

CS361 Programming languages and Implementation

Instructor

- Dr. Christelle Scharff
- PhD from France - 1999
Research: Automated deduction and theorem proving, New technologies in education, verification of hardware and software, data mining.
- French accent.
- Teaching in France, in Cambodia, in USA (State University of New York at Stony Brook).
- cscharff@pace.edu
- <http://www.csis.pace.edu/~scharff/>

What is cs361?

- The goal of CS 361 is to introduce the fundamental concepts in programming languages.
- It provides a study of history of programming languages including imperative, functional and logical varieties.

Emphasis will be on principles of language design, semantics and implementation strategies.

It introduces formal syntax and semantics and overviews the compilation process.

- Practically it will focus on C/C++, JAVA, PROLOG, parallel programming and particularly SML.

Description

- <http://www.csis.pace.edu/~scharff/cs361>
- Everything is on the web.
- Class time
Lecture + Exercises
- Office Hours - Where?
- Textbooks
- Exams
- Assignments
- Grades
- Academic dishonesty
- Guidelines for assignments

Schedule

- Introduction (Chap 1, Chap 15)
- Language description - syntax (Chap 2)
 - Expressions and values
 - Abstract syntax trees
 - Grammars
 - Regular expressions
- Imperative programming (Part 2)
 - Structured language
 - Types
 - Allocation, pointers
 - Binding
 - Scope, visibility
 - Parameter passing
 - Correctness of imperative programs
 - C programming

- Object oriented programming
 - Structured language
 - Abstraction, information hiding
 - Encapsulation: Abstract Data Type, objects, genericity, parametrized types.
 - Inheritance, polymorphism, overloading
 - JAVA/C++ programming

- Functional programming
 - Types
 - Functions declaration
 - Expressions evaluations
 - SML programming
 - * Types
 - * Lists, datatypes, pattern matching
 - * Polymorphism
 - * Let
 - * Currying, higher order functions

- Logic programming
 - Relations
 - Programming techniques: Resolution
 - Cut
 - PROLOG programming
- Concurrent programming
 - Processes
 - Synchronization
 - Multithreading in JAVA (instead of PVM)

Goals

- List and define the major programming paradigms and the associated thought processes for each
- Explain how languages features contribute to programming practice
- Outline the language translation process
- Understand the halting problem...

What is a language?

- A language on an alphabet Σ is a set of finite sequences of elements of the alphabets.
- The English language is the set of English words. The alphabet is $\{a, \dots, z\}$.
- Examples in mathematics:
 - The language of decimal representation of integers on the alphabet $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
 - The language of binary representation of integers on the alphabet $\{0, 1\}$.
- Abstract examples:
 - The language containing the same number of a and b on the alphabet $\{a, b\}$.
 - The language of sequence of parentheses well-parenthesized on the alphabet $\{(,)\}$.
- Examples in Computer Science:
 - The language of real numbers in PASCAL on the alphabet $\{0, 1, \dots, 9, E, +, -\}$.
 - Programming languages.

What is a programming language?

- It is a language!
- It has a syntax (?) and a semantics (?).
- A programming language:
 - makes computing convenient for people with
 - making efficient use of computing machines.
- Specific.

The art of programming

- “the art of programming is the art of organizing complexity” ,
- “we must organize the computations in such a way that our limited powers are sufficient to guarantee that the computation will establish the right effect” , Dijkstra.
- The Millenium bug.
- INTEL processor bugs galore.
- The nine-hour breakdown of AT&T’s long-distance telephone network in Jan. 1990, caused by an untested code patch, dramatized the vulnerability of complex computer systems everywhere. *“Ghost in the Machine,” Time Magazine, Jan. 29, 1990. p. 58.*
- The Ariane 5 satellite launcher malfunction was caused by a faulty software exception floating point to 16-bit integer conversion.

