Today

- Survey (Quiz 0)
- Course Overview
- MyPL intro

Assignments

- HW1 (out, due next Tues)
Course Overview

Communication

- Website: http://www.cs.gonzaga.edu/bowers/courses/cpsc326
- Submit code via Github Classroom
- Piazza (you’ll receive an invite)

Please be sure to **carefully** read the syllabus

- **Office hours**: TR 11-1, W 1-2, or by appointment
- **Grading**: 40% hw, 10% quizzes, 30% 3 mid-semester exams, 20% final
- *Concepts of Programming Languages (11th Edition)*, Sebesta

Exam, Quiz, and Homework basics

- Exams and Quizzes are closed book, closed note, your own work
- Okay to work on, discuss assignments together (but plagiarism not allowed)
- You are responsible for understanding how to complete the assignments
- You will be tested on material from assignments (written & programming)
- Must turn in homework individually
Warning: This course has significant amounts of programming / hw

- First half of course assignments build on each other
- Second half of course, new languages
- Second half of course also has a project

Class Sessions

- Mix of lecture, exercises, and short quizzes
- **No electronic devices** allowed during lecture, discussion, quizzes
  - Please see me for accommodations
  - Okay to use during exercises
- Quizzes typically once per week at beginning of class

Homework

- All assignments must have a cover sheet
- Due dates are firm ... don’t fall behind
Course Topics

This course covers various aspects of programming languages

Focus is on:

- programming language design and implementation
- programming language paradigms (or “families”)
- use of different languages

General topics

About 1/2 of course (approximately 7 weeks)

- Language design and implementation (syntax, semantics, interpretation)
- ... while not a compiler class, we’ll implement a (fake) language from scratch
- ... focus is on compiler “front end"
- ... learning about design and implementation demystifies languages
- ... can make it easier to learn new languages
- ... implementation in Java

About 1/2 of course (approximately 8 weeks)

- Functional programming in Haskell
- Logic programming using ASP (Clingo)
- ... learning these languages will make you a better programmer!
Quizzes and Project

We'll have frequent quizzes

- No make up quizzes
- Each quiz is worth same amount (10 points per quiz)

Why?

- Give you feel for kind of questions I ask on exams
- Give you feedback on your understanding of material
- Give me feedback on your understanding
- Short quizzes shown to help improve final course grades / retention

End of semester project on an aspect of programming languages

- extend the interpreter (constructs)
- add new features to the interpreter (e.g., garbage collection)
- rewrite parts using parsing tools
- dive deeper into “back end” of compilation
Brief intro to “MyPL” v3.0

Basics

- Made up language for exploring language implementation ideas
- Mixes explicit and implicit typing (for type checking)
- Has many basic constructs you would find in a PL

Data Types:

int, double, bool, char, string, nil

Values:

0, 1, 7, 10, 20, 1000000000000 (ints)
1.0, 1.01, 10.3, 0.5, etc (doubles)
true, false (bools)
’a’, ‘b’, ‘1’, etc (chars)
"foo", "bar", "", etc (strings)
nil (nil value)

Comments:

# this is a single line comment
# we only support single line comments

Variable declarations (Must have initializers)

```plaintext
var int x := 5         # optional explicit type
var int y := 5*3 + 2   # optional explicit type
var z := true          # implicit type (preferred)
var u := "foo"         # implicit type (preferred)
var string x := nil    # required explicit type
```
Variable assignments

set x := 10          # x must be declared
set z := false      # z must be declared

For loops (only work over int values)

for x := 1 to n do  # x declared locally
   ...
end

Relational comparators and Boolean operations

x = 1, x < 1, x <= 1, x > 1, x >= 1, x != 1
true and false or true

While loops

while x > 1 and x < 2 do
   ...
end

Conditionals

if x = 1 or x = 2 then
   ...
elif y > 20 and y <= 30 then
   ...
else
   ...
end
Math ops

- x + y # int, double addition
- x - y # int, double subtraction
- x * y # int, double multiplication
- x / y # int, double division
- x % y # int mod
- neg x # int, double negation

Functions

- # function that returns an int
  fun int f(int x, int y)
    var z := x + y
    if x < y then
      set z := neg z
    end
    return z
  end

- # no return value
  fun nil g(int x)
    print(itos(x)) # convert int to string
    print("\n")
  end

  var x := f(1, 2) # call f
  g(x) # call g
Structured types

type Node
    var val := 0
    var Node next := nil
end

var n1 := new Node # instance of Node
set n1.val := 10 # attribute access
var n2 := new Node
set n2.val := 20
set n1.next := n2
set n2.next := nil

Built in functions

var string s := nil
var int x := nil
var char c := nil

set s := itos(4) # converts int to string
set x := stoi("4") # converts string to int
set s := dtos(3.1) # converts double to string
set x := stod("3.1") # converts string to double

print("foo") # prints to standard output
set s := read() # read from standard input

set x := length("foo") # number of string chars
set c := get(0, "foo") # get first string char

set s := concat("ab", "cd") # string concatenation
set s := append("ab", ‘c’) # add char to end of string
MyPL Programs

- we only consider single-file programs
- normal scoping rules (more later)
- structs and functions have to be defined outside of other functions
- no “main” function (like in python and other scripting languages)