CSE 130 [Spring 2014] Programming Languages

Course Introduction

Filling in today for Ranjit Jhala

Ravi Chugh

Apr 01
Two variables
- $x, y$

Three operations
- $x++$
- $x--$
- $(x=0) ? L1 : L2$

L1: $x++;$
L2: $y--;$
(y=0) ? L2 : L1

Fact: This is “equivalent to” to every PL!

Good luck writing quicksort ...
... or Windows, Google, Spotify!
“A different language is a different vision of life”
- Federico Fellini
So Why Study PL?

Programming Language
Shapes
Programming Thought
So Why Study PL?

Programming Language

Affects How:

• Ideas are expressed
• Computation is expressed
Course Goals

“Free your mind”
- Morpheus
Learn New Languages / Constructs

New ways to:
- describe
- organize
- think about computation
Write Programs That Are:

- Readable
- Correct
- Extendable
- Modifiable
- Reusable
I WANT YOU

To Learn ...
Learn How to Learn New PLs

No Java, C# 20 years ago

Learn the anatomy of a PL
- Fundamental building blocks
- Different guises in different PLs

Re-learn the PLs you already know
I WANT YOU

To Design New Languages
Learn How to Design New PLs

... “Who, me?”

Buried in every extensible system is a PL

• Emacs: Lisp
• Word, Powerpoint: Macros, VBScript
• Unreal: UnrealScript (Game Scripting)
• Facebook: FBML, FBJS
• SQL, Renderman, LaTeX, XML ...
Choose Right Language
Learn How to Choose Right PL

“... But isn’t that decided by

• libraries,
• standards,
• and my boss?”

Yes!

My goal: educate tomorrow’s tech leaders & bosses, so you’ll make informed choices
Speaking of Right and Wrong...
Imperative Programming
x = x + 1
x = x + 1
Imperative = Mutation
Don’t Take My Word For It

John Carmack
Creator of FPS: Doom, Quake,...

I am starting to remove op= operator overloads to discourage variable mutation.
Don’t Take My Word For It

Tim Sweeney (Epic, Creator of UNREAL)

“In a concurrent world, imperative is the wrong default”
Functional Programming
Functional Programming

No Assignment.
No Mutation.
No Loops.
OMG!
Who uses FP?!?
So, Who Uses FP?

MapReduce
So, Who Uses FP?

Linq, F#
So, Who Uses FP?

Facebook

Erlang
So, Who Uses FP?

Scala
So, Who Uses FP?

OCaml
So, Who Uses FP?

CSE 130!
Course Staff

Instructor

Ranjit Jhala

TAs

Eric Seidel

Tutors

Patrick Torbett
Derek Huynh
Lucas Cycon

Office Hours TBD, Check Website ...

Daniel Ricketts
cseweb.ucsd.edu/classes/sp14/cse130-a/

• Nothing printed, everything on the web

• Feel free to bring a laptop to class
• **Piazza** will be used for all questions, announcements, clarifications, etc.

• Check often!

• If you haven’t received a request to join, email **jhala@cs.ucsd.edu**
Discussion Sections

Wednesdays
5:00pm to 5:50pm
6:00pm to 6:50pm
SOLIS 104 (same as lecture)

Starts Tomorrow!
Quick Poll:

How many people do not already have an i>clicker?
Clickers + Peer Instruction (ish)

• Make class interactive
  • Help YOU and ME understand what’s tricky

• Everyone must bring an i>clicker to class
  • by Tues 4/8 (we’ll “practice” until then)
  • available at bookstore

• Seating in assigned groups
  • Check course webpage
In-Class Clicker Exercises

1. Solo Vote: Think for yourself, then vote

2. Group Discussion: Groups of ~3 students
   - Practice analyzing, talking about tricky notions
   - (Try to) reach consensus
   - If you have questions, raise your hand!

3. Group Vote: Everyone in group votes

4. Class Discussion:
   - What did you find easy/hard?
   - Questions from here show up in exams
In-Class Clicker Exercises

- Participation counts for 5% of your grade

- Respond to 75% of the questions throughout the quarter

- So, don’t fret if you miss a class or two

- Register your clicker! (check webpage)
Grading and Exams

- No Official “Before-Class” Homework
- In-Class Exercises: 5%
- Programming Assignments (6-8): 30%
- Midterm: 30%
- Final: 35%
Programming Assignments

Schedule up on webpage

Usually due on Mondays or Fridays at 5:00 PM

You may use up to four late days total
• Each late day is a “single” or “whole unit”
• 5 mins late = 1 late day
• Plan ahead, no other extensions
No Textbook

- Online lecture notes
- Resources posted on webpage
- Pay attention to lecture and section!
- Do assignments yourself!
Plan for Next 10 Weeks

1. FP, OCaml, ~5 weeks
2. OO, Scala, ~4 weeks
3. Logic, Prolog, ~1 week
Unfamiliar languages
+ Unfamiliar environments

Start Early!
No Compile, No Score
Forget Java, C, C++, …
... other 20\textsuperscript{th} century PLs

Don’t complain
... that OCaml is hard
... that OCaml is @!%@#

Immerse yourself in new languages
Free your mind.
• Programming Assignments done ALONE
• We use plagiarism detection software
  • Have code from all previous classes
  • MOSS is fantastic, plagiarize at your own risk
while(mis==ul){ //showing values of the multiple between the limits/
    printf("%10d\n",mis); //10 spaces for each value, right align*/
    if((mis%2)==1)="/Counting odds and evens*/
        even++;
        rsun+=mis;
        counter++; //add mult:
    if(mis%2)
        odd++;

    if(counter==6){//summing at end of each row*/
        printf("%10d\n",rsun);
        sum+=rsun; //running total sum by adding row sums*/
        rsun=0; //reset row sum for next row*/
    }
    while(counter <= 5 && (counter != 1)){//placing spaces to keeplignment*/
        printf(" ");
        counter++;
    }
    if(counter==6){//summing at end of the last row*/
        printf("%10d\n",rsun);
        sum+=rsun;
        counter = 1;\n    }

    printf("\n\nThere are %d odd and %d even numbers.\n",odd,even); //stating the
    printf("The sum of all numbers is: %d\n\n",sum); //showing the sum/
    /*resting the value for next time running through program*/
    even=0;
    odd=0;
    sum=0;
}

return 0;

printf("%10d\n",mis); //Keep track of how many even and odd multiples are displayed
    if(dishMult%2) //odd:
        odd++;
    else
        even++;
    rsun+=dishMult; //sum up all multiple values in the row*/
    counter++; //reset counter for next row*/
    dishMult+=1; //set display value to next multiple value*/
    /*do not display a multiple value of zero*/
    if(dishMult==0)
        dishMult=mult;
    /*display the sum of the row at the end of the row*/

    if(counter==6){
        printf("%10d\n",rsun);
        sum+=rsun; //add the sum of the row to the total sum:
        rsun=0; //reset row sum for new row*/
        counter=0; //reset counter for new row*/
    }
}

/*display blank spaces to keep table nice*/
while((counter==6) && (counter==1)) {
    printf(" ");
    counter++;
}
/*display the sum of the row at the end of the table*/
if(counter==6)
    printf("%10d\n",rsun);
    sum+=rsun;
    counter=0;
/*reset values for next repetition*/
    even=0;
    odd=0;
    sum=0;
}
return 0;
• Programming Assignments done ALONE
• We use plagiarism detection software
  • Have code from all previous classes
  • MOSS is fantastic, plagiarize at your own risk
• Zero Tolerance
  • offenders punished ruthlessly
• Please see academic integrity statement
• Click Fraud is also not allowed!
I WANT YOU
To Ask Me Questions!
void sort(int arr[], int beg, int end){
    if (end > beg + 1){
        int piv = arr[beg];
        int l = beg + 1;
        int r = end;
        while (l != r-1){
            if(arr[l] <= piv)
                l++;
            else
                swap(&arr[l], &arr[r--]);
        }
        if(arr[l]<=piv && arr[r]<=piv)
            l=r+1;
        else if(arr[l]<=piv && arr[r]>piv)
            {l++; r--;}
        else if (arr[l]>piv && arr[r]<=piv)
            swap(&arr[l++], &arr[r--]);
        else
            r=l-1;
        swap(&arr[r--], &arr[beg]);
        sort(arr, beg, r);
        sort(arr, l, end);
    }
}

let rec sort xs =
    match xs with
    [] -> []
    | h::t ->
    let (l,r) = List.partition ((<=) h) t in
    (sort l) @ h::(sort r)
Say Hello to OCaml

Readability matters...

```j
sort=:(((&:(<#@[),=#@[),(&:(>#@[))(~ ?@[#]))^: (1:<@#))
```

Quicksort in J
let rec sort xs = 
  match xs with 
  [] -> [] 
  | h::t -> 
    let (l,r) = List.partition ((<=) h) t in 
    (sort l) @ h::(sort r)

Quicksort in OCaml
Plan for Next ~5 Weeks

1) Fast Forward
   • Rapid introduction to what’s in OCaml

2) Rewind

3) Slow Motion
   • Go over the pieces individually
Introduction to OCaml
“Meta Language”  
Designed by Robin Milner  
To manipulate theorems & proofs

Several dialects:
• Standard ML (SML)
  • Original syntax
• Objective Caml: (OCaml)
  • “The PL for the discerning hacker”
  • State-of-the-art, extensive library, tool, user support
• F# (ML + .NET) released in Visual Studio
ML’s Holy Trinity

- Everything is an expression
- Everything evaluates to a value
- Everything has a type
Interacting with ML

“Read-Eval-Print” Loop (REPL)

Repeat:
1. System reads expression $e$
2. System evaluates $e$ to get value $v$
3. System prints value $v$ and type $t$

What are these expressions, values and types?
• Demo: `ocaml-top` on ieng6

• Extended demo in Section tomorrow

• We will collect data from your submissions to learn about usability of error messages, etc. in practice
  
  o More details to follow...
Complex expressions using “operators”: (why the quotes ?)

- +, -, *
- div, mod

Base Type: Integers
Complex expressions using “operators”: (why the quotes ?)

- Concatenation ^
Complex expressions using “operators”:

• “Relations”: =, <, <=, >=
• &&, ||, not
Untypable expression is rejected

- No casting, No coercing
- Fancy algorithm to catch errors
- ML’s single most powerful feature (why?)
1. Enter an expression $e$
2. ML infers a type $t$ or emits an error
3. ML evaluates expression $e$ down to a value $v$
4. Value $v$ is guaranteed to have type $t$
So Far, A Fancy Calculator...

... What do we need next?
Complex Type: Tuples (Products)

\[(2+2, 7>8)\] \rightarrow \text{int * bool} \rightarrow (4, \text{false})
Complex Type: Tuples (Products)

- Pairs, Triples, Quadruples, ...
- Nesting:
  - Everything is an expression
  - Nest tuples in tuples
Complex Type: Lists

- Unbounded size
- Can have lists of anything (e.g. lists of lists)
- But ...
Complex Type: Lists

All elements must have same type

[1; “pq”];
List operator “Cons” ::

1::[];
1::[2;3];
“a”::[“b”;“c”];

Can only “cons” element to a list of same type

1::[“b”;“c”];
Lists: “Cons”truct

Nil operator

Cons operator

\[
\begin{align*}
\text{Nil operator: } & \quad \emptyset, \quad \emptyset \rightarrow \emptyset \\
\text{Cons operator: } & \quad e_1 : T, \quad e_2 : T \text{ list}, \quad e_1 \cdot e_2 : T \text{ list} \\
& \quad \text{int list: } 1 : [2; 3], \quad [1;2;3] \\
& \quad \text{list: } [(\quad)] \Rightarrow [(\quad)] \\
& \quad \text{Nil operator: } \emptyset, \quad \emptyset \Rightarrow \emptyset
\end{align*}
\]
Complex Type: Lists

List operator “Append” @

- \([1;2]@[3;4;5]\);
- \(["a"]@["b"]\);
- \([]@[1]\);

\([1;2;3;4;5]\) int list
\(["a","b"]\) string list
\([1]\) int list

Can only append two lists...

... of the same type
List operator “Head”  

ML types can’t catch some errors though...
List operator “Tail” \texttt{List.tl}

The tail of empty list is a run-time error...

Expressiveness of type systems is an active area of research!
Lists: Deconstruct (or Destruct)

Head

\[
\begin{align*}
e & : T \text{ list} \\
hd \ e & : T \\
e & \Rightarrow v1::v2 \\
hd \ e & \Rightarrow v1
\end{align*}
\]

Tail

\[
\begin{align*}
e & : T \text{ list} \\
tl \ e & : T \text{ list} \\
e & \Rightarrow v1::v2 \\
tl \ e & \Rightarrow v2
\end{align*}
\]

\[(hd [[[]];[1;2;3]]) = (hd [[[]];[“a”]])\]

\[
\begin{align*}
e_1 : T \\
e_2 : T \\
e_1 = e_2 : \text{bool}
\end{align*}
\]

int list

string list
Recap: Tuples vs. Lists

What’s the difference?

• Tuples:
  - **Different** types, but **fixed** number:
    - pair = 2 elts
      - (3, “abcd”)  (int * string)
    - triple = 3 elts
      - (3, “abcd”, (3.5, 4.2))  (int * string * (float * float))

• Lists:
  - **Same** type, **unbounded** number:
    - [3; 4; 5; 6; 7]  int list

• Syntax:
  - Tuples = *comma*  Lists = *semicolon*