

Lecture 24: Commands & Control Structures

CSCI 131
Spring, 2011

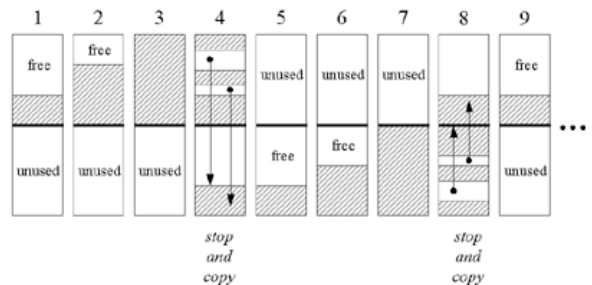
Kim Bruce

Back to Garbage Collection

Copying Collector

- Divide memory in half -- working vs. free
- When working exhausted
 - Copy live nodes from working to free (use forwarding address)
 - Swap halves
- Evaluation:
 - Only looks at live cells, but can be incremental
 - Needs twice as much space, but respects cache
 - Allocation very cheap! Always one big block free
 - GC fast if most are dead

Copying Collector



Memory as time passes ...

Diagram from Bill Venners, Inside the Java VM

Generational Collector

- Only try to collect recently allocated blocks
 - Infant mortality - majority of blocks die young!
- Divide memory into two or more generations.
- Modern Java uses copying collector for youngest and older uses mark-compact scheme
 - youngest gets lots of garbage quickly
 - mark-compact doesn't move lots of older objects

Commands

Commands

- Statements affect state
- Distinguish between state and environment
 - **State**: contents of memory (& I/O)
 - **Environment**: association b/n names and values (including locations)

Assignments

- Order of evaluation
 - $A[f(i)] := j * f(i) + j$
- Meaning of assignment
 - assignment by copying
 - assignment by sharing

Control Structures

- FORTRAN 1
 - GO TO n
 - GO TO (17, 43, 12, 99), I (also other variants)
 - IF(arith exp) 17, 43, 12
means go to statement number 17 if arith exp is negative, 43 if zero, and 12 if positive
 - DO label ivble = 1, 20, 2
- Close to machine code

Goto Statements

- Why need repetition - can do it all with goto's?
- "The static structure of a program should correspond in a simple way with the dynamic structure of the corresponding computation."
Dijkstra letter to CACM.

ALGOL 60

- GO TO 99
- IF ... THEN ... ELSE (hierarchical)
- for i := 3, 7, 11 step 1 until 16, i/2 while i >= 1, 2 step i until 32 do ..
 - BAROQUE, all expressions re-eval each time through loop:
 - 3, 7, 11, 12, 13, 14, 15, 16, 8, 4, 2, 1, 2, 4, 8, 16, 32.
- switch - like in C/C++/Java.

Pascal

- go to
- if .. then .. else
- for, while, repeat (confusion w/positive vs. negative exit)
- labelled case - Tony Hoare
 - clear & efficient
 - construct jump table,
 - optimize depending on size,
 - self-documenting.

More on Case

- Modula 2 improved by adding otherwise clause
- Haskell & ML's pattern matching is compiled into a case statement:


```
fun reverse l = case l of
  nil => nil |
  (h::rest) => (reverse rest)@[h];
```
- if-then-else as well

Ada

```
iteration specification loop
  body
end loop.
```

- where iteration specification can be:
 - while condition,
 - for vbl in discrete range
e.g. for i in 1..10 loop .. end loop
- exit and "exit when"

Refresher: Natural Semantics of Commands

$$\frac{(e, ev, s) \Rightarrow (v, s')}{(x := e, ev, s) \Rightarrow s'[loc:=v]} \quad \text{where } ev(x) = loc$$

$$\frac{(C_1, ev, s) \Rightarrow s' \quad (C_2, ev, s') \Rightarrow s''}{(C_1; C_2, ev, s) \Rightarrow s''}$$

$$\frac{(b, ev, s) \Rightarrow (true, s') \quad (C_1, ev, s') \Rightarrow s''}{(if\ b\ then\ C_1\ else\ C_2, ev, s) \Rightarrow s''}$$

If every statement returns a value then also return v from semantics

Semantics of While

$$\frac{(b, ev, s) \Rightarrow (false, s')}{(while\ b\ do\ C, ev, s) \Rightarrow s'}$$

$$\frac{(b, ev, s) \Rightarrow (true, s') \quad (C, ev, s') \Rightarrow s'' \quad (while\ b\ do\ C, ev, s'') \Rightarrow s'''}{(while\ b\ do\ C, ev, s) \Rightarrow s'''}$$

Notice similarity between.
 while E do C
 and
 if E then begin
 C;
 while E do C
 end

Iterators

- Abstract over control structures


```
for c : char in string_chars(s) do ...
- where
string_chars = iter (s : string) yields (char);
index : Int := 1;
limit : Int := string$size (s);
while index <= limit do
  yield (string$fetch(s, index));
  index := index + 1;
end;
end string_chars;
```

Implementing Iterators

- Just another object w/state in o-o language
- What about procedural?
- How can we retain state?
- Specific kind of coroutine.

When Good Programs Go Bad!

Handling Errors

- What happens when something goes wrong, e.g., with read from file.
- In C returns error condition, which is generally ignored.
- In more modern languages, throw exception, which must be handled or crash.

Exceptions

- Designed to handle unexpected errors.
- Exception handlers based on dynamic calls, not static scope.
- Allows program to recover from exceptional conditions, esp. beyond programmers control
- Can be abused!

Example Exceptions

- Arithmetic, array bounds, or I/O faults,
- Failure of preconditions
- Unpredictable conditions
- Tracing program flow in debugger

Exception Handling

- Ada:
 - raise exception_name;
 - handling:

```
begin
  C
exception
  when excp_name1 => C'
  when excp_name2 => C''
  when others => C'
```
- Java, C++ similar w/ “throw” & “try-catch”

Handling Exceptions

- When throw exception -- where look for handler?
 - Same unit? (Ada/C++/Java)
 - Calling unit? (Clu)
 - If not find, continue up call chain

After Handling ...

- (Ada/Java/ML/Haskell): Return from block
- PL/I: Resumption model: re-execute failed statement.
- Eiffel: Re-execute block where failure occurred
- ML & Java -- exceptions can take parameters

Haskell uses Monads

```
data Exn a = Oops      String
           | Answer a   deriving (Show)

instance Monad Exn where
  return a = Answer a  -- return :: a -> Exn a

  -- (>=) :: M a -> (a -> M b) -> M b
  (Oops s) >= f      = Oops s
  (Answer a) >= f    = f a

throw :: String -> Exn a
throw = Oops

catch :: Exn a -> (String -> Exn a) -> Exn a
catch (Oops l) h = h l
catch (Answer r) _ = Answer r
```

See Stone's ExcInterp.hs

Exceptions in Java

- Objects from subclass of Exception class

```
try {
    ...
} catch (ExