Lecture 1: Hello, World!

Ranjit Jhala
UC San Diego
A Programming Language

• Two variables
  - \( x, y \)

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

\[
\begin{align*}
L1: & \quad x++; \\
   & \quad y--; \\
   & \quad (y=0)? L2:L1 \\
L2: & \quad ...
\end{align*}
\]

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

Good luck writing quicksort

... or Windows, Google, Spotify!
So why study PL?

“A different language is a different vision of life”
- Federico Fellini
So why study PL?

Programming language shapes
Programming thought
So why study PL?

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
Goal: Enable you to Program

- Readable
- Correct
- Extendable
- Modifiable
- Reusable
Learn How To Learn
Goal: How to learn new PLs

No Java (C#) 15 (10) years ago

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

Re-learn the PLs you already know
To Design New Languages
Goal: How to design new PLs

...“who, me ?”

Buried in every extensible system is a PL

- Emacs, Android: Lisp
- Word, Powerpoint: Macros, VBScript
- Unreal: UnrealScript (Game Scripting)
- Facebook: FBML, FBJS
- SQL, Renderman, LaTeX, XML ...

Choose Right Language
Enables you 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
WTF?

\[ x = x + 1 \]
Imperative = Mutation
Imperative = Mutation

Bad!
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?

Google

MapReduce
So, Who Uses FP?

Linq, F#
So, Who Uses FP?

Erlang

facebook

Erlang
So, Who Uses FP?

Scala
So, Who Uses FP?

Wall Street
(all of the above)
So, Who Uses FP?

...CSE 130
Course Mechanics
Nothing printed, everything on Webpage!

http://ucsd-progsys.github.io/cse130/
Peer Instruction (ish)
Peer Instruction/Clickers

- Make class interactive
  - Help YOU and ME understand what's tricky

- **Clickers Not Optional**
  - Cheap ones are fine
  - 5% of your grade
  - **Respond to 75% questions**

- **Seating in groups** (links on Piazza)

- Bring laptop if you have one
In Class Exercises

1. **Solo Vote**: Think for yourself, select answer

2. **Discuss**: Analyze Problem in Groups
   - Reach consensus
   - Have questions, raise your hand!

3. **Group Vote**: Everyone in group votes
   - Must have same vote to get points

3. **Class Discuss**: Everyone in group votes
   - What was easy/hard?
Requirements and Grading

- The good news: No Homework
- In-Class Exercises: 5%
- Midterm: 30%
- Programming Assignments (7-8): 30%
- Final: 35%

Grading on a curve. Two hints/rumors:
1. Lot of work
2. Don’t worry (too much) about grade
No Recommended Text

• Online lecture notes

• Resources posted on webpage

• Pay attention to lecture and section!

• Do assignments yourself!
Suggested Homeworks

• On webpage after Thursday lecture

• Based on lectures, section of previous Tue, Thu

• Recommended, ungraded, HW problems are sample exam questions

• Webpage has first samples already
Weekly Programming Assignments

Schedule up on webpage

Due on Friday 5 PM

Deadline Extension:
- Four “late days”, used as “whole unit”
- 5 mins late = 1 late day
- Plan ahead, no other extensions
Plan

1. FP, Ocaml, 4 weeks
2. OO, Scala, 4 weeks
3. Logic, Prolog, 1 week
Weekly Programming Assignments

Unfamiliar languages

+ Unfamiliar environments

Start Early!
Weekly Programming Assignments

Scoring = Style + Test suite

No Compile, No Score
Weekly Programming Assignments

Forget Java, C, C++ ...  ... other 20th century PLs

Don’t complain  ... that Ocaml is hard  ... that Ocaml is @!%@#
Immerse yourself in new language

It is not.
Immerse yourself in new language

Free your mind.
Word from our sponsor ...

- Programming Assignments done ALONE

- We use plagiarism detection software
  - I am an expert
  - Have code from all previous classes
  - MOSS is fantastic, plagiarize at your own risk

- Zero Tolerance
  - offenders punished ruthlessly

- Please see academic integrity statement
while(mdis<=ul){ /* showing values of the multiple between the limits */
printf("%d\n",mdis); /* 10 spaces for each value, right align */
if((mdis%2)!=1) /* counting odds and evens */
  even++;
  rsum+=mdis;
  counter++;
if(mdis2)
  odd++;
}
if(counter>=6) /* summing at end of each row */
  printf("\n\n\rsum=%d\n",rsum);
  sum=0; /* running total sum by adding row sums */
if(counter>=5) /* reset row sum for next row */
  counter1=1; /* reset counter for next row */
while((counter <= 5) & (counter != 1)) /* placing spaces to keep alignment */
  printf(" ");
if(counter>=6) /* summing at end of the last row */
  printf("\n\n\rsum=\n",rsum);
  sum=0;
  counter1=1;
}

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 */
  even=0;
  odd=0;
  sum=0;
}

return 0;
I WANT YOU

To Ask Me Questions?
Say hello to OCaml

Quicksort in C

```c
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);
    }
}
```

Quicksort in Ocaml

```ocaml
let rec sort xs =
    match xs with [] -> [] |
    (h::t) ->
        let(l,r)= List.partition ((<=) h) t in
        (sort l)@h::(sort r)
```
Why readability matters...

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)
Plan (next 4 weeks)

1. Fast forward
   - Rapid introduction to what's in ML

2. Rewind

3. Slow motion
   - Go over the pieces individually
“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# (Ocaml+.NET) released in Visual Studio
ML’s holy trinity

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

“Read-Eval-Print” Loop

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?
Base type: Integers

Complex expressions using “operators”: *(why the quotes ?)*

- +, -, *
- div, mod
Base type: Strings

Complex expressions using "operators": (why the quotes ?)

• Concatenation ^
Base type: Booleans

true
false
1 < 2
"aa" = "pq"

(true) && (false) = false
(true) && (true) = true
false && (true) = false
true && (true) = true

Complex expressions using “operators”:
• “Relations”:    = , <, <=, >=
• &&, ||, not
Type Errors

Untypable expression is rejected

- No casting, No coercing
- Fancy algorithm to catch errors
- ML’s single most powerful feature (why ?)
Complex types: Product (tuples)

\((2+2, 7>8)\);

\((4, \text{false})\)

\(\text{int} * \text{bool}\)
Complex types: Product (tuples)

- Triples,…
- Nesting:
  - Everything is an expression
  - Nest tuples in tuples
Complex types: Lists

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

```
[1; "pq"];
```

All elements **must have same type**
Complex types: Lists

List operator “Cons”  ::

1::[];  [1]  int list
1::[2];  [1;2]  int list
“a”::[“b”;“c”];  [“a”;“b”;“c”]  string list

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

1::[“b”; “cd”];
Complex types: Lists

List operator “Append” @

- [1;2]@[3;4;5];
- [“a”]@[“b”];
- []@[1];

int list
string list
string list

Can only append two lists...

...of the same type
Complex types: Lists

List operator “head” \( \text{hd} \)

Only take the head a nonempty list

\[ \text{hd} [1;2]; \]
\[ \text{hd} (["a"]@["b"]); \]
Complex types: Lists

List operator “tail” \( \text{tl} \)

Only take the tail of nonempty list \( \text{tl} \ [\ ]; \)
Recap: Tuples vs. Lists?

What’s the difference?

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

• Lists:
  - Same type, unbounded number:
    - [3;4;5;6;7]  int list
So far, a fancy calculator...

... what do we need next?