High-Level Languages
Languages

- Assembly vs Machine Code
- Assembly vs high-level language
Why Learn New Language

- availability
- special features
- porting
- maintenance
- more tools
- required for job
- cost
Why Design a Language?

- **Special Need**
  - Assembly, FORTRAN, COBOL, etc.
  - Cg – C for graphics

- **Commercialism**
  - FORTRAN by IBM

- **Proselytism**
  - Pascal by Wirth for structured programming

- **Creativity**

- **Research**
  - May escape lab - Smalltalk

- **Standards**
Purposes of a Language?

- General-purpose
- Special-purpose
- As development environments
Purposes of a Language?

- General-purpose
  - Production-oriented
  - Science
    - FORTRAN
    - C
    - C++
  - Business
    - COBOL
- Special-purpose
- As development environments
Purposes of a Language?

- **General-purpose**
- **Special-purpose**
  - to solve logic problems
    - prolog
  - evolution of previous language
    - ALGOL -> Pascal -> Modula
    - b -> c -> c++
  - To teach programming
    - BASIC
    - Pascal
  - Portability
    - Java
- **As development environments**
Purposes of a Language?

- General-purpose
- Special-purpose
- As development environments
  - suite of tools for development compiler, design tools, debugger, editor)
    - Smalltalk
  - Later suites designed around language
    - Borland pascal, c, c++
    - Microsoft c++
    - Visual Studio
    - Java Tools
Language Evolution

- Hardware and OS
- Assembly
- High-level language
- Applications
- Standardization
Language Evolution

- Hardware and OS
  - Instruction sets for doing certain tasks
  - Tied to specific machine
- Assembly
- High-level language
- Applications
- Standardization
Language Evolution

- Hardware and OS
- Assembly
  - As a mnemonic for machine code
    - increase level of abstraction
  - Assumed to be done by scientists.
- High-level language
- Applications
- Standardization
Language Evolution

- Hardware and OS
- Assembly
- High-level language
  - Another level of abstraction
  - Don't worry about memory locations and how to set them
    - \( a = b + c \)
  - Design algorithms for doing jobs (computer programming as a new job)
    - FORTRAN, BASIC, ALGOL, COBOL
- Applications
- Standardization
Language Evolution

- Hardware and OS
- Assembly
- High-level language
- Applications
  - Business
  - Scientific computing
  - Parallel computing
  - AI
  - Language Queries
- Standardization
Language Evolution

- Hardware and OS
- Assembly
- High-level language
- Applications
- Standardization
  - Ada
Language Evolution

- Growing abstraction
  - Higher-level constructs
  - More powerful and more built-in functions
- Growing facilities within language for defining abstraction
  - Abstract-data structures
- Growing facilities for program structure
  - Separate modules
Language Evolution

- Older languages still in use are those that have evolved with newer techniques.
- Landmark languages tend to be general-purpose, but may be more convenient for limited class of problems.
Landmark Languages
1936 Turing Machine

- Church's Thesis proved all computable functions are capable of being computed with Boolean logic, *i.e.* a Turing Machine.
- Exceedingly simple
- Weak built-ins, simple I/O
- Unsuitable as a programming language
Landmark Languages
1940's Machine Code

- Binary or Octal machine code
- Used to directly program a particular machine
- Powerful but difficult to use
Landmark Languages
1950's Assembly

- Symbolic language for machine code
- Easier to read and program than machine code

```assembly
/title Hello World Program (hello.asm)
; This program displays "Hello, World!"
dosseg
.model small
.stack 100h
.data
hello_message db 'Hello, World!',0dh,0ah,'$'
code
main proc
    mov ax,@data
    mov ds,ax
    mov ah,9
    mov dx,offset hello_message
    int 21h
    mov ax,4c00h
    int 21h
main endp
end main
```
Landmark Languages
1956 FORTRAN

- **FORmula TRANslator**
- Major factor in IBM's growth in 50s & 60s
- Most suited to mathematical and scientific problems
- Efficient so can compete with assembly
- Flexible enough for other uses.
- Handles numbers well
- Not a free format
- Implicit variables
  - i-n ints, others floats
- Introduced:
  - types, subprograms, formatted I/O

Program Hello

```fortran
DO while (.NOT. DONE)
  write(*,10)
END DO
```

```
10 format('Hello, world.')
END
```
Landmark Languages
1959 COBOL

- **CO**mmon **B**usiness-Oriented **L**anguage
- Designed for business applications
  - List and file processing
  - Good for report writing
  - Verbose (or non-cryptic) code.
- Still in use
  - Lots of COBOL programmers
  - Lots of code to convert
- DoD language
- *Designed by users not implementors*
- Introduced:
  - data-structure defs (record)
ALGO: ALGOOrithmic Language 60

- Designed to be universally-applicable
- Alternative to FORTRAN & IBM
- Mostly used in Europe
- Precursor of Pascal
- Designed by users
- Difficult to implement

**Introduced:**
- Block structure for scope, if-then-else, recursion

In Algol 60 there was no output defined so Hello, World was somewhat impossible:-) But given a suitable library of IO procedures:

```plaintext
printstring(`Hello, World')
```
Landmark Languages
1960 LISP

- LIST Processing
- Based on symbol manipulation instead of mathematical
- Strongly functional
- Used for AI
- Data-types are lists and atoms
- A program is a function defined in terms of other functions.

```
(DEFUN HELLO-WORLD ()
  (PRINT (LIST 'HELLO 'WORLD)))
```
Landmark Languages
1962 APL

- A Programming Language
- Very high-level
- Basic structure is the array
- Powerful operators for arrays
- Very compact programs

'HELLO WORLD'

or with a variable

a←'HELLO WORLD'

a
Landmark Languages
1965 BASIC

- Beginners **All-Purpose Symbolic Instruction Code**
- First language designed specifically for teaching the programming process
- First interactive environment (interpreted)

10 print “Hello World!”
Landmark Languages
PL/I

- Programming Language One
- Included all current high-level features
- Very complex language
- Introduced: interrupt or exception handling

HELLO:  PROCEDURE OPTIONS (MAIN);

    /* A PROGRAM TO OUTPUT HELLO WORLD */
    FLAG = 0;

LOOP:     DO WHILE (FLAG = 0);
            PUT SKIP DATA('HELLO WORLD!');
        END LOOP;
END HELLO;
Landmark Languages
1967 Simula

- **SIMUlation LAngauges.**
- Designed to describe systems and their simulation
- But much more powerful
- An extension of ALGOL 60
- First object-oriented language
- Introduced:
  - classes, encapsulating data structure and function to operate on them.

Begin
While 1=1 do begin
    outtext("Hello World!");
    Outimage;
End;
End;
Landmark Languages
1968 ALGOL 68

- **ALGO**rithmic **L**anguage 68
- **Strong orthogonality**
  - Small number of constructs that can be combined smoothly to build more powerful constructs
    - + for numbers should also apply to arrays
- **Had some influence on C**
  - long
  - union
  - Type constructor

```plaintext
print("Hello World");
```
Landmark Languages
1969 Pascal

- Simple language including most existing good features
- Based on ALGOL
- Teachability very important
- Designed to teach “good”, ie structured, programming

Program Hello (Input, Output);

Begin
  Writeln ('Hello World!');
End.
Landmark Languages
1972 Prolog

- **Pro**rogrammation en **logique** (french for programming in logic)
- Designed for AI
- Program is description of desired solution
- Declarative language

```prolog
% HELLO WORLD. Works with Sbp (prolog)
hello :-
printstring("HELLO WORLD!!!!").
printstring([]).
printstring([H|T]) :- put(H), printstring(T).
```
Landmark Languages
1972 Smalltalk

- Strictly Object-Oriented
- Generalized class control from Simula

Transcript show:'Hello World';cr
High-level constructs but still have access directly to the underlying machine (memory, etc.) for systems programming

Unix written in C

AT&T Bell Labs, based on B

1985 C++, adds object oriented programming to C.

```c
#include <stdio.h>
main()
{
    printf("Hello World!\n");
}
```
Landmark Languages
1982 Modula-2

- Extension to Pascal for system programming
- Modules, abstract data-types (classes)
- Coroutines, typed procedures, all control structures have terminating keyword

MODULE hello;
FROM InOut IMPORT writestring, writeln;
begin
  WriteString("Hello, world!");
  Writeln;
end hello.
Landmark Languages

Ada

- 73-74 US government spending $3 billion/year on software, ½ embedded.
- Armed Forces propose single embedded language standard
- Apr 75 strawman requirements
- Aug 75 revised to woodenman
- Jan 76 tinman was created
- Evaluated 23 languages against this, none worked
- Jan 77 determined that Pascal, ALGOL 68 and PL/I were good starting point.
Landmark Languages
Ada

