*Continuations* provide a novel way to suspend and "re-execute" computations.

2. ML ("*M*eta *L*anguage") Strong, compile-time type checking.

Types are determined by *inference* rather than declaration.

Naturally polymorphic (one function declaration can be used with many different types).

Pattern-directed programming (you define patterns that are automatically matched during a call). Typed exceptions are provided. Abstract data types, with constructors, are included.

3. Prolog (*Pro*gramming in *Log*ic) Programs are Facts and Rules. Programmers are concerned with definition, not execution. Execution order is automatically determined.

CS 538 Spring 2008

12

CS 538 Spring 2008

4. Pizza

Extends a popular Objectoriented language, Java, to include

- Parametric polymorphism (similar to C++'s templates).
- First-class functional objects.
- Algebraic data types, including patterns.

### 5. C#

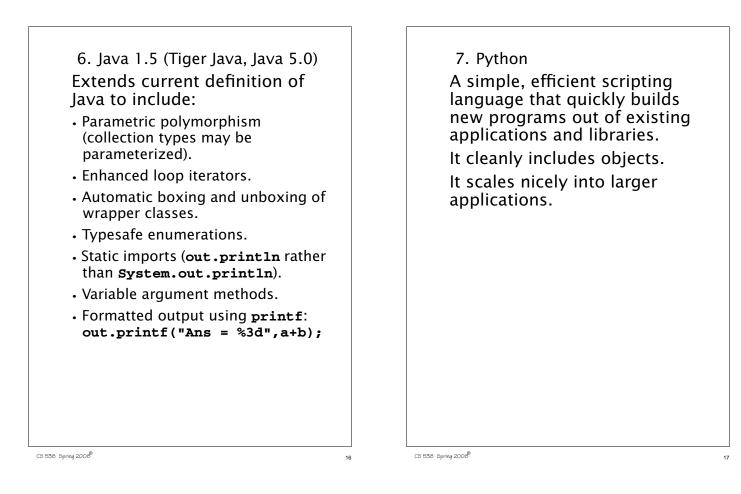
Microsoft's answer to Java. In most ways it is very similar to Java, with some C++ concepts reintroduced and some useful additions.

- Events and delegates are included to handle asynchronous actions (like keyboard or mouse actions).
- Properties allow user-defined read and write actions for fields.
- Indexers allow objects other than arrays to be indexed.
- Collection classes may be directly enumerated:
- foreach (int i in array) ...
- Structs and classes co-exist and may be inter-converted (boxed and unboxed).
- Enumerations, operator overloading and rectangular arrays are provided.
- Reference, out and variable-length parameter lists are allowed.

14

CS 538 Sprina 2008

13



## Evolution of Programming Languages

In the beginning, ...

programs were written in absolute machine code—a sequence of bits that encode machine instructions.

#### Example:

34020005 0000000c 3c011001 ac220000

### This form of programming is

- Very detailed
- Very tedious
- Very error-prone
- Very machine specific

# Symbolic Assemblers

Allow use of symbols for operation codes and labels.

### Example:

li \$v0,5 syscall sw \$v0,a

Far more readable, but still very detailed, tedious and machine-specific.

Types are machine types.

Control structures are conditional branches.

Subprograms are blocks of code called via a "subroutine branch" instruction.

All labels are global.

CS 538 Sprina 2008

# Fortran (Formula Translator)

## Example:

do 10 i=1,100 10 a(i)=0

Developed in the mid-50s.

### A major step forward:

- Programming became more "problem oriented" and less "machine oriented."
- Notions of control structures (ifs and do loops) were introduced.
- Subprograms, calls, and parameters were made available.
- Notions of machine independence were introduced.
- Has evolved into many new variants, including Fortran 77, Fortran 90 and HPF (High Performance Fortran).

CS 538 Spring 2008

## **Cobol (***Co*mmon *B*usiness *O*riented *L*anguage**)**

### Example:

multiply i by 3 giving j.
move j to k.
write line1 after advancing
1 lines.

Developed in the early 60s.

The first widely-standardized programming language.

Once dominant in the business world; still important.

Wordy in structure; designed for non-scientific users.

Raised the issue of who should program and how important readability and maintainability are.

CS 538 Spring 2008

20

Algol 60 (Algorithmic Language)

## Example:

```
real procedure cheb(x,n);
value x,n;
real x; integer n;
cheb :=
    if n = 0 then 1
    else if n = 1 then x
    else 2 × x ×
    cheb(x,n-1)-cheb(x,n-2);
```

Developed about 1960.

A direct precursor of Pascal, C, C++ and Java.

Introduced many ideas now in wide use:

- Blocks with local declarations and scopes.
- Nested declarations and control structures.

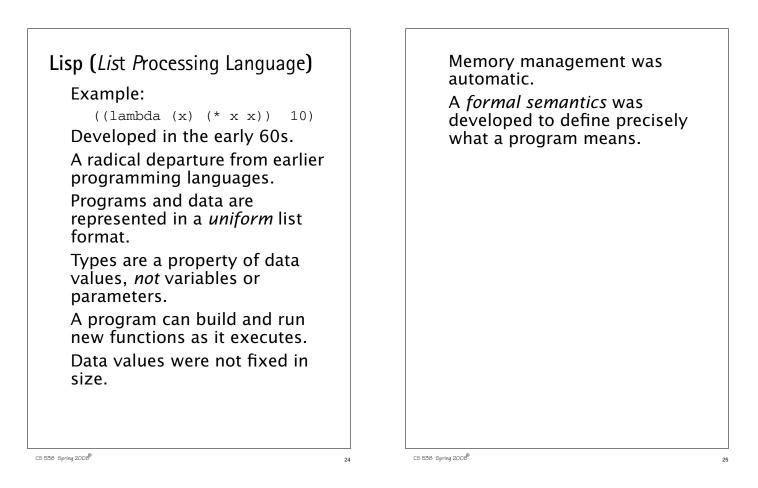
- Parameter passing
- Automatic recursion.

#### But,

- I/O wasn't standardized.
- IBM promoted Fortran and PL/I.

21

CS 538 Spring 2008



# Simula 67 (Simulation Algol)

### Example:

Class Rectangle (Width, Height); Real Width, Height; Boolean Procedure IsSquare; IsSquare := Width=Height; End of Rectangle;

Developed about 1967.

Introduced the notion of a class (for simulation purposes).

Included *objects*, a garbage collector, and notions of extending a class.

C++ was originally C with classes (as Simula was Algol with classes).

## C and C++

C was developed in the early 70's; C++ in the mid-80s.

These languages have a concise, expressive syntax; they generate high quality code sufficient for performance-critical applications.

C, along with Unix, proved the viability of *platformindependent* languages and applications.

C and C++ allow programmers a great deal of freedom in bending and breaking rules.

Raises the issue of whether one language can span both novice and expert programmers.

26

#### Interesting issue—if most statements and expressions are meaningful, can errors be readily detected?

```
if (a=b)
a=0;
else a = 1;
```

## Java

Developed in the late 90s.

Cleaner object-oriented language than C++.

Introduced notions of dynamic loading of class definitions across the Web.

Much stronger emphasis on secure execution and detection of run-time errors.

Extended notions of platform independence to system independence.

CS 538 Spring 2008

28

CS 538 Spring 2008