HOW DOES COMPUTER PROGRAMMING WORK?

MAGIC.
Computer Components

- Hardware
- User Programs
- Operating System

Software

- Hardware

Diagram showing the components of a computer system, with a focus on hardware, user programs, and an operating system, which are part of the software layer.
What are Programs?

- Programs provide instructions for computers
- Similar to giving directions to a person who is trying to get from point A to point B. A program may say... if starting at location A, go north 3 blocks, then go East 2 blocks.
What are Programs?

- Input
- Processing
- Output
High/Low Level Programming Languages

- Low level programming languages are languages which have instruction sets that are limited and specific to the computer hardware being run on. Programmers need to know how the hardware works to use it.

- High level programming languages provide a layer of abstraction that allows for the programmer to only have to learn a hardware independent language to write software.
Low Level Languages

• Pros
  • Can be more optimized
  • Usually smaller executables

• Cons
  • Platform dependent
  • Slower development
  • Harder to understand
  • Easier to introduce bugs

• When to use it: Need extra optimization, Need small executables (embedded systems)

• Example: Assembly (different for each OS/Architecture)
High Level Languages

• Pros
  • Faster development (libraries/etc.)
  • Easier to understand (abstraction)
  • Can be platform independent

• Cons
  • Usually larger executables
  • Abstraction layer usually adds overhead for processing resulting in slower executables

• When to use it: Need portability, need faster turn-around time for development, when programmer time is more valuable than processing time

• Examples: Java, C, C++, C#, Javascript, PHP, Perl, Python, Lisp, Scheme, R, etc.
Assembly Example (Low Level)

mov ax, cs
mov ds, ax
mov ah, 9
mov dx, offset Hello
int 21h
xor ax, ax
int 21h
Hello:
  db "Hello World!", 13, 10, "$"
Python Example (High Level)

print(“hello World!”)
Java Example (High Level)

class HelloWorld {
    static public void main( String args[] ) {
        System.out.println( "Hello World!" );
    }
}

Anatomy of a Program

Traditional Language:
- Source Code
  - Compiler
  - Compiled Code
    - Linker
    - Executable

Interpreted Language:
- Source Code
  - Interpreter
  - Byte Code
    - Virtual Machine
    - Execution
Bytecode vs. Executable

• Bytecode
  • Cross platform
  • Allows for replacement of small components without recompiling entire programs
  • Generally slower performance

• Executable
  • Runs on one platform
  • Programs generally compile down to larger executables
  • Generally faster performance
Algorithms
Algorithms

- A series of repeated instructions that solve a problem
Algorithms

- A series of repeated instructions that solve a problem

- For example:
  
  - To read a file and print its contents a program could: open a file it isn't yet, check if it has reached the end of a file, if not read a line and print its contents, repeat
  
  - To sort a list of numbers it could: compare the first two number, swap the number if necessary, then proceed to second and third numbers and repeat until all numbers are sorted
Bugs
Bugs

• All programmers run into bugs... lots and lots of bugs. Bugs are errors in a program.

• Bugs come in three different varieties:
  •
  •
  •
Bugs

• All programmers run into bugs... lots and lots of bugs. Bugs are errors in a program.

• Bugs come in three different varieties:
  • Compile time errors: (syntax/type/etc.) The rules of the language have been violated. Examples: improper spacing, missing colon, etc.
Bugs

- All programmers run into bugs... lots and lots of bugs. Bugs are errors in a program.

- Bugs come in three different varieties:
  - Compile time errors: (syntax/type/etc.) The rules of the language have been violated. Examples: improper spacing, missing colon, etc.
  - Runtime errors (crashes): Errors during execution. Example: input file may not be readable
Bugs

• All programmers run into bugs... lots and lots of bugs. Bugs are errors in a program.

• Bugs come in three different varieties:
  • Compile time errors: (syntax/type/etc.) The rules of the language have been violated. Examples: improper spacing, missing colon, etc.
  • Runtime errors (crashes): Errors during execution Example: input file may not be readable
  • Logic errors: A problem that is caused by a flawed algorithm or set of instructions used to solve a problem. Example: If you use a less than sign where you should have used a greater than sign
Pseudo-Code

- The expression of programming logic in a language independent nature.
- Good for design phase of coding
- Examples:
  ```python
  if student_grade >= 60
    print "passed"
  else
    print "failed"
  ```
Writing a Program

1. Design
2. Write Code
3. Fix Compile Time Errors
4. Run Program

Flowchart:
- Design → Write Code → Fix Compile Time Errors → Run Program
- Return arrows to Design from Write Code and Run Program.
Basic Programming Concepts

- Keywords
- Statements
- Operators
- Data Types
- Variables
- Assignment
- Constants
Keywords

- Keywords are reserved words that have special meaning in a particular programming language. These words can not be used for any other purpose (e.g. variable names). Java keywords are:

    abstract, continue, for, new, switch, assert, default, goto, package, synchronized, boolean, do, if, private, this, break, double, implements, protected, throw, byte, else, import, public, throws, case, enum, instanceof, return, transient, catch, extends, int, short, try, char, final, interface, static, void, class, finally, long, strictfp**, volatile, const*, float, native, super, while
Statements

- Programs are made up of a series of statements
- Similar to a sentence in a natural language, each statement presents a command that the interpreter understands
- The interpreter parses or reads the statement determines what is being requested and executes that statement
Basic Data Types

- Integers (Real numbers): short (16 bits), int (32 bits), long (64 bits)
- Single text character: char
- 32bit decimal numbers: float
- 64bit (more precise) decimal numbers: double
- True/False (similar to a bit): boolean
- 8 bits: byte
- Strings (are actually objects): store a series of characters
Operators

- Different data types have different operators
- Operators can be part of a statement
- Operators act on the operands around them
- Unary operators – take 1 argument
  
  e.g. setting a number to a negative value: -3
- Binary operators – take 2 arguments
  
  e.g. arithmetic operators: 1 - 3
- Ternary operators – takes 3 arguments
  
  // if x < y set largest to y, else set it to x
  largest = x < y ? y : x;
Operators on Numbers
(in order of precedence with respect to divisions)

+      Additive operator (also used for String concatenation)

-      Subtraction operator

*      Multiplication operator

/      Division operator

%      Remainder operator

( )    Dictate order of operations
Comments

- Provides:
  - Documentation
  - Clarifying what specific code is doing
  - Make code easy for the author or other programmers to understand

- Line comments start with the // characters
  
  // this is a comment

- Block comments start with /* and end with */
  
  /* this is a block comment
   * comment, it spans multiple lines */
Variables Names

• Variables are labels used by programmers for storage of data

• Variable name in java:
  • Can be long
  • Are case sensitive (Alpha != alpha)
  • Can contain letters, numbers, and underscores ( _ )
  • Must not start with a number
  • Can not be a keyword
thisVariableNameIsValid = "valid naming conventions"
this_variable_name_is_valid = "also valid"
names = "names are valid"
Names = "not the same as names"
numbers_between_0_and_9 = "another valid"
0_9_can_not_start_a_name = "invalid"
if = "keywords are not allowed"
special_chars_!_allowed = "invalid"
Constants

- A constant is a stored value that doesn't change
- Used for things that will remain constant throughout the program. (e.g. pi, conversions between metric and standard units, etc)
- When representing data that doesn't change it's often a good idea to use constants
- Generally constants are in all CAPITAL letters
Variable Assignment Statements

public static final float PI = 3.14159265;
public float radius = 5;
public float area_circle = PI * radius * radius;
Data Output

- System.out.println()
- System - http://docs.oracle.com/javase/7/docs/api/java/lang/System.html
- out.println()
  http://docs.oracle.com/javase/7/docs/api/java/io/PrintStream.html
For numbers with strings:

```java
public static String year = "2013"
System.out.print("The year is: "+ year)
```
Data Output

- For numbers with strings:

  ```java
  public static String year = "2013";
  
  // Not what we want:
  System.out.println("Next year is: " + (1 + year);
  
  // Convert string to int then add:
  System.out.println("Next year is: " + (1 + Integer.toString(year)));
  ```
Type Conversion

- In certain circumstances your data types may need to be converted to other data types (e.g. converting a string into an integer or vice versa).

- **String → int**
  
  ```java
  foo = "31337";
  int bar = Integer.parseInt(foo);
  ```

- **int → String**
  
  ```java
  Integer.toString(bar)
  ```

- **int != Integer**
  
  int is a primitive with no functions
  Integer is an object that has functions to manipulate it
Data Input

• For strings:

Scanner reader = new Scanner(System.in);
System.out.println("Enter input: ");
String input=reader.nextLine();

• For numbers:

Scanner reader = new Scanner(System.in);
System.out.println("Enter an intger");
int input=reader.nextInt();
Natural Languages

- Syntax: punctuation and spacing
- Grammar: forms well defined sentences (e.g. subject–verb-object)
- Parts of speech: nouns, verbs, adjectives
- Semantics: The meaning of the words
- Example: The quick brown fox jumped over the lazy dog.
Programming Languages

- Syntax: punctuation (e.g. parentheses, colons, spacing)
- Grammar: forms well defined statements (e.g. if statement is true then perform subsequent statement)
- Parts of speech: keywords, variables, operators, etc.
- Semantics: The meaning of the words
- Example: if a == b: print(“a is equal to b”)
Coding Style

• Just as in natural languages the style of coding matters.

• Style determines the level of readability, maintainability, and efficiency.

• Several things make up a coding style:
  • Formatting, Naming schemes, Comments, and more!