Introduction

Comp 333: Concepts of Programming Languages
Spring 2015

Instructor: Professor Schwartz

Concepts of Programming Languages

- History
- Syntax and Semantics
  - Compilers
- Language Constructs
  - Names, Binding, Scoping, Data Types
  - Expressions, Control Structures, Subprograms
- Programming Language Types
  - Imperative
  - Functional
  - Logic
  - Concurrent
  - Object Oriented
  - Scripting Languages

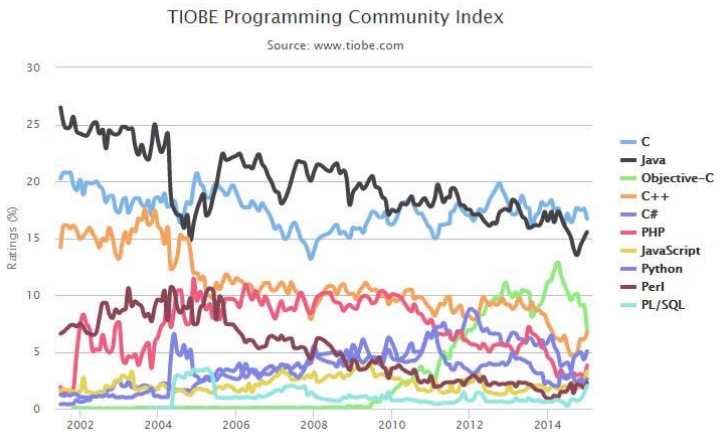
Small Group Discussion (10 minutes)

- Introduce yourself to your group
- What programming languages are you familiar with? How familiar?
- What are the advantages of learning more than one programming language?
- Make a written list of these advantages.

Programming Language Spectrum

- Imperative Languages
  - C, C++, Fortran, Java
- Functional Languages
  - Lisp, Scheme, ML
- Logic Programming Languages
  - Prolog
- Object-Oriented Languages
  - Java, C++, Smalltalk
- Scripting Languages
  - Javascript, Perl, Python
- Tiobe Programming Community Index

http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html



Why are there so many languages?

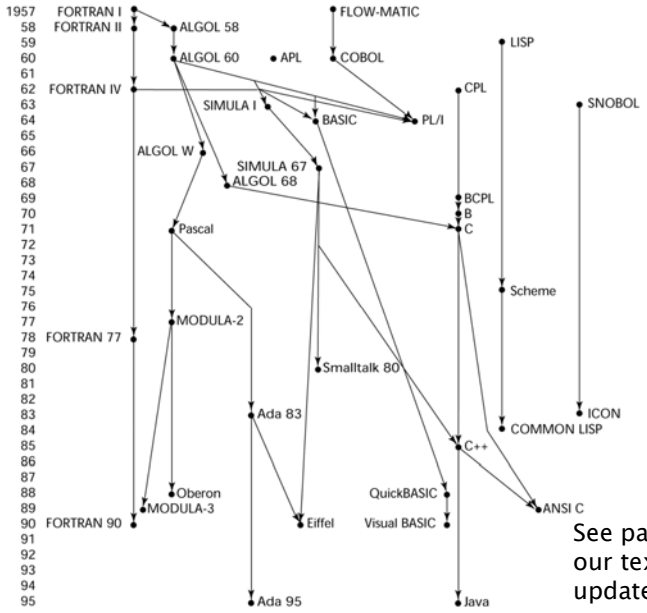
- Different program domains
  - Scientific applications (Fortran , C)
  - Business applications (Cobol)
  - Artificial Intelligence (Lisp, Prolog)
  - Systems programming (C)
  - Web programming (Javascript, Perl, PHP)
  - Embedded Systems DOD (ADA)
  - Education (Pascal)
- Complexity of modern software
  - Need for Increased Program Modularity
  - Need for Increased Reliability and Maintainability

All languages evolve over time

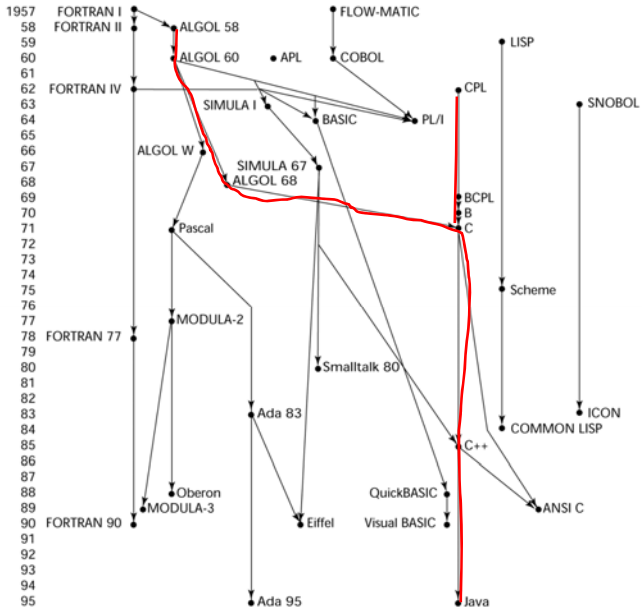
- Features are added or modified to
  - make problems easier to solve
  - make programs easier to write
  - make programs easier to understand
  - standardize language features
- New languages are created
  - usually evolved from older languages

Programming Language Features Added Over Time

- Variables: x,y,z
- Arithmetic Expressions: z = x + y
- Data types: int, double, string
- Block structure: local scope rules
- Functions and procedures
- Data structures: arrays, records, pointers
- Recursion
- Runtime Exception Handling
- Support for concurrency: threads
- Object –Oriented Language Features
  - classes, objects, inheritance, polymorphism



See page 820 of our text for an update.



What makes a language successful?

- Facilitates writing clear, concise, reliable and maintainable code
- Easy to learn
- Easy to implement (compilers, interpreters)
- Standardization (for portability)
- Good supporting tools (compilers, libraries)
- Economic Issues
  - Free, easy to install compilers and support tools
  - Legacy code makes it expensive to move to a new language (e.g. Cobol)

First Programming Languages: Assembly Languages

- Symbolic locations and opcodes
- Computation of N = I + J (Pentium 4)

```
FORMULA:    MOV     EAX,I
            ADD     EAX, J
            MOV     N, EAX
I           DD      3       ;reserve 4 bytes
J           DD      4       ;reserve 4 bytes
N           DD      0       ;reserve 4 bytes
```

FORTRAN (Imperative Programming)

- Fortran Mid 1950s
  - Developed by John Backus and his group at IBM
  - Used to perform math computations (formulas)
  - One of the first "high level" languages
  - Continued development Fortran IV, Fortran77,..
- Features
  - Variables, expressions, statements
  - Arrays
  - Iteration and conditional branching
  - Subroutines (independently compiled)
  - FORMAT for input and output

Fortran IV Fragment of a Program (See handout)

```
      DIMENSION X(52), Y(2,50), LITERL(1)
      DOUBLE PRECISION S1,S2,S3,S4,S5,T,S, BBAR
      WRITE (5,10)
10    FORMAT('0',1X,'*   *   * LINEAR REGRESSION ANALYSIS  *   *   *',//)
      WRITE (5,20)
20     FORMAT(1X,'HOW MANY PAIRS TO BE ANALYZED?'$)
      READ (5,*) N
       IF (N.GT.50) GOTO 70
      WRITE (5,30)
30     FORMAT(//1X,'Enter one pair at a time')
      WRITE (5,40)
40    FORMAT(1X,'and separate X from Y with a comma.'//)
      WRITE (5,50)
50    FORMAT(1X, 'Enter pair number one : '$)
      READ (5,*) X(1), Y(1,1)
      DO 60 I=2,N
        WRITE (5,55) I
55    FORMAT(1X,'Enter pair number',I3,' : '$)
         READ (5,*) X(I), Y(1,I)
60    CONTINUE
      GOTO 200
70    WRITE (5,80)
80    FORMAT(1X,'At present this program can only handle 50 data pairs.')
      STOP
```

FORTRAN IV Example -- Fragment

```
200         DO 210 I=1,N
            S1=S1+X(I)
            S2=S2+Y(1,I)
            S3=S3+X(I)*Y(1,I)
            S4=S4+X(I)*X(I)
            S5=S5+Y(1,I)*Y(1,I)
210           CONTINUE
          T=N*S4-S1*S1
          S=(N*S3-S1*S2)/T
           B=(S4*S2-S1*S3)/T

        WRITE( S, 260) S
260     FORMAT(//, 1X, 'SLOPE = ',     , D22.16  )
          . . .
           . . .


          END
```

Note: This simplified program does not use the second row of Y

LISP (Functional Programming)

- Lisp (1959-1960)
  - Developed by John McCarthy at IBM
  - Symbolic processing (e.g. differentiation)
  - Ancestor of Scheme
- Features
  - Symbolic processing language (e.g. list processing)
  - Built on lists, atoms, selectors and constructors
  - Dynamically allocated linked lists
  - Garbage Collection
  - Recursion
  - Functions are first class objects

Factorial Function in a Dialect of LISP

```
(define factorial
  ( lambda (n)
      (if (=  n  0)
            1
            ( *  n   (factorial ( - n 1)))
      )
  )
)
```

This function is written in Scheme

Prolog

- Logic programming language
- Makes explicit use of logic
- Very useful for problems that require searching for solutions to logic problems
- Used for automatic theorem
- Early Prolog interpreter and compiler developed in 1977 in Edinburgh

Jerry is a mouse

All mice eat cheese

Deduce: Jerry eats cheese

Prolog World

Factorial Function in Prolog

```
factorial ( 0, 1) .

factorial ( N, Result)  :-
        N > 0,
        A is N-1,
        factorial ( A, Z),
        Result is N * Z.
```

C Programming Language

- C (1972)
  - Designed by Dennis Ritchie at Bell Labs
  - Ancestor of Java, C++
- Features
  - Language for systems programming
  - C compiler was part of the UNIX operating system
  - Used in many application areas
  - Official (ANSI) description of C (1989)

See Tiobe chart. C is very long lasting! Why?


```
#include <stdio.h>  //Needed for C IO
  void strcopy ( char* s, char* t)
  {
    while( ( *s = *t)  != '\0')
    {
      s++;
      t++;
   }
  }

  int main()
  {
    char a[] = "applepie";
    char b[] = "chocolatecake";
    strcopy ( b,a);

    printf("%s\n", a);
    printf("%s\n",b);

  }
```

Very insecure code. What happens if b is set to “123”?

PASCAL

- Designed by Niklaus Wirth in 1960s.
- Simplified version of Algol 68
- Widely used as a teaching language in the 1970s and 1980s.


```
program pascalEx( input,output)
        type  intlisttype = array [ 1..99] of integer;
        var
                intlist: intlisttype;
                listlen, k, sum, average, result  : integer;
        begin
                result := 0;
                sum := 0;
                readln( listlen);
                if ( listlen > 0) and ( listlen < 100)  then
                    begin
                        for  k:= 1 to listlen do
                        begin
                                readln( intlist[k]);
                                sum := sum + intlist[k]
                        end;

                        average := sum /listlen;
                        for  k := 1 to listlen do
                                if ( intlist[k]  > average) then
                                        result := result + 1;
                        {Print result}
                        writeln('The number of values > average is ', result)
                    end
                else
                        writeln('Error – input list length is not legal')
        end.
```

Class Exercise

How does this Pascal program differ from a similar Java program?