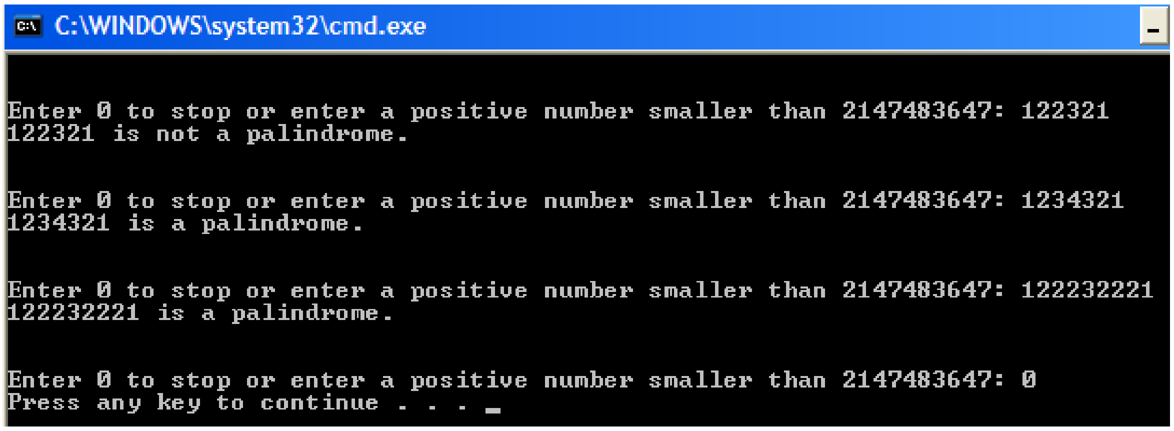
Also, waiting for scanf input does not automatically make a program "paused". The program is still running, although it seems stuck.

Assignment 1

Two programs, ex1.c and ex2.c (already given to you by email attachment in Lab3.zip) do not produce the right result. Use debugger to execute the program step by step, inspect the side effect on the variables and the screen output after each instruction is executed to make the corrections.

Assignment 2

An integer is a palindrome if its numerical value read from left to right is equal to the numerical value read from right to left. For example, the integer 5162772615 is a palindrome while 123211 isn't. Write a full program in C language to read an integer from the keyboard and check if the integer is a palindrome. The program runs repeatedly until 0 is entered as the integer. A session of program execution is as follows:


C:\WINDOWS\system32\cmd.exe

```
Enter 0 to stop or enter a positive number smaller than 2147483647: 122321
122321 is not a palindrome.

Enter 0 to stop or enter a positive number smaller than 2147483647: 1234321
1234321 is a palindrome.

Enter 0 to stop or enter a positive number smaller than 2147483647: 122232221
122232221 is a palindrome.

Enter 0 to stop or enter a positive number smaller than 2147483647: 0
Press any key to continue . . . _
```

As an illustration, let a 5-digit number be represented as $a_4a_3a_2a_1a_0$. The number is a palindrome if $a_0 \times 10^4 + a_1 \times 10^3 + a_2 \times 10^2 + a_3 \times 10^1 + a_4 \times 10^0 = a_4a_3a_2a_1a_0$.

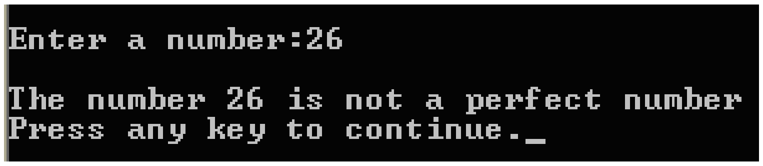
The % operator used to compute the remainder from the integer division can be used in this program.

Assignment 3

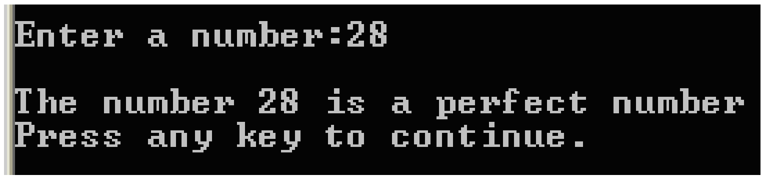
A perfect number n is a positive integer which sum of its positive divisors excluding n is equal to n. For example, 6 is a perfect number since the relevant divisors are 1, 2 and 3. The sum of these divisors is 1 + 2 + 3 = 6.

The next perfect number is 28 since 1 + 2 + 4 + 7 + 14 = 28.

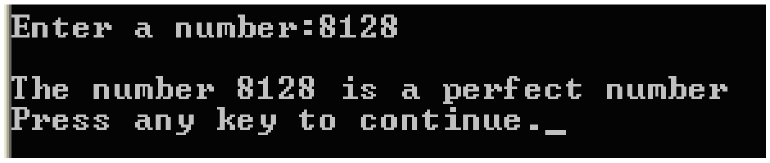
Write a C program named as **perfect.c** to check whether a number entered from the keyboard is a perfect number or not. These are the results of 3 separate complete executions of the program:



```
Enter a number:26

The number 26 is not a perfect number
Press any key to continue._
```



```
Enter a number:28

The number 28 is a perfect number
Press any key to continue.
```



```
Enter a number:8128

The number 8128 is a perfect number
Press any key to continue._
```

You should catch up with your Lab-1 and Lab-2 if you have not finished the assignments given in the first two lab sessions.