- Jan 77 Ironman - ideal spec
- RFP & of 17, 4 contracts awarded
  - red, green, blue, yellow
- Red & Green finalists of public review
- Jun 78 - steelman - final requirements
- May 79 - Green (Cii Honeywell Bull) wins
- Called DoD-1 & in May 1979 named Ada
  - Augusta Ada Byron, Countess of Lovelace
- Dec 10 1980 Military Standard 1815
- Jan 83 1815a & Feb 83 ANSI standard
- Apr 83 Ada/Ed implementation validated
Landmark Languages
Ada

- **Packages** -
  - data types, data objects, and procedures encapsulation

- **Generics**
  - procedures that don't require specified type

- **Exception handling**
  - What happens when run-time errors occur

```ada
with Text_Io; use Text_Io;
procedure hello is
begin
   put ("Hello world!");
end hello;
```
Landmark Languages
Java

- Designed for portability

class HelloWorld {
    public static void main (String args[]) {
        for (;;) {
            System.out.print("Hello World ");
        }
    }
}
How many languages

- [website](http://people.ku.edu/~nkinners/LangList/Extras/langlist.htm) - 2500 languages
- [website](http://www.99-bottles-of-beer.net/) - same program in 1500 languages
  - Check out zombie and Cow
- [website](http://www.roesler-ac.de/wolfram/hello.htm) - hello world in 441 languages
- Check out the esoteric language hub
  - [website](http://hub.webring.org/hub/esolang)
Generations of Languages

- 1\textsuperscript{st} machine code
- 2\textsuperscript{nd} assembly code
- 3\textsuperscript{rd} high-level languages
- 4\textsuperscript{th} application-generation for non programmers
- 5\textsuperscript{th} logic-oriented languages
Kinds of Languages - 1

**Imperative**
- Specifies how a computation is to be done.
- Examples: C, C++, C#, Fortran, Java

**Declarative**
- Specifies what computation is to be done.
- Examples: Haskell, ML, Prolog

**von Neumann**
- One whose computational model is based on the von Neumann architecture.
- Basic means of computation is through the modification of variables (computing via side effects).
- Statements influence subsequent computations by changing the value of memory.
- Examples: C, C++, C#, Fortran, Java
Kinds of Languages - 2

Object-oriented
- Program consists of interacting objects.
- Each object has its own internal state and executable functions (methods) to manage that state.
- Object-oriented programming is based on encapsulation, modularity, polymorphism, and inheritance.
- Examples: C++, C#, Java, OCaml, Simula 67, Smalltalk

Scripting
- An interpreted language with high-level operators for "gluing together" computations.
- Examples: AWK, Perl, PHP, Python, Ruby

Functional
- One whose computational model is based on the recursive definition of functions (lambda calculus).
- Examples: Haskell, Lisp, ML
Kinds of Languages - 3

Parallel
- One that allows a computation to run concurrently on multiple processors.
- Examples
  - Libraries: POSIX threads, MPI
  - Languages: Ada, Cilk, OpenCL, Chapel, X10
  - Architecture: CUDA (parallel programming architecture for GPUs)

Domain specific
- Many areas have special-purpose languages to facilitate the creation of applications.
- Examples
  - YACC for creating parsers
  - LEX for creating lexical analyzers
  - MATLAB for numerical computations
  - SQL for database applications

Markup
- Not programming languages in the sense of being Turing complete, but widely used for document preparation.
- Examples: HTML, XHTML, XML
Fundamental Elements of PL

- **Programming model**
  - The programming model is the model of computation encapsulated into the programming language.
    - For example, C is an imperative language, designed around the von Neumann model of computation.

- **Program structure**
  - A program typically consists of one or more translation units stored in files.
    - In C, a translation unit is a sequence of function definitions and declarations.

- **Character set and lexical conventions**
  - Source and target character sets may be different.
    - The character set of C source programs is contained within seven-bit ASCII.
  - A token is a meaningful sequence of characters in a source program.
    - C has six classes of tokens: identifiers, keywords, constants, string literals, operators, and separators.
Fundamental Elements of PL

- **Names, scopes, bindings, and lifetimes**
  - Names (often called identifiers) have a specified lexical structure.
    - In C identifiers are sequences of letters (here, underscore is considered a letter) and digits. The first character of an identifier must be a letter. At least the first 31 characters in an identifier are significant.
  - The scope of a name is the region of the program in which it is known (visible).
    - A binding is an association between two things such as between a variable and its type or between a symbol and the operation it represents. The time at which this association is determined is called the binding time. Bindings can take place at various times ranging from language design time to run time.
  - The lifetime of a variable is the time during which the variable is bound to a specific memory location.

- **Data types and operators**
  - A data type defines a set of data values and the operations allowed on those values.
    - C has a small number of basic types, including char, int, double, float, enum, void.
    - C has a potentially infinite number of recursively defined derived types such as arrays of objects of some type, functions returning objects of some type, pointers to objects of some type, structures containing a sequence of objects of various types, and unions containing any one of several objects of various types.
    - C has a rich set of arithmetic, relational, logical, and assignment operators.
Fundamental Elements of PL

- **Expressions and assignment statements**
  - Expressions are the primary means for specifying computations in a programming language.
  - Assignment statements are basic constructs in imperative programming languages. Assignment statements allow the programmer to dynamically change the bindings of values to variables.

- **Control flow**
  - Flow of control refers to the sequence in which the operations specified in a program are executed at run time. There are flow-of-control issues at many levels such as flow of control within expressions, among statements, and among program units. Most programming languages have control statements and other control structures for controlling the flow of control within a program.
    - C has a variety of flow-of-control constructs such as blocks and control statements such as if-else, switch, while, for, do-while, break, continue and goto.
Fundamental Elements of PL

- **Functions and process abstraction**
  - Perhaps the most important building blocks of programs.
  - Often called procedures, subroutines, or subprograms.
  - Break large computing tasks into smaller ones and facilitate code reuse.

- **Data abstraction and object orientation**
  - Data abstraction in the form of abstract data types was introduced into programming languages after process abstraction. The programming language Simula67 was instrumental in motivating constructs for supporting object-oriented programming in modern programming languages such as C++, C#, and Java.
**Fundamental Elements of PL**

- **Concurrency**
  - Concurrent execution of programs has assumed much more importance with the widespread use of multi-core and many-core processors.
  - Concurrency in software execution can occur many levels of granularity: instruction, statement, subprogram, and program.
  - Concurrency can be achieved with libraries (like MPI for Fortran, pthreads for C) or with direct language support (as in Cilk, X10).
  - However, effective exploitation of concurrency is still an open research area in software.

- **Exception and event handling**
  - Many languages have facilities for reacting to run-time error conditions. C++ has the try-catch construct to catch exceptions raised by the throw statement.
  - Event handling is like exception handling in that an event handler is called by the occurrence of an event. Implementing reactions to user interactions with GUI components is a common application of event handling.
Language Characteristics

- **Compiled/interpreted**
  - many languages can be either

- **Data types supported**
  - Text - LISP, SNOBOL, Perl
  - Numbers - FORTRAN
  - Both

- **Extensible**
  - New elements create a new dialect - Pascal
  - Are new elements just required by the app
    - FOURTH, smalltalk, lisp, scheme
Language Characteristics

- **Structured/unstructured**
  - Is structured programming allowed, encouraged, disallowed, ....
    - Dijkstra - program made of blocks with single point of entry and exit
    - Subunits
    - GOTOs

- **Algorithmic - or imperative**
  - Problems solved by defining series of steps that result in a solution
    - FORTRAN, BASIC, Pascal, COBOL, C
Language Characteristics

- **Functional or applicative**
  - Treat solutions as an aggregate application of mathematical functions - logic programming
  - LISP, APL, SNOBOL

- **Object-Oriented**
  - As extensions
    - C -> C++, Pascal -- Modula 2
  - Fundamentally OO
    - SIMULA, Smalltalk
Language Characteristics

- **Special purpose/general purpose**
  - Do you determine this based on design, use, or potential use?
  - All-Purpose
    - Commercial – COBOL
    - Scientific – FORTRAN
    - System development – C
    - String manipulation – SNOBOL
  - Special-Purpose
    - Command – JCL, UNIX
    - Editors – Scribe, TeX
Language Characteristics

- **Query Languages (4GLs)**
  - Special-purpose to allow abstract (high-level) data specification and retrieval from structured data files (databases)
    - SQL (Structure Query Language)
    - QBE (Query By Example)

- **Environment used in**
  - Batch
  - Interactive
  - Real-time

- **Compiled/Interpreted**
  - Language feature, or implementation?
Language Characteristics

- **How procedural**
  - Procedural languages describe solution via an algorithm (prescriptive)
  - Nonprocedural describe the solution and system determines method (declarative)
Language Application

- Business
- Science/Engineering
- Real Time Apps
- Embedded systems
- Simulation
- Expert systems
- Teaching
- General

- COBOL, Query languages
- FORTRAN, Pascal, C, C++
- ADA, Modula-2, C, Assembly
- ADA
- Smalltalk
- LISP, PROLOG, M1
- Pascal, BASIC, LOGO
- C, Pascal