

Programming Languages

Outline

- What makes programming languages an interesting subject?
 - The amazing variety
 - The odd controversies
 - The intriguing evolution
 - The connection to programming practice
 - The many other connections

The Amazing Variety

- There are very many, very different languages
- (A list that used to be posted occasionally on `comp.lang.misc` had over 2300 published languages in 1995)
- Often grouped into four families:
 - Imperative
 - Functional
 - Logic
 - Object-oriented

Imperative Languages

- Example: a factorial function in C

```
int fact(int n) {  
    intsofar = 1;  
    while (n>0) sofar *= n--;  
    return sofar;  
}
```

- Hallmarks of imperative languages:
 - Assignment
 - Iteration
 - Order of execution is critical

Functional Languages

- Example: a factorial function in ML

```
fun fact x =  
    if x <= 0 then 1 else x * fact(x-1) ;
```

- Hallmarks of functional languages:
 - Single-valued variables
 - Heavy use of recursion

Another Functional Language

- Example: a factorial function in Lisp

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

- Looks very different from ML
- But ML and Lisp are closely related
 - Single-valued variables: no assignment
 - Heavy use of recursion: no iteration

Logic Languages

- Example: a factorial function in Prolog

```
fact(X,1) :-  
    X == 1.  
fact(X,Fact) :-  
    X > 1,  
    NewX is X - 1,  
    fact(NewX,NF) ,  
    Fact is X * NF.
```

- Hallmark of logic languages
 - Program expressed as rules in formal logic

Object-Oriented Languages

- Example: a Java definition for a kind of object that can store an integer and compute its factorial


```
public class MyInt {  
    private int value;  
    public MyInt(int value) {  
        this.value = value;  
    }  
    public int getValue() {  
        return value;  
    }  
    public MyInt getFact() {  
        return new MyInt(fact(value));  
    }  
    private int fact(int n) {  
        int sofar = 1;  
        while (n > 1) sofar *= n--;  
        return sofar;  
    }  
}
```

Object-Oriented Languages

- Hallmarks of object-oriented languages:
 - Usually imperative, plus...
 - Constructs to help programmers use “objects”—little bundles of data that know how to do things to themselves

Strengths and Weaknesses

- The different language groups show to advantage on different kinds of problems
- Decide for yourself at the end of the semester, after experimenting with them
- For now, one comment: don't jump to conclusions based on factorial!
 - Functional languages do well on such functions
 - Imperative languages, a bit less well
 - Logic languages, considerably less well
 - Object-oriented languages need larger examples

About Those Families

- There are many other language family terms (not exhaustive and sometimes overlapping)
 - Applicative, concurrent, constraint, declarative, definitional, procedural, scripting, single-assignment, ...
- Some *multi-paradigm* languages straddle families: JavaScript, OCaml, Python, Ruby
- Others are so unique that assigning them to a family is pointless

Example: Forth Factorial

: FACTORIAL

1 SWAP BEGIN ?DUP WHILE TUCK * SWAP 1- REPEAT ;

- A stack-oriented language
- Postscript is similar
- Could be called *imperative*, but has little in common with most imperative languages

Example: APL Factorial

$\times / \mathbf{1} \mathbf{X}$

- An APL expression that computes X 's factorial
- Expands X it into a vector of the integers $1..X$, then multiplies them all together
- (You would not really do it that way in APL, since there is a predefined factorial operator: $!X$)
- Could be called *functional*, but has little in common with most functional languages

Outline

- What makes programming languages an interesting subject?
 - The amazing variety
 - **The odd controversies**
 - The intriguing evolution
 - The connection to programming practice
 - The many other connections

The Odd Controversies

- Programming languages are the subject of many heated debates:
 - Partisan arguments
 - Language standards
 - Fundamental definitions

Language Partisans

strong supporter

- There is a lot of argument about the relative merits of different languages
- Every language has partisans, who praise it in extreme terms and defend it against all detractors

Language Standards

- The documents that define language standards are often drafted by international committees
- Can be a slow, complicated and rancorous process
- Fortran ~~82~~ ~~8X~~ ~~88~~ 90 standard released in 1991

Basic Definitions

- Some terms refer to fuzzy concepts: all those language family names, for example
- No problem; just remember they are fuzzy
 - Bad: Is X really an *object-oriented* language?
 - Good: What aspects of X support an *object-oriented* style of programming?
- Some crisp concepts have conflicting terminology: one person's *argument* is another person's *actual parameter*

Outline

- What makes programming languages an interesting subject?
 - The amazing variety
 - The odd controversies
 - **The intriguing evolution**
 - The connection to programming practice
 - The many other connections

The Intriguing Evolution

- Programming languages are evolving rapidly
 - New languages are being invented
 - Old ones are developing new dialects

New Languages

- A clean slate: no need to maintain compatibility with an existing body of code
- But never entirely *new* any more: always using ideas from earlier designs
- Some become widely used, others do not
- Whether widely used or not, they can serve as a source of ideas for the next generation

Widely Used: Java

- Quick rise to popularity since 1995 release
- Java uses many ideas from C++, plus some from Mesa, Modula, and other languages
- C++ uses most of C and extends it with ideas from Simula 67, Ada, Clu, ML and Algol 68
- C was derived from B, which was derived from BCPL, which was derived from CPL, which was derived from Algol 60

Not Widely Used: Algol

- ❑ One of the earliest languages: Algol 58, Algol 60, Algol 68
- ❑ Never widely used
- ❑ Introduced many ideas that were used in later languages, including
 - Block structure and scope
 - Recursive functions
 - Parameter passing by value

Dialects

- Experience with languages reveals their design weaknesses and leads to new dialects
- New ideas pass into new dialects of old languages

Some Dialects Of Fortran

- Original Fortran, IBM
- Major standards:
 - Fortran II
 - Fortran III
 - Fortran IV
 - Fortran 66
 - Fortran 77
 - Fortran 90
 - Fortran 95
 - Fortran 2003
 - Fortran 2008?
- Deviations in each implementation
- Parallel processing
 - HPF
 - Fortran M
 - Vienna Fortran
- And many more...

Outline

- What makes programming languages an interesting subject?
 - The amazing variety
 - The odd controversies
 - The intriguing evolution
 - The connection to programming practice
 - The many other connections

The Connection To Programming Practice

- Languages influence programming practice
 - A language favors a particular programming style—a particular approach to algorithmic problem-solving
- Programming experience influences language design

Language Influences

Programming Practice

- Languages often strongly favor a particular style of programming
 - 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

Fighting the Language

- ❑ Languages favor a particular style, but do not force the programmer to follow it
- ❑ It is always possible to write in a style not favored by the language
- ❑ It is not usually a good idea...

Imperative ML

ML makes it hard to use assignment and side-effects. But it is still possible:

```
fun fact n =  
  let  
    val i = ref 1;  
    val xn = ref n  
  in  
    while !xn > 1 do (  
      i := !i * !xn;  
      xn := !xn - 1  
    );  
    !i  
  end;
```

Non-object-oriented Java

Java, more than C++, tries to encourage you to adopt an object-oriented mode. But you can still put your whole program into static methods of a single class:

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


Functional Pascal

Any imperative language that supports recursion can be used as a functional language:

```
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;
```

Programming Experience Influences Language Design

- Corrections to design problems make future dialects, as already noted
- Programming styles can emerge before there is a language that supports them
 - Programming with objects predates object-oriented languages
 - Automated theorem proving predates logic languages

Outline

- What makes programming languages an interesting subject?
 - The amazing variety
 - The odd controversies
 - The intriguing evolution
 - The connection to programming practice
 - **The many other connections**

Other Connections:

Computer Architecture

- Language evolution drives and is driven by hardware evolution:
 - Call-stack support – languages with recursion
 - Parallel architectures – parallel languages
 - Internet – Java

Other Connections:

Theory of Formal Languages

- Theory of formal languages is a core mathematical area of computer science
 - Regular grammars, finite-state automata – lexical structure of programming languages, scanner in a compiler
 - Context-free grammars, pushdown automata – phrase-level structure of programming languages, parser in a compiler
 - Turing machines – Turing-equivalence of programming languages

Turing Equivalence

- Languages have different strengths, but fundamentally they all have the same power
 - {problems solvable in Java}
 - = {problems solvable in Fortran}
 - = ...
- And all have the same power as various mathematical models of computation
 - = {problems solvable by Turing machine}
 - = {problems solvable by lambda calculus}
 - = ...
- *Church-Turing thesis*: this is what “computability” means

Conclusion

- Why programming languages are worth studying (and this course worth taking):
 - The amazing variety
 - The odd controversies
 - The intriguing evolution
 - The connection to programming practice
 - The many other connections
- Plus...there is the fun of learning three new languages!