Brief History of Languages

The Evolution Of Computer Programming Languages

- Hex
- Assembler
- C
- Fortran
- C++
- Java
- Ruby
Long Long Time Ago

Build it

Program it!!
Design Leap

• Fixed program computer:
  • Designed to perform a single task

• Von Neumann architecture: Stored-Program
  • Flexibility to reprogram a machine
Early days

• Early realization of the Von Neumann architecture: ENIAC

• Programming by toggling switches and cables: Weeks

• Used in the Manhattan project
Evolution: Machine Code

- Early programmers were limited by
  - Expressivity of Machine Code: Hex or Binary
  - High Error Rate
  - Slow Development and Insanity !!!

Punch Card: http://commons.wikimedia.org
Machine Code to Assembly Language

• Machine code represents CPU instructions
  - 1010101 [0x55]
  - 1000100100001110101 [0x89 0xe5]
  - 1000001111011100011100 [0x83 0xec 0x3c]

• Hard for humans to parse

• **First Evolution:**
  - Translate instructions to mnemonics

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x55</td>
<td>push %ebp</td>
</tr>
<tr>
<td>0x89 0xe5</td>
<td>mov %esp,%ebp</td>
</tr>
<tr>
<td>0x83 0xec 0x3c</td>
<td>sub $0x3c,%esp</td>
</tr>
</tbody>
</table>
Assembly

- **Evolution**: Use computers to generate machine code
- No need to worry about variable length instructions
- **Assembler**: Generate machine code from assembly languages.
- Still in use: NASM, C asm

Example:

Intel x86-32 machine code of C `main()`
Why not Assembly?

- **Limitations:**
  - Machine dependent
  - Not portable
  - **No error checking**
  - Hard to read and tedious to write
  - Even simple tasks required 100’s of lines of code
  - Very high barrier to learn, not suitable for scientists
Why not Assembly

```c
qsort( a, lo, hi ) int a[], hi, lo;
    int h, w, p, t;
    if (lo < hi) {
        w = lo;
        h = hi;
        p = a[hi];
        do {
            while ((w < h) && (a[w] <= p))
                w = w+1;
            while ((h > w) && (a[h] >= p))
                h = h-1;
            if (w < h) {
                t = a[w];
                a[w] = a[h];
                a[h] = t;
            }
        } while (w < h);
        t = a[w];
        a[w] = a[hi];
        a[hi] = t;
        qsort( a, lo, w-1 );
        qsort( a, w+1, hi );
    }
```
Higher-Level Languages

• Assembly is still very close to the hardware, hard to master and write extensible programs

• Requirements:
  
  • **Data and Control flow abstractions:** Loops, Conditionals etc
  
  • **Extensible and Portable:** Machine Independent
  
  • Low barrier to entry: **Human Friendly Syntax**

  • Take advantage of computation power to generate machine code using **Compilers and Interpreters**
What is a Programming Language?

Star Stories and other languages, including the major justifications that the test led to his own. This is usually prepared by the infection of the Sinai to the back and the Star Destroyers in the parliament, by the speed of these books and the revival of environmental problems of their new Arab states of the Arctic as a more and the Vertex cover problem. BBCs list of the attention that the ICCs powers should have been due to CFCs used by the program is a day of St Cyprian, his party and to India to an entire city, and in space was a second home for his museum, which had served and in it to put any of the computer that could then take the next in the remaining rights to the population of this speed when the others in a single by the Council that the football is a goal is for children of those was the first US Festival for the championship in 1991, with the left and a sequence of the story, of which reached the stage of his ashes to a god, and the ‘astern Europe, has the other that the other where he saw him that the former GDR might cover This major human gut ur which was the largest city of the two countries. The British and the French, and was not the first time, and was not the case in the history of the war, Hong Kong was returned to the U.S. Rather, such a declaration of war, and in the United States President Bill Clinton was elected to the United States. In the Yuan dynasty. One of the United States, and the United Provinces at the time of the Soviet government.

Source: Mason, Joshua; Small, Sam; Monrose, Fabian; and MacManus, Greg. English Shellcode. CCS 2009

Is this a language?

It will execute !!!
What is a Programming Language?

- Java?
- HTML?
- Javascript?
- LaTeX?

A programming language is a formal language that is both
- **universal** (any computable function can be defined)
- **implementable** (on existing hardware platforms).

• Is Turing Complete
  • Simulate any turning machine

• Reality:
  • Simulate turning machine with space constraints
Language Design Tradeoffs

**Declarative**
- Ease of expressing the algorithm
- Close to the user
- Shallow learning curve: Problem Domain specific

**Imperative**
- Provide base primitives to build the algorithm
- Close to the machine: Mostly Efficient
- Potentially steep learning curve

“do what I mean”

“do exactly what I say”
High Level Language History

1960
- Algol
  Backus, Bauer et al
  Specify Algorithms
  Design from First Principles

1958
- LISP
  Mac Carthy
  Symbolic Computing
  Artificial Intelligence

1954
- FORTRAN
  John Backus (IBM)
  Data Processing
  First real alternative to assembly
  Scientific Computing

15
- Simple Syntax: Almost English like
- Still In Use
Language Spectrum

**Declarative**
Close to the user:
Steps needed to achieve it

- Functional
  - (Ex: LISP/Scheme, ML, Haskell)

- Logic and constraint-based
  - (Ex: Prolog)

- Dataflow
  - (Ex: Id, Val)

**Imperative**
Close to the machine:
How to perform a task

- Procedural / Von Neumann
  - (Ex: Fortran, Pascal, C)

- Object-Oriented
  - (Ex: Smalltalk, Eiffel, C++, Java)

- Scripting
  - (Ex: Shell, TCL, Perl, Python)
Procedural Languages

- Progression in programming languages from assembly
- Model how computers work e.g. Load Store
- Most popular language family: C, Fortran

Modeled directly on **manipulating data** (variables) usually simple **typed data**.

A procedural program defines a sequential computation of statements that directly manipulates data in memory.

Abstractions of execution units is achieved via subroutines
Object Oriented

- Next-Generation of Languages: Distributed Model of computation
- Modeled as interactions between objects that collaborate with each other
- Complex problems can be decomposed and the interactions specified as procedural code

Imperative

Close to the machine: How to perform a task

Procedural / Von Neumann
(Ex: Fortran, Pascal, C)

Object-Oriented
(Ex: Smalltalk, Eiffel, C++, Java)

Scripting
(Ex: Shell, TCL, Perl, Python)
Scripting Languages:

Fuzzy category of high-level languages that focus heavily on developer productivity (“rapid development”).

Often used for integration of components (“glue languages”), more recently for web development.

Traditionally imperative model, but there is a trend to include object-oriented and functional design elements.

Imperative

Close to the machine: How to perform a task

Procedural / Von Neumann
(Ex: Fortran, Pascal, C)

Object-Oriented
(Ex: Smalltalk, Eiffel, C++, Java)

Scripting
(Ex: Shell, TCL, Perl, Python)
**Functional Languages**: Mathematics-inspired model: program defined in terms of mathematical functions (equivalences).

A Program is a **series of functions** with **inputs and outputs**. The computer’s job is to **compute the result of applying the program (a function) to the input**.

There is **no concept of memory**: functions simply map values onto other values.

There is **no concept of state**: functions are only defined in terms of their arguments and other functions.

How this is done is not specified in the program. **Control flow is implicit** and based on **recursion**.
Language Spectrum

Declarative

Close to the user: Steps needed to achieve it

Functional
(Ex: LISP/Scheme, ML, Haskell)

Logic and constraint-based
(Ex: Prolog)

Dataflow
(Ex: Id, Val)

Logic Languages:

Inspired by propositional logic. Program is defined in terms of

facts (the “knowledge base”),

rules (implications, “if X then also Y”), and a

goal (query, “is Y true?”, “what makes Y true?”).

A program is a series of logical rules using facts to obtain a given goal.

The computer’s job is to construct a proof based on the given axioms (facts + rules).
Design Considerations

What are the primary use cases?

Communicate ideas.
- Programs are read more often than written.
- Maintenance costs.

Exactly specify algorithms.
- Succinct and precise.
- No ambiguity.

Create useful programs.
- Development must be economically viable.

Readability

Expressivity

Writability

Reliability
Expressivity

Quick sort in LISP

(we will discuss LISP in detail later in the semester)

Quick sort in C

```c
qsort(a, lo, hi) int a[], hi, lo;
{
    int h, w, p, t;
    if (lo < hi) {
        w = lo;
        h = hi;
        p = a[hi];
        do {
            while ((w < h) && (a[w] <= p))
                w = w+1;
            while ((h > w) && (a[h] >= p))
                h = h-1;
            if (w < h) {
                t = a[w];
                a[w] = a[h];
                a[h] = t;
            }
        } while (w < h);
        t = a[w];
        a[w] = a[hi];
        a[hi] = t;
        qsort(a, lo, w-1);
        qsort(a, w+1, hi);
    }
}
```

(we will discuss LISP in detail later in the semester)
Expressivity

Quicksort in LISP

(defun quicksort (lis)
  (if (null lis) nil
    (let* ((x (car lis)) (r (cdr lis))
           (fn (lambda (a) (< a x)))
           (append (quicksort (remove-if-not fn r)) (list x)
                   (quicksort (remove-if fn r))))))

Works on Lists as the abstraction

No operators + is A FUNCTION

Evaluation is recursive

No strong typing: Can use any datatype

Quicksort in C

qsort(a, lo, hi) int a[], hi, lo;
{
  int h, w, p, t;
  if (lo < hi) {
    w = lo;
    h = hi;
    p = a[hi];
    do {
      while ((w < h) && (a[w] <= p))
        w = w+1;
      while ((h > w) && (a[h] >= p))
        h = h-1;
      if (w < h) {
        t  =  a[w] ;
        a[w] = a[h];
        a[h] = t;
      }
    } while (w < h);
    t = a[w];
    a[w] = a[hi];
    a[hi] = t;
    qsort(a, lo, w-1);
    qsort(a, w+1, hi);
  }
}
Writability Factors

Abstraction
- Focus on defining concepts once

Reusability
- **DRY principle**: “don’t repeat yourself”
- Encapsulate and provide interface
- Generic programming: Design Patterns

Quality of development tools.
- **Efficiency** of compiler-generated code.
- Rapid Prototyping
- Leniency of compiler / language system.
- Turnaround time of *edit-compile-test* cycle.

Documentation.
- Availability and **quality**.

LISP: Concentrate on the inputs and outputs to solve the problem

C++/Java: Object oriented, define objects reuse data and interface

Python: Large library, Simple syntax and easy to prototype.

gcc: some warnings not used in Linux due to excessive false positives.

Java: javadoc support ensures standardized, indexable documentation.
Summary

• **History.**
  - Started with a movement away from machine code and higher abstraction
  - Simultaneously “natural” languages were developed

• **Programming Language Spectrum.**
  - Performance vs Abstraction
  - Different paradigms defined often with reuse of themes

• **Categories.**
  - **Declarative**: what to do.
    - Functional, logic-based, dataflow.
  - **Imperative**: how to do it.
    - Procedural, object-oriented, scripting.