Welcome - CSC 301

CSC 301- Foundations of Programming Languages

- Instructor: Dr. Lutz Hamel
- Email: hamel@cs.uri.edu
- Office: Tyler, Rm 251
- Office Hours: TBA
- TA: TBA

MODERN PROGRAMMING LANGUAGES: A PRACTICAL INTRODUCTION
Adam Brooks Webber
Why Study Programming Languages?

- Amazing variety
  - One of the moderated email lists counted ~2300 different programming languages (comp.lang.)*
- “Strange” controversies
  - Should a programming language have a ‘goto’ statement?
  - Should an OO language allow for global functions?
  - Terminology: argument vs. actual parameter.
- Many connections
  - Programming languages touch upon virtually all areas of computer science: from the mathematical theory of formal languages and automata to the implementation of operating systems.
- Intriguing evolution
  - Programming languages change!
    - New ideas and experiences trigger new languages.
    - New languages trigger new ideas, etc.
There are many different programming language classes, but four classes or paradigms stand out:

- Imperative Languages
- Functional Languages
- Logic/Rule Based Languages
- Object-Oriented Languages
Recursive definition of the factorial operator

\[ x! = \begin{cases} 
1 & \text{if } x = 1, \\
x(x - 1)! & \text{otherwise.} 
\end{cases} \]

for all \( x > 0 \).
Imperative Languages

- Hallmarks: assignment and iteration
- Examples: C, FORTRAN, COBOL
- Example Program: factorial program in C

```c
int fact(int n) {
    int sofar;
    sofar = 1;
    while (n > 1) {
        sofar = sofar*n;
        n--; // assignment
    }
    return sofar;
}
```
Observations:

- The program text determines the order of execution of the statements.
- We have the notion of a ‘current value’ of a variable – accessible state of variable.

This is not always true in other languages.
Functional Languages

- Hallmarks: recursion and *single valued variables*.
- Examples: ML, Lisp, Haskell
- Example Program: factorial program in ML

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

recursion
Functional Languages

Observations:

- There are no explicit assignments.
- The name stems from the fact that programs consist of recursive definitions of functions.
Logic Programming Languages

- Hallmarks: programs consist of **rules** that specify the problem solution.
- Examples: Prolog, Maude
- Example Program: factorial program written in Prolog

\[
\begin{align*}
\text{fact}(1,1). \\
\text{fact}(X,F) & : - \\
& \quad \text{X} > 1, \\
& \quad \text{X1} \text{ is X} - 1, \\
& \quad \text{fact}(-X1, F1), \\
& \quad F \text{ is X} \times F1.
\end{align*}
\]
Observations:

- Rules do not appear in the order of execution in the program text.
- No specific order of execution is given – rules ‘fire’ when necessary.
Object-Oriented Languages

- Hallmarks: bundle data with the allowed operations
  
- Examples: Java, C++, Smalltalk
  
- Example Program: factorial program in Java

```java
class FactInt {
    private int val;
    public FactInt(int x) {
        val = fact(x);
    }
    public int getVal() {
        return val;
    }
    private int fact(int n) {
        int sofar = 1;
        while (n > 1) {
            sofar = sofar * n;
            n--;
        }
        return sofar;
    }
}
```
General Observations:

- Programming languages guide programmers towards a particular programming style:
  - Functional $\rightarrow$ mathematical functions
  - OO $\rightarrow$ objects
  - Logic $\rightarrow$ rules

- Programming itself guides the developer towards new language ideas:
  - Recursion was introduced by John McCarthy in the 1950’s with the programming language Lisp to solve problems in AI.
  - Classes and objects were developed by Nygaard and Dahl in the 1960’s and 70’s for the language Simula in order to solve problem in simulations.
There exist many programming languages today (> 2000)

In order to understand the similarities and differences ⇒ sort into classes
- Imperative
  - assignment and iteration
- Functional
  - Recursion, single valued variables
- Logic/rule based
  - programs consist of rules
- Object-oriented
  - bundle data with the allowed operations
Reading

- Chapters 1&2