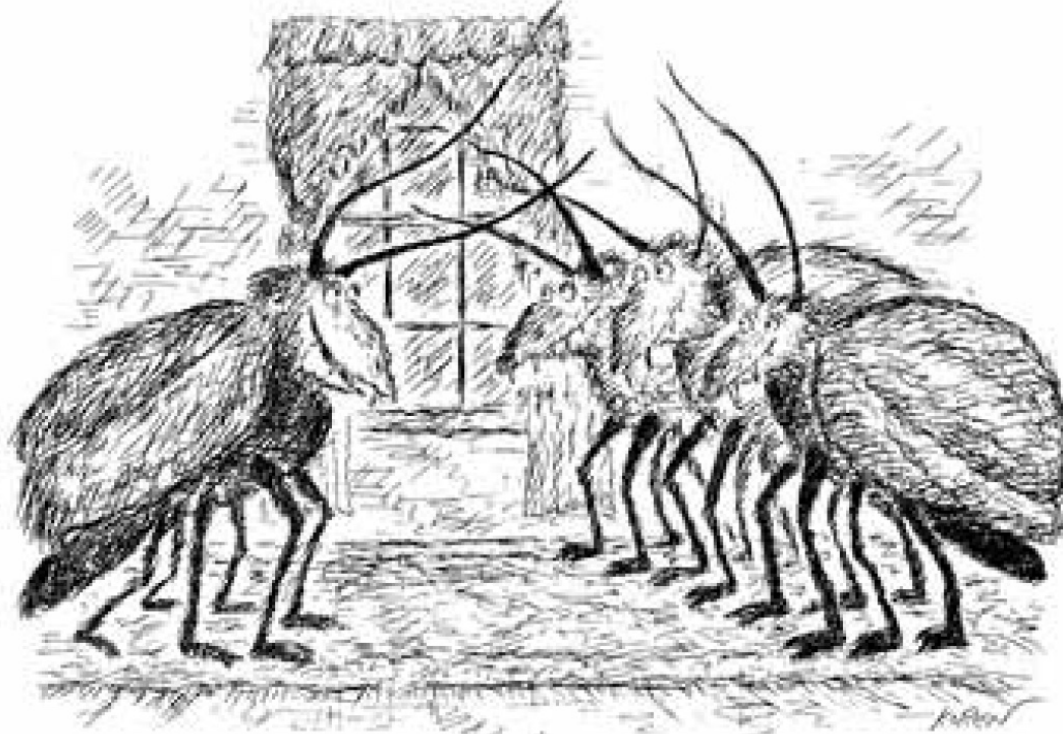


CS 32

Lecture 3: Reorganizing



“It’s a perfect day to reorganize those closets!”

Switch Gears

- Or horses in midstream
- Will be using Phill Conrad's course outline
 - Uses submit.cs uniformly
 - Labs about the same
 - Numerous small Programming Assignments
 - Coordinated with reading

First Lab

- Still will be the intro
- Will be an intro to me and our TAs too
- Let's all hack together to make it work

Where We Were

- More than one way to skin a cat
 - Object-Oriented Programming
 - Prototype-Based Programming
 - Functional Programming
 - Aspect-Oriented Programming

Truce

- Both Scala and Swift make point of allowing multiple kinds of programming
- This will be the trend for general-purpose languages

Going Deep

- Author subtitled “Language and Thought”
 - Which isn’t lightweight chatter
 - Mentions Edward Sapir
 - Famous (or notorious) linguist

Edward Sapir

- “Human beings do not live in the objective world alone...but are very much at the mercy of the particular language...for their society.”
- “We see and hear and otherwise experience as we do because [of] the language habits of our community.”

Grinds to Halt

- Author gives example of two different paradigms
 - One in FORTRAN
 - One in APL
 - Who followed that?
- Classic moral
 - Algorithms count
- Modern: C++ vs. Python

Language is All

- The Sapir–Whorf thesis
- Goes in and out of favor
- Right now it's out of favor...
- Oops, back in!

Left Turn?

- Author seems to tangent off into Church–Turing thesis
- What is it with these hyphenated theses?
 - A function is human–computable iff it is computable by Turing Machine
 - Bold statement about consciousness?

Strange Opposite

- Church–Turing says “All languages are essentially the same”
- Sapir–Whorf says “Every language is an island”
- Who will win?

The Nuts and Bolts

- OOP is useful in small doses
- Platforms provide larger trees of classes
- Ordinary people instantiate those classes, design their own small hierarchies
- Manic people design their huge hierarchies and never graduate

The Truce

- Those OOP hierarchies may themselves use
 - Functional programming
 - Logical programming (e.g. Prolog)
 - Generic programming
 - Assembly coding

Down to Business

- OK, we know inheritance is useful
- Let's get on already with what it does

Hierarchy aka Tree

- Types are no longer individual things
- They form a hierarchy
 - aka subset/containment relation
 - aka a Tree

Jump to Chapter 15?

- Inheritance doesn't appear until Chapter 15 of PSC++
 - Nodes, lists, stacks, queues intervene
 - Have we seen this in CS 24?

Go On In Reader

- Messages vs. function calls
 - What makes a method special?
 - It has a specific receiver
 - Function calls can mimic this by e.g. a default first parameter

Dynamic

- Meaning of a message depends on the runtime type of the receiver
- Runtime :: dynamic as
buildtime :: static
- Overused buzzwords too

Change

- Why would the runtime type of a variable be different to the compile-type?
 - `Animal* a;`
 - `a = new Feline();`
 - Why can't compiler tell?

Fear Change

- And what if the runtime type of a variable is incompatible with its compile-time type?
 - Crash
 - Do nothing
 - Get rerouted

Switch Gears Again

- CS 16 was Problem Solving
 - Mostly concerned with teaching C++
- Covered Chapters 1 – 10
 - What was all that?

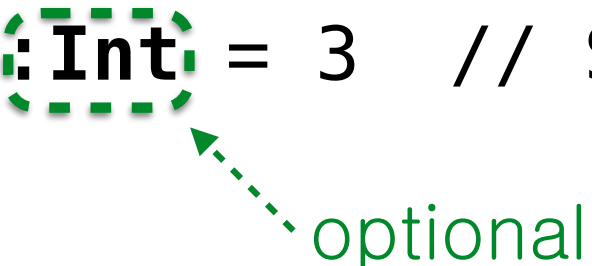
Recap of CS 16

- These are things you are expected to be familiar with already
 - Initializing variables
 - [...much more...]
 - Pointers

Initializing

- Declaring a variable
 - A high-level concept
 - Pointers involved somewhere
- Initializing a variable
 - Providing an explicit value

Initialize on the Spot

- `int i(1); // C++ initializer`
- `/* it's complicated */ // Java`
- `var count:Int = 3 // Swift`


Naming Things

- Two main problems in computing
 - Naming things
 - Caching things
 - Off-by one errors

Scramble

- Take a working program
- Permute its variable names
- Rebuild and...
- ...it still works!

Chaos

- In fact, any 1:1 mapping between variable names will preserve the program's behavior
- At most one variation will be unreadable
- Clearly there must be some better or worse choices
- But we all know this..., right...?

Single Words

- These are pretty easy
 - Nouns for object-y things
 - Verbs for process-y things

Collision

- Some words work as both nouns and verbs
 - Test
 - Query
 - Call
 - Set // and many more...

Double Trouble

- Compound variable names
 - `underscore_style`
 - `camelCaseStyle`
- What is a `SetQuery`?
 - Or a `CallTest`?

Seen in the Wild

- UpdateThread
 - A function that updates a Thread
 - A particular kind (Update) of Thread
 - A function that threads an Update
 - ???

It Got Worse

- UpdateThreadBackup
 - Too many interpretations
- And.. some variable names got swapped
 - Or otherwise out of sync

char broil

- The mysterious type char in C/C++
- 8 whole bits
 - aka one byte
- A number from 0 to 255... (00–FF)
- ...or from –128 to 127 (80–7F)

Rome Still Conquers

- Eight bits are enough to encode the entire Roman alphabet with room to spare
- The spare room got used in multiple, inconsistent ways

21st Century

- This will no longer hack it
- `wchar_t` typedef worked...
- ...for a while
- Some OS distributions give `wchar_t` just 16 bits

Best Practice?

- For char, use unsigned int
- Don't use char
 - If you do, it's a signed byte

Braces

- Always use them

```
if a < b {  
    // Code here...  
} else {  
    // Code here...  
}
```



```
if a < b  
    // Code here...  
else  
    // Code here...
```



```
if a < b { etc... }  
else     { etc... }
```

if you must

++ and — Operators

- Make C-style looping work
- `for (i = 0; i < N; i++)`
- Each operator has two modes
 - prefix `++i;`
 - suffix `i++;`

Expression v. Instruction

- `i++` can either be
 - An instruction: increment `i`
 - An expression: `j = i++;`
 - It's both at once!

Who's On First?

- $j = i++$
 - Give me the value of i then increment
- $j = ++i$
 - Increment i then give me the value

Used to Matter

- The distinction was important in getting performance out of certain processors
- The first thing you did after compiling was look at the assembly code your compiler emitted

Fading Away

- Swift started with +=, -=, ++, and — operators
- Swift 3 retired the ++ and —

switch statements

- The more your program resembles a state machine, the more it will have `switch` statements
- Gigantic `if` statement centered on one value
- Sign of weakness?

Deciding

- Every program will have to have decisions
- In modern languages, we decide with “dispatch”
 - OOP dynamic dispatch
- `switch` considered harmful?

Sidebar

- goto Considered Harmful
 - March 1968, Dijkstra
 - Wirth changed the title
- “goto Considered Harmful” Considered Harmful
 - March 1987, Rubin
- ““goto Considered Harmful” Considered harmful” Considered Harmful

switch harmful?

- Back in favor
- Swift has upgraded `switch` statements
 - Pattern matching
 - `where` clauses
 - Enum-aware

Scope

- Nesting function calls
 - Tracked with a LIFO (stack)
- Locals vanish when you exit scope
- But globals are frowned upon
- What to do?

Pointers

- Everybody's friend
 - Frenemy at best
- Pointer is “one level of indirection” away from the value
 - Need the flexibility
 - Hate the consequences

Down With Globals?

- Pointers usually point to
 - Strings
 - Vectors or arrays
 - Instances of classes
 - aka objects

Mr Big

- Very often there are classes or data structures that are unique
- Singletons
 - Can't live with them
 - Can't live without them

Read!

- Go on to Chapter 2 of Reader #2
- Start reading Chapter 11, 12 of CS 16 text PSC++
- Chapter 12 (searching and hashing) in CS 24 text DSC++