SM2-21st C Programming Lab 3

Debugging Techniques and Conditional and Iterative Instructions

7 Sep 2017 Thursday 6:45pm
Dinner will be served at 6pm outside LT27.

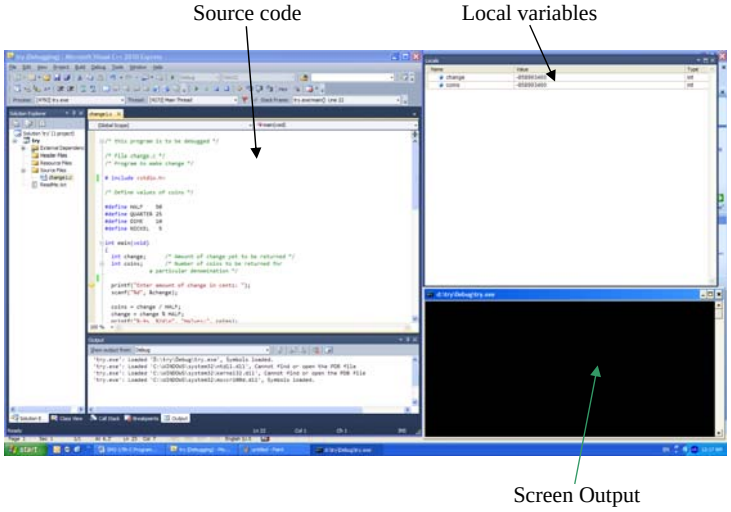
Unzip Lab3.zip sent to you by email attachment. In this session you are going to learn the debugging skill. After you have corrected all the grammatical mistakes in your program and you get a successful compilation, it only means that the syntax of your program is correct. It does not necessarily mean 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.

Debugging skill needs not be complex. After you have successfully complied your program with the use of Visual C++, you can use the function keys F10 (Step over), or F11 (Step into) to execute your program step by step. I will use the program **change1.c** (already given to you in email attachment) as an example. The result of the program is wrong as shown above.

```
C:\WINDOWS\system32\cmd.exe
Enter amount of change in cents: 97
Halves:     1
Quarters:  22
Dimes:      0
Nickels:    0
Pennies:    1
Press any key to continue . . .
```

To debug this program, you use the F10 key to start the debugging process. Press the F10 twice. Subsequently I suggest you divide your screen into 3 segments, namely (i) source code, (ii) local variables and (iii) screen output as shown below.



To display the local variables, you should use these clicks: **Debug**, **Windows**, **Locals**. You are advised to adjust the frame size of the panel and drag the frame for the local variables to the right top corner of your screen. Subsequently you can use the Alt-Tab key to navigate to the frame for Output Screen, adjust the frame size and drag it to the right bottom corner.

After the screen is customized into 3 segments, you have to place the mouse cursor at the frame for source codes and click once to make it an active frame. Every time when you press the F10 key, you should observe the side effect on the local variables and output screen and check whether the changes have made sense. In this way you will learn to debug your program.

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: