Debugging Techniques

Lecture 9
Sections 3.11, 5.13

Robb T. Koether

Hampden-Sydney College

Fri, Sep 14, 2018

Outline

Hand-Tracing

- As demonstrated in Lab 2, we can hand-trace a simple program.
- Set up a table with one column for each variable.
- Fill in the initial values of each variable.
- As you process each statement, update the values of the variables.
- Find the first place where your results are different from the programs results.

Hand-Tracing Example

Hand-Tracing Example

```
int a = 5;
int b = 8;
int a = a * b;
int b = a - 30;
int c = a + b + 9;
int d = c / b
int e = c % b;
```

a	b	c	d	e

Hand-Tracing Example

Hand-Tracing Example

```
int a = 5;
int b = 8;
int a = a * b;
int b = a - 30;
int c = a + b + 9;
int d = c / b
int e = c % b;
```

a	b	c	d	e
5	8			

Hand-Tracing Example

Hand-Tracing Example

```
int a = 5;
int b = 8;
int a = a * b;
int b = a - 30;
int c = a + b + 9;
int d = c / b
int e = c % b;
```

a	b	c	d	e
5	8			
40				

Hand-Tracing Example

Hand-Tracing Example

```
int a = 5;
int b = 8;
int a = a * b;
int b = a - 30;
int c = a + b + 9;
int d = c / b
int e = c % b;
```

a	b	c	d	e
5	8			
40	10			

Hand-Tracing Example

Hand-Tracing Example

```
int a = 5;
int b = 8;
int a = a * b;
int b = a - 30;
int c = a + b + 9;
int d = c / b
int e = c % b;
```

a	b	c	d	e
5	8	59		
40	10			

Hand-Tracing Example

Hand-Tracing Example

```
int a = 5;
int b = 8;
int a = a * b;
int b = a - 30;
int c = a + b + 9;
int d = c / b
int e = c % b;
```

a	b	c	d	e
5	8	59	5	
40	10			

Hand-Tracing Example

Hand-Tracing Example

```
int a = 5;
int b = 8;
int a = a * b;
int b = a - 30;
int c = a + b + 9;
int d = c / b
int e = c % b;
```

a	b	c	d	e
5	8	59	5	9
40	10			

Outline

Using the Visual Studio Debugger

- A C++ program may be executed one line at a time by using the debugger.
- To start the debugger, press F10.
- The lower-left window will show the values of all “local” variables (variables in `main()`) as they are defined.
- The yellow arrow in the left margin indicates the statement to be executed next.
- To advance to the next statement, press F10.

Example

- Run `MakeChange.cpp`.
- Run `Savings.cpp`.

Entering Functions

- If the program involves a function call (to be discussed later), then press F11 to enter the function (“step into”).
- Otherwise, press F10 to execute the entire statement, including the function call (“step over”).
- When the last function statement is executed, the debugger will return to the calling statement.
- Or press Shift-F11 to complete the function and return at once (“step out”).

Example

- Run `GradeReport.cpp` and `GradeStats.cpp`.

Breakpoints

- Often, the program will execute a very large number of statements before reaching the statement with the error.
- To advance quickly, set a breakpoint on the statement in question.
- Click in the left-most gray margin to set the breakpoint. A red dot appears.
- Click on a breakpoint to remove it.
- Press the green arrow labeled “Continue” to execute down to the next breakpoint.

Example

- Run `FloatAndDouble.cpp` from Lab 2.

Outline

Using Output Statements

- Often, it is overkill to step through the program, even with breakpoints set.
- A faster way to learn what is happening is to display the values of critical variables at key points in the program.
- This technique is especially useful in longer, more complicated programs.

Example

- Run `Savings`.

Outline

Assertions

- In computer programs, an assertion is a statement that is supposed to be true.
- The `assert()` macro will test an assertion to see whether it is true.
  - If it is true, then the program continues without interruption.
  - If it is false, then the program halts and displays an error message.

Assertions

- Include the header file `cassert`.

  ```
  #include <cassert>
  ```

- At appropriate points in the program, add the statement

  ```
  assert(assertion);
  ```

- Use

  ```
  #define NDEBUG
  ```

  to deactivate the assertions without removing them.

Example

- Run `MixedNuts.cpp`.

Outline

Assignment

Assignment

- Read Sections 3.11 and 5.13.