Some Computer Language

- FORTRAN (FORmula TRANslation) - 1956
- COBOL (Common Business Oriented Language) - 1959
- BASIC - “Simple version of FORTRAN Beginner’s All-Purpose Symbolic Instruction Code” - 1964

Some Computer Language

- PASCAL - 1970 - Easy to learn
- C - based on a language called “B” - 1979
- C++ - Object-oriented C - 1979
- JAVA - 1995 - “C-like” language for the WEB
Scripting Computer Languages

- “Shell” - UNIX
- “.bat” files - MS-DOS - 1987
- Perl - 1987
- Ruby, Python, PHP

Compiled Languages

- Source code is translated by a program called a compiler into machine language, resulting in an executable file.
- Advantages:
  - Runs fast, because cost of translation only done once.
  - Executable is a “stand-alone” program.
- Disadvantages:
  - Less feedback during program development (errors only seen until after program executes).
  - Source code changes require re-compilation.
Interpreted Languages

- Source code is translated by an interpreter rather than being read.
- Advantages:
  - Translation and execution happen together, resulting in better feedback to user.
  - Results of source code changes can be seen immediately.
- Disadvantages:
  - Slower execution (up to 10x) due to translation overhead.
  - No stand-alone executable.

Ok, Back to Language Itself

- Let’s talk about the basic language constructs:
  - Assignment
    - \(a = b \times c\)
    - \(a\) is a memory location, normally denoted as a variable.
    - \([a] = [b] \times [c]\)
    - Associates a memory location and the value stored in that location
Language Constructs

- Looping
  - Does the same set of instructions over and over again
  - This way we do NOT have to write EVERY instruction
  - Can do it a SPECIFIC number of times
    - for (Start;End;Increment)
      
      ```
      { 
      Code block 
      }
      ```
  - Can do it UNTIL a specific CONDITION e.g.

```
while(CONDITION does NOT exist)
{
  Code block
}
```

- Math
  - Normal operators
    - +
    - -
Language Constructs

• / (division)
• * (multiplication)
• ^ or ** (exponent)
• Other special constructs (like % [mod])
• Uses precedence rules
  \[ 3*5^3/6+3 = ? \]
  – () first
  – Exponents
  – Multiplication and Division (left to right)
  – Addition and subtraction (left to right)

Language Constructs

• We use TONS of Parenthesis
  – Decisions
    • if(condition)
      then {block of code}
      else {other block of code}
  • Branch if….
    – bne
    – bge
    – ble
Language Constructs

• Flow Control
  – Using the decision statements for flow control
  – J commands

• Input/output
  – Used to get information in and out
  – Many, many commands
  – cin >>
  – cout <<

Language Constructs

• Not every language uses the same key words
• But, all have to have similar functions
• The difference is SYNTAX
• Let’s look at an actual example
What Marketing Asked For

http://www.uoregon.edu/~ftepfer/SchlFacilities/TireSwingTal

What the Programmers Designed

http://www.uoregon.edu/~ftepfer/SchlFacilities/TireSwingTable
What the Customer Wanted

http://www.uoregon.edu/~ftepfer/SchlFacilities/TireSwingTable.html

SW Procedure

- The Waterfall Method
  - Requirements
  - Design
  - Implementation
  - Testing
  - Maintenance

- As you can see, very similar
- Does not let you return to a previous step
Life Cycles

• The “Modified Waterfall” allows going back to previous steps
• Other Life Cycle models include:
  – Evolutionary
  – Rapid Prototype
  – Spiral
• These models recognize the need for multiple passes in development

Evolutionary Model

www.cs.uidaho.edu/~billjunk/courses/cs485/s04html/sld003.html
Rapid Prototype Model

Life Cycle Model Steps

- Requirements
  - Here we decide what the system SHOULD
  - If we are smart, we decide how we are going to design the test to prove we did it
  - No code is written
  - Only a list of requirements and a proposed schedule comes from this step

www.cs.pitt.edu/~chang/163/03life.html
Life Cycle Model Steps

• Design
  – Characterized by LOTS of pictures
  – The step where the “what” to design is turned into “how” it will be done
  – Normally give to the developers
  – Completed when the plan of “how much” “how long” are answered

Life Cycle Model Steps

• Implementation
  – This is the translation of “how” into code
  – Includes a great deal of documentation
  – Documentation is VERY important
    • If an error is found, you need to be able to trace it to the source
    • Want to be able to remember what you have done
    • Mostly notes to one’s self
    • THERE is NO SUCH THING as SELF DOCUMENTING CODE
Life Cycle Model Steps

• Testing
  – All of the final testing is done here
  – Tests include:
    • Boundary conditions
    • Branch testing
    • Normal use test
    • “Stupidity” testing
  – Alpha Site testing
  – Beta Site testing

Life Cycle Model Tests

• Can take a long time
• Involves test engineers
• Tries to find all of the bugs and eliminate them

• Support, or Maintenance
  – Nothing is perfect
  – Needs change
  – Longest step – can take years
Life Cycles

• How much work is done in each stage
  – Depends on who is doing the work, and
  – But, in general:
    • Requirements Phase  5% - 10%
    • Design Phase  5% - 10%
    • Coding Phase  10% - 20%
    • Testing Phase  20% - 30%
    • Maintenance Phase  Forever after

Costs of Errors

• Errors cost money to fix
• The later they are found, the more money it takes to fix them
• The idea is to catch errors as soon as is possible
• So, how much does it cost to find and fix errors?
Cost of errors

- By Life Cycle Stage
  - Requirements - $1
  - Design - $5 - $10
  - Implementation - $10
  - Testing - $50
  - Maintenance - $1000 – $10000