

CSc 372 — Comparative Programming Languages

1 : Introduction

Christian Collberg
Department of Computer Science
University of Arizona
collberg+372@gmail.com

Copyright © 2005 Christian Collberg

August 22, 2005

1 Why learn programming languages?

- In this class we will study three languages: Prolog, Haskell, and Icon.
- There are several reasons why you would want to learn a large number of languages:
 1. There will always be new languages used in industry. Recently, we've gone from C to Ada to C++ to Java and (maybe) to C#. Every computer scientist should be ready to make this change.
 2. Learning a new *programming paradigm* teaches you new ways to solve problems.

2 Functional Programming (FP)

- Functional programming is a way to program in a more “mathematical” way.
- An FP program consists of a collection of simple functions which are combined into more complex functions, which are combined. . . , etc.
- FP programs are easier to reason about mathematically than imperative (C) or object-oriented programs.
- We are going to study *Haskell*, one of the more popular modern FP languages.

3 Logic Programming (FP)

- Logic programming is a way to program using ideas from logic, such as first order predicate calculus.
- There really is only one well-know language in this class, *Prolog*, and that is what we will study.
- Prolog allows you to solve some very complex problems very easily.

4 String Processing

- *Icon* is a string processing language developed here at the UofA.
- *Icon* is really a general purpose imperative language, but it has some very powerful ways of manipulating strings.
- Other, more modern, languages in this class are *Perl*, *Python*, *Tcl*, and *Ruby*.
- These languages are used more and more in real applications, since writing a *Perl* program is often much faster than writing the equivalent *Java/C/C++* program.

A Preview

5 3 Languages — A Preview

You Are Not Supposed to Understand This
Lecture!!!

yet...



6 Hello World (Prolog)

The file `hello.pl`

```
hello :-  
    write('Hello World!'),nl.
```

Loading and running

```
> gprolog  
| ? - ['hello.pl'].  
| ? - hello.  
Hello World!  
  
yes  
| ? -
```

7 Hello World (Haskell)

The file `hello.gh`

```
main = putStr ("Hello World")
```

Loading and running

```
> hugs  
  
Main> :load hello.gh  
Main> main  
Hello World  
Main>
```

8 Hello World (Icon)

[The file hello.icn](#)

```
procedure main()
  write(" Hello World!")
end
```

[Compiling and running](#)

```
> icont hello.icn
> hello
Hello World!
```

9 Hello World (Java)

```
class Hello {
  String message;

  Hello(String message) {
    this.message = message;
  }

  void sayit() {
    System.out.println(message);
  }

  public static void main(String[] args) {
    Hello myHello = new Hello(" Hello World");
    myHello.sayit();
  }
}
```

10 Repeating Hello World (Prolog)

[The file hello.pl](#)

```
hello2(0).
hello2(N) :-
  N>0,
  write(' Hello World! '), nl,
  N1 is N - 1,
  hello2(N1).
```

[Loading and running](#)

```
> gprolog
| ? - ['hello.pl'].
| ? - hello2(2).
Hello World!
Hello World!
```

11 Repeating Hello World (Haskell)

[The file `hello.gh`](#)

```
main n = putStr (unlines (take n (repeat "Hello World!")))
```

[Loading and running](#)

```
> hugs
Main> :load hello.gh
Main> main 2
Hello World!
Hello World!
```

- `repeat "Hello World!"` generates an infinite list of strings.
- `take n [...]` takes the first `n` elements of a list, and throws away the rest.
- `unlines [...]` concatenates a list of strings into one string.

12 Repeating Hello World (Icon)

[The file `hello.icn`](#)

```
procedure hello(n)
  every i:= 1 to n do
    write(" Hello World!")
end

procedure main()
  hello(2)
end
```

[Compiling and running](#)

```
> icont hello.icn
> hello
Hello World!
Hello World!
```

13 3 Languages — A Preview

Remember...

You Are Not Supposed to Understand This
Lecture!!!

yet...

...but you will need to know it all for the final!



14 Readings and References

- Hello World! in over two hundred languages: <http://www2.latech.edu/~acm/HelloWorld.shtml>.

15 Homework

- Go to the 372 web page and browse around the information about the different languages.
- If you own your own computer, download and install the different compilers/interpreters.
- Try to run the examples in this lecture, on your own machine, on **lectura**, or on the Windows machines in the lab.

16 Summary

- In this class we will study three languages: Prolog, Haskell, and Icon.
- Haskell is a *functional programming languages*.
- Prolog is a *logic programming language*.
- Icon is a *string processing language*.