The Development Process

Syntax, Semantics, The Curator, and Debugging

Syntax and Semantics

- *Syntax* is the set of rules for combining language components (words, punctuation, etc.) into valid instructions.
- *Semantics* is the set of rules defining the meaning of valid instructions.
Syntax and Semantics (Example)

- Syntax for the assignment statement
  
  \[
  \text{Variable} = \text{Expression} ;
  \]

- Semantics
  - Assign the value of the expression on the right of the equals sign to the variable named on the left.

Syntax Errors

- Failed to follow the syntax rules
- Detected by the compiler
  - When you build your executable
- Examples
  - Forgetting a semicolon at the end of declarations
  - Using a variable name that is not declared
    - Misspelling or different capitalization
  - Forgetting a curly brace to end the main program
  - Forgetting a comma between two identifiers in a variable declaration
  - Assigning a string to an int variable
Syntax Errors (Recovery Examples)

- Look carefully at the line the error occurred on to see if you broke a syntax rule (missing semicolon)
- For invalid identifiers, check your declarations and the line of the error for misspelling or different capitalization.

Linker Errors

- When the linker cannot find object files or other information for building your program
  - After the Linking... message
- Examples
  - Misspelled standard features to include like using `iostraem` instead of `iostream`
  - Using wrong project type
    - `WinMain` is somewhere in the error message
Execution Errors

- Program compiles just fine (Yahoo!), but something goes wrong during execution
  - Division by 0
- Detected when you run the program
- Computer usually prints an error message and stops
  - Sometimes you'll see the "Blue Screen of Death"

Logic Errors

- Program compiles OK
- Program executes and finishes normally
- The output of the program is wrong
  - Programmer has to find and fix errors
  - Most difficult to find
  - Must check algorithm and compare algorithm with code.
  - Often requires testing by hand.
Curator Grading

- The Curator automatically grades your program output by comparing it to the output of an instructor solution.
- The Curator responds to a submission via email.
- All programs have a raw score of 100 points.

Curator Grading

- Interpreting a line in the instructor solution output.

```plaintext
// Correct Output File: 12 lines
// [ 0]Programmer: Bill McQuain
```

- Points the line is worth
- Contents of the line of output
Curator Grading

- Points are split up evenly across tokens in a line.
  - Text between spaces
    - // Macro$oft Corporation Payroll

Curator Grading

- Most common mistakes
  - Spelling
    - valeus instead of values
  - Extra spaces
    - summed : instead of summed:
  - Missing spaces
    - GrossPay instead of Gross Pay
  - Capitalization
    - Values instead of values
Curator Grading

- Fails to compile
  - May have submitted input file or executable, submit source (.cpp) file instead
- Fails to create output file
  - Make sure output file name is spelled properly
  - Make sure input file name is spelled properly
  - Check for other run time errors using input sent by the Curator

Curator Grading

- If you fail to score 100 on the raw score
  - Don't submit again before identifying and fixing the problem
  - Test your program with the Curator's input file
- Other problems
  - Extra or missing lines
  - Never completing (we'll talk about this later)