- Eighteen errors were detected during the 10-day flight of Apollo 14. *From G. J. Myers, Software Reliability: Principles & Practice, p. 25.*
- An error in a single FORTRAN statement resulted in the loss of the first American probe to Venus. *From G. J. Myers, Software Reliability: Principles & Practice, p. 25.*

Before at <http://www-courses.cs.uiuc.edu/~cs376/horror.html>

The tower of Babel of programming languages

- List all the programming languages you hear about, you learn or you know.
- Different families.

Why a tower of babel of programming languages?

- Major reason: the proliferation of programming tasks to be done.
- Another reason: Different philosophies developed for solving problems.
- Another reason: Improvement.

Families

- Classifications.
- Choices.
- Low level/High level
- Paradigms:
 - Paradigm:** Model or mental framework for representing or thinking about something.
 - Procedural/Imperative languages
 - Functional languages
 - Logic languages
 - Parallel/concurrent languages
 - Special purposes languages ...

Generations of programming languages

- **First generation (lowest level):** machine language.
What is the machine language?
0 and 1.
- **Second generation: assembly languages.**
 - What is assembly language?
Improvements with respect to machine language:
 - * Use of symbolic operation codes rather than numeric ones
 - * Use of symbolic memory addresses rather than numeric ones
 - * Pseudo-operations that provide user-oriented services such that data generationLanguage related to the architecture of the machine.

- Example:

```
.BEGIN
LOAD ONE, R0
// Put a 1 into register R0
STORE R0, I
// Store the constant 1 into I
INCREMENT I
// Add 1 to the memory location value at address
I.
I : .DATA 0
// The index value. Initially it is 0.
ONE: .DATA 1
// The constant 1.
.END
```

- How can a computer understand assembly languages?

Software: **Assembler**.

Assembly language (*source program*) \rightarrow *Assembler*
Machine language (*object program*)

The object code is also called **executable program**.

- **Third generation:** high-level programming languages.

Abstraction of the machine.

Program easier to write and read.

Machine independent.

How can a computer understand high-level programming languages?

Need translation from high-level language to machine language.

Software: Compiler/Interpreter.

High-level language (*source program*) $\xrightarrow{\text{Compiler}}$ Machine language (*object program*)

High-level language (*source program + expression-input*) $\xrightarrow{\text{Interpreter}}$ Evaluation of the expression

Java Byte Code

- Java byte code is not the machine language for any particular machine. It is the machine language for a hypothetical computer that is like average of all computers. This hypothetical computer is called **Java Virtual Machine**.
- The object code for the Java Virtual Machine is interpreted depending on the architecture.

Procedural languages - Imperative languages

- A program written in a procedural language consists of sequences of statements that manipulate data items; that is, they change the contents of memory cells.
- The programmer is still directing via program instructions every change in the value of a memory location. It is of course at a more abstract level than in machine or assembly languages.
- High-level programming languages - Third generation of languages.
- **Examples:** FORTRAN, COBOL, Pascal, C, C++, ADA, JAVA...

Why high-level languages?

- The assembly languages have the following disadvantages:
 - The programmer must manually manage the movement of data items between and among memory locations (cells or registers).
 - The programmer must have a *microscopic* view of a task.
 - An assembly language is *machine specific* (digital, sun, Pentium ...)
 - “Statements are not English-language-like” .

Expectations of high-level programming languages

- The programmer does not need to manage the details of the movement of data and instruction within memory.
- The programmer does not need to pay attention to exactly where the data and instructions are stored.
- The programmer can have a *macroscopic* view of tasks.
- Programs written in high-level languages are **portable**.
- Programs written in high-level languages are closer to standard English and use standard mathematical notations.

SOME PROGRAMMING LANGUAGES

FORTRAN

- FORMula TRANslation
- 1957 - First version - IBM - John Backus
- Different versions: fortran77, fortran90, high performance fortran.
- First high-level programming language
- Fortran was designed to support *numerical computations* and to optimize the object code (Machine resources are precious!).

Fortran allows external libraries of well-written, efficient and thoroughly tested code modules. (Reuse)

Fortran is still an effective language; it was the first one and a large number of applications were written and are still used.

FORTRAN

- A piece of code:

```
10 IF (NUMBER .LT. 0) GO TO 20
...
READ (*,*) NUMBER
GO TO 10
20
```

- Fortran requires variable identifiers to be upper-case.
- Early versions of Fortran do not have mathematical symbols as $<$. (*.LT.*)
- Early versions of Fortran do not have *loop mechanism*. (GO TO, labels...)
Go To Controversy, 1968, Communication of the Association of Computing Machinery, (E.W. Dijkstra)
- An error in a single FORTRAN statement resulted in the loss of the first American probe to Venus.
From G. J. Myers, Software Reliability: Principles & Practice, p. 25.

COBOL

- COmmon Business-Oriented Language.
- 1959-60 - Grace Hopper of the US Navy.
- Cobol was designed to serve *business needs* such that managing inventories and payrolls.

Much of the processing in the business world concerns updating *master files* with changes from *transactions files*.

COBOL is more adept at handling file inputs than keyboard inputs.

- COBOL uses standard mathematical notations to allow “all” people to write programs.

```
ADD A TO B GIVING SUM.
```

- Still the most widely used language (75 % of the existing code). Why?
- Y2K problem.

PASCAL

- 1970 - Niklaus Wirth

- Pascal was designed to help enforce good programming techniques and to be easy to learn.

It was the primary language taught to computer science majors.

- Pascal was not plainly accepted as an industrial language.

But it is used now in Delphi, Borland International Inc. Delphi is a programming environment to develop windows-based applications with modern graphical user interfaces.

C

- C - 1970 - Dennis Ritchie - AT&T Bell Laboratories
- It was originally designed for systems programming - to write the operating system UNIX.
- C is popular because of UNIX and because of its efficiency (speed with which its operations can be executed). C is also portable.
- C is closer to assembly language than other high-level languages (Use of pointers).

```
int *intPointer;  
intPointer = (int *)800;  
*intPointer = 50;
```

What does it mean?

- C is convenient to write systems software as operating systems, assemblers, compilers, programs that allow the computer to interact with input/output devices...
- C is one of the most widely used language. It combines the power of high-level languages with the ability to circumvent that level of abstraction and work at the assembly-language-like level.

C++

- 1980 - Bjarne Stroustrup - AT&T Bell Laboratories
- C++ is a superset of C.
- OOP (Object Oriented Language).
- Standardization (ANSI, ISO), object-orientation and a strong collection of libraries have helped to make C++ one of the most popular of the modern industrial strength-languages.

ADA

- Ada Augusta Byron Lovelace.
- 1979 - US armed services
- It was designed to develop a common high-level programming language for use by defense contractors.
- First requirements of ADA: efficiency, reliability, readability and maintainability.
- OOP.
- Multi-processing: multiple tasks are executing independently and they can synchronize and communicate.

Java

- Sun Microsystems – Different projects (1991).
- An accident – Success of the WEB.
- 1996 – the first version of Java.
- OOP.

Procedural languages

- Weaknesses and strengths.
- Improvements.
- How to determine the qualities of a language?

OOP

- In the procedural languages we can distinguish non OOP and OOP languages.
- The **divide and conquer method** is a traditional approach to split problems into subtasks.
- Object-oriented method is a **new and original approach**.
- A program is considered as a simulation of some part of the world that is the domain of interests.
Objects populate this domain.
- A class has subtasks associated with it. All objects of a class can perform those subtasks.
Objects are instances of classes. Each object that exists has its own properties during the execution of the program.
The new idea is that instead of asking a class to carry out subtasks (traditional method – divide and conquer), we ask to objects to carry out subtasks.
- There are semantics relations between classes.
- Answer the questions: How to manage ...? What is ...?

- **Examples:**

- **World = bank**

- Objects are:

- employees...

- savings accounts, checking accounts, loans...

- **World = medical office**

- Objects are:

- patients, doctors...

- **World = store**

- Objects are:

- employees, managers, full-time and part-time employees...

- items, clothes, food, luxury items...

- **World = geometry**

- Objects are:

- ellipse, circle...

- polygon, rectangle, square, triangle...

- **World = the department of motors and vehicles**

- Objects are:

- vehicles, cars, trucks...

Alternative programming paradigms

- **Paradigm:** Model or mental framework for representing or thinking about something.

Example: The paradigm of procedural languages says that a sequence of detailed instructions is provided to the computer and each instruction is concerned with accessing and modifying the contents of a memory location.

- There are languages that are completely different from procedural languages in form, structure and syntax. They use a different paradigm.
 - *Functional programming:* Based on functions (LISP, ML, SML, CAML, OCCAML...).
 - *Logic programming:* Based on logical deductions from existing facts (PROLOG...).
Logic deals with the formalization of natural language and reasoning methods.
 - *Parallel programming:* Multiple copies of the same subtask or several subtasks of the same problem being performed simultaneously by different processors (Multithreading in JAVA, OCCAM, PVM...).

Functional programming

- Every task is viewed in terms of *functions*.
- Complex combination of functions that use the results of applying other functions, that use the results of applying other functions...

Applicative languages

SML

- Standard Meta Language
- Close from Mathematics (sets, tuples, functions...).
- No declarations as in Imperative languages.

- No side effects.

Side effects: A code module changes values that has no business changing.

- *Inference of type.*

You do not always need to specify the type of your variables. The type of your functions and variables is computed.

- Types in SML: int, real, bool, lists (int list, real list, int int list), tuples (int*int, real*real*real...)

Examples:

- [1, 4, 5] is a list of integers (type: int list).
- (1, 1) is a tuple (type: int*int).

- **Examples:**

- The doubling function $f(x) = 2 * x$ is written:

- `fun double(x) = 2*x;`

- The square function $f(x) = x * x$ of 2 integers is written:

- `fun f(x:int) = x*x;`

Logic Programming

- In logic programming, various *facts* are asserted to be true and on the basis of these facts, a logic program can infer or deduce other *facts*.

When a *query* is posed to the program, it answers the query by beginning with the facts and then by attempting to apply logical deductions.

Logical deductions are based on doing **proofs by contradiction** (assuming the inverse of what one wants to prove and deriving a contradiction), more precisely on **Resolution**.

- A logic program is also called a *declarative program*.
- Logic programming is used to write *expert systems* (Artificial intelligence).

PROLOG

- PROgramming in LOgic
- A Prolog program is:
 - a set of facts, and
 - a set of rules.
- Consider the following program:

```
P(Edward VII, George V).  
P(Victoria, Edward VII).  
P(Alexandra, George V).  
P(George VI, Elizabeth II).  
P(George V, George VI).  
G(x,y):-P(x,z),P(z,y).
```

- The facts specify the relation *Parent*.
- The last rule defines the *Grandparent relation* in terms of the *Parent relation*: a person x is a grandparent of y if there is a third person z , such that x is the parent of z , and z the parent of y .

– Queries:

– Alexandra is the mother of George V?

?- $P(\text{Alexandra}, \text{George V})$.

$P(\text{Alexandra}, \text{George V})$ is a fact so it is true.

– Victoria is the grand-mother of George V?

?- $G(\text{Victoria}, \text{George V})$.

We have $P(\text{Victoria}, \text{Edward VII})$ and
 $P(\text{Edward VII}, \text{George V})$ that are facts.

Furthermore by the rule:

$G(x, y) \leftarrow P(x, z), P(z, y)$,

we can deduce that $G(\text{Victoria}, \text{George V})$.

So it is true.

– Who are the parents of George V?

?- $G(X, \text{George V})$.

From $P(\text{Edward VII}, \text{George V})$ and
 $P(\text{Alexandra}, \text{George V})$, we can deduce that the
parents of George V are Edward VII and Alexandra
V.

Parallel Programming

- Sequential (Von Neumann Architecture) versus parallel.
- *Parallel Programming* is a catch-all term for a variety of computing architectures and approaches to algorithms.
- Different architectures:
 - * *SIMD*: (Single Instruction Multiple Data) A single control unit broadcasts a single program instruction to multiple ALUs, each of which carries out that instruction on its own local data stored in its local memory.
 - * *MIMD*: (Multiple Instruction Multiple Data) Numerous multi-processors execute their own programs on their own data communicating results as necessary.

Special purpose languages

- Languages designed for specialized tasks.
- **Examples:** SQL, PERL, HTML...

SQL

- Structured Query Language
- 1986 - IBM.
- It is designed to be used with *databases*, which are collections of related facts and information.
- A **database** stores data. The user of that database must be able to add and to retrieve data.
- SQL uses the relational model.

In the relational model, a database is a set of *relations*.

Example

Management of a store

- Database *Store* composed of the relations: *Employees* and *Items*.
- *Design of the database*

- *Employees*

ssnb	Fname	Gname
001-77-4888	Smith	John
113-75-4889	Smith	Georges
111-52-7531	Jordan	James
888-78-2585	Di Caprio	John

- *Items*

nbitem	name	brand	price
88	shirt	levis	7.5
89	caviar	MyRussia	100
111	television	Panasonic	259

- *Implementation of the database in a Database Management System*

Examples: Sybase, Oracle are database management systems.

- *Queries*

- Family names of the people in the relation *Employees*.

```
SELECT Fname  
FROM Employees
```

Answer?

- Given name of the people with family name *Smith*.

```
SELECT Gname  
FROM Employees  
where Employees.Fname = 'Smith'
```

Answer?

- Price of the item number 89.

```
SELECT price  
FROM Items  
where Items.nbitem = 89
```

Answer?

Another example

- A Shakespearian *killed* relation would be:

Killer	Victim
Brutus	Caesar
Hamlet	Laertes
Hamlet	Polonius
Laertes	Hamlet
Brutus	Brutus
Cassius	Caesar

- *Who killed Caesar?*

PERL

- Practical Extraction and Report language
- Perl is designed *to scan arbitrary text files, to extract information in these files and to print reports based on the extracted information.*
- Perl uses sophisticated pattern-matching techniques.
Pattern-matching: To find a pattern.

Examples:

- Find all the family names of the students that begin with 'S'.
 - Find all the Word files on your hard disk.
- **Examples where Perl is used:** Fill out a form on the web.

HTML

- HyperText Markup Language
- This language is used to create HTML documents that become Web Pages when viewed with Web Browse Software.
Examples: Internet Explorer and Netscape are Web Browsers.
- An *HTML program* consists of:
 - the text to display on the Web page,
 - a number of special characters called *tags* that achieve formatting, and
 - references to other HTML documents (Web pages, images...).

- Skeleton of an HTML program:

```
<html>
<head>
<title> .... </title>
</head>

<body>
....
</body> </html>
```

- The text between

```
<head>
```

and

```
</head>
```

and

```
<title>
```

and

```
</title>
```

frames what will appear in the title bar of the Web browser and in the favorites if the Web page is bookmarked.

- The text between

```
<body>
```

and

```
</body>
```

frames what will be on the Web page itself.

- Link to another document:

```
<A HREF="file.html"> ... </A>
```

```
<A HREF="http://www.address.com/file.html"> ... </A>
```

- Insertion of an image:

```
IMG SRC = "img.jpg"
```

- An HTML program is saved in a file with extension *html*.

Examples: file.html, foo.html...

Example

- What is the result of:

```
<p>These are the <i>times</i> that try <b>men's  
souls<b><p>
```

- These are the *times* that try **men's
souls**

Conclusion

- A number of languages.
- Traditional paradigm: procedural paradigm.
- Some languages are designed for special-purpose tools.
- Original languages.
- It is certain that the programming language world has been and continues to be a tower of babel.