CHAPTER 1: INTRODUCTION

CMPSC 460 – Principles of Programming Languages

Introduction

- What is a programming Language?
  - Syntax
  - Semantics
  - Implementation

ADT

- How do you represent an integer?

Binary 10011001

Decimal:	153	−25	−102	−103	99
Representation:	Unsigned	Sign and magnitude	One's complement	Two's complement	Binary-coded decimal

Introduction

- There are very many, very different languages:
  - List of programming languages
  - List of Programming Languages in Alphabetical Order
  - TIOBE Index

Why are there so many programming languages?

- Evolution
- Personal Preference
- Special Purpose
  - Orientation toward special problem
  - Orientation toward special hardware
- Socio-Economic Factors

Why are there so many programming languages?

- Easy to learn
- Easy to express things, easy to use once fluent
- Easy to implement
- Standardization
- Open source and Efficient Compilers

Classification of Programming Languages

- Programming Paradigms
  - Imperative Languages
  - Functional Languages
  - Logic Languages

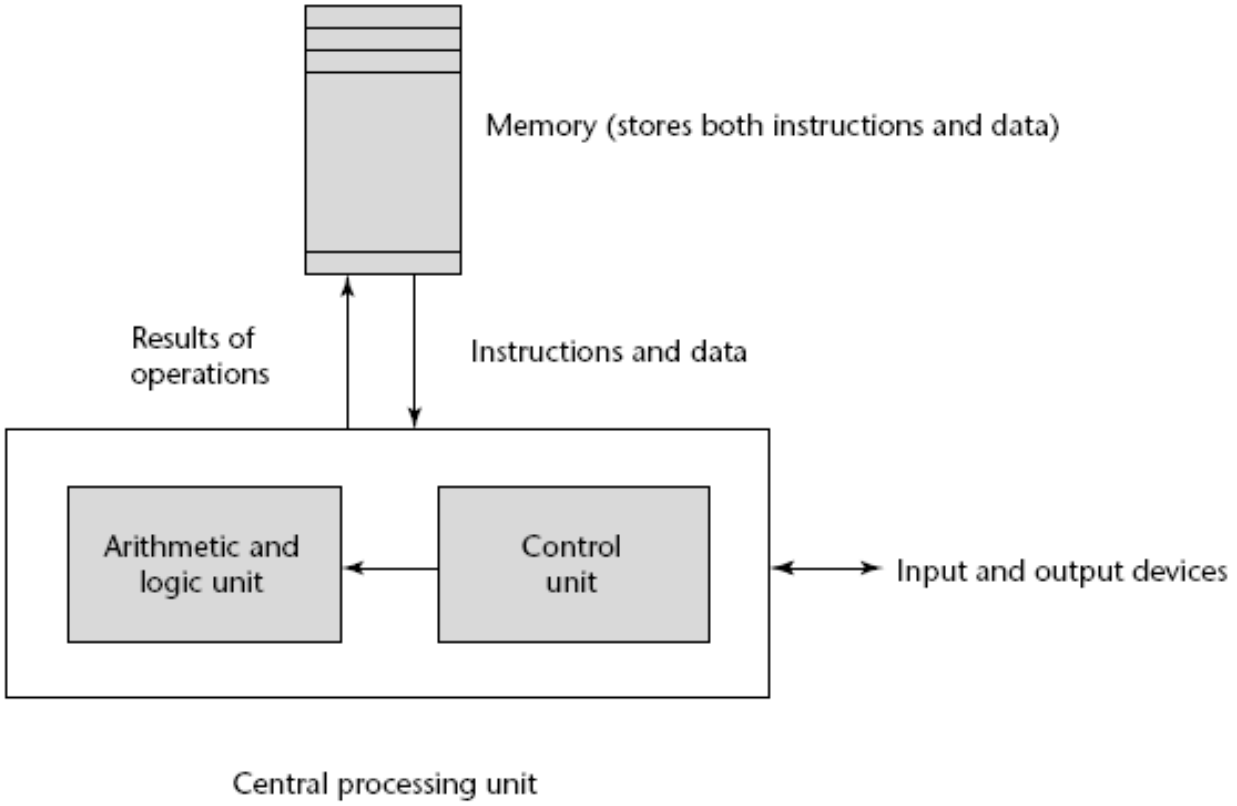
Classification of Programming Languages

- **Imperative**
  - Procedural (Fortran, Pascal, Basic, C)
  - object-oriented (Smalltalk, Eiffel, C++)
  - scripting languages (Perl, Python, JavaScript, PHP)
- **Declarative**
  - functional (Scheme, ML, pure Lisp, FP)
  - logic, constraint-based (Prolog)

Imperative languages

Figure 1.1

The von Neumann computer architecture



From: Robert W. Sebesta, Concepts of Programming Languages, Addison Wesley, 10th edition, 2012

Imperative languages

```
#include <iostream>
using namespace std;
int fact(int n) {
	int sofar = 1;
	while (n>0)
		sofar *= n--;
	return sofar;
}

int main() {
	int num;
	cout << "Please enter an integer value:";
	cin >> num;
	cout << endl << fact(num) << endl;
	return 0;
}
```

Imperative languages

- Particularly the von Neumann languages
- Predominate
- Hallmarks of imperative languages:
  - Assignment
  - Iteration
  - Order of execution is critical

Functional languages

- Hallmarks of functional languages:
  - Expression evaluation
  - Heavy use of recursion
  - High order functions
- Example: a factorial function in Lisp

```
(defun fact (x)
  (if (<= x 0) 1 (* x (fact (- x 1)))))
```

Logic languages

- Hallmarks of logic languages:
  - Program expressed as rules in formal logic

- Example: a factorial function in Prolog

```
factorial(0,1).

factorial(N,F) :-
    N>0, N1 is N-1,
    factorial(N1,F1),
    F is N * F1.
```

Why study programming languages?

- Make it easier to learn new languages
  - Some languages are similar
  - Increase the capacity to express ideas

Why study programming languages?

- Help you choose a language.
  - C vs. Java vs. C++ for systems programming
  - Fortran vs Scheme vs. AWK for symbolic data manipulation
  - Java vs. C++ vs. Visual Basic for graphical user interface

Language Influences Programming Practice

- Object-oriented languages:
  - a style making heavy use of objects
- Functional languages:
  - a style using many small side-effect-free functions
- Logic languages:
  - a style using searches in a logically-defined problem space

Example 1 - Java

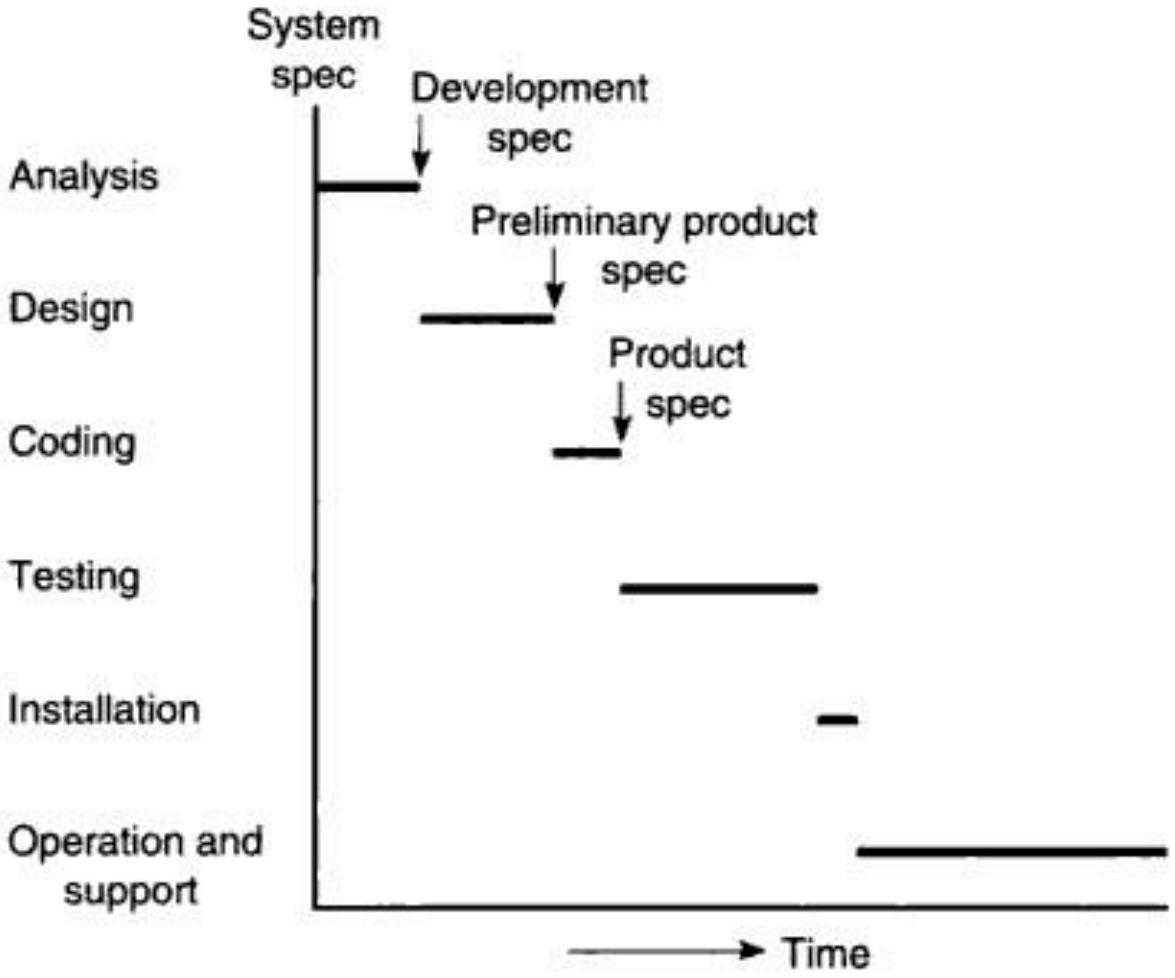
```
class Fubar {
  public static void main (String[] args)
  {
    // whole program here!
  }
}
```

Example 2 -Pascal

```
function ForLoop(Low, High: Integer): Boolean;
  begin
    if Low <= High then
      begin
        {for-loop body here}
        ForLoop := ForLoop(Low+1, High)
      end
    else
      ForLoop := True
  end;
```

Language Evaluation Criteria

Software Life Cycle



Language Evaluation Criteria

- Four Major Evaluation Criteria:
  - Readability
  - Writability
  - Reliability
  - Cost

References

- Michael L. Scott, Programming Language Pragmatics, Morgan Kaufmann, 3rd edition, 2009.
- Robert W. Sebesta, Concepts of Programming Languages, Addison Wesley, 10th edition, 2012
- Adam Brooks Webber, Modern Programming Languages, Franklin, Beedle & Associates Inc., 2nd edition, 2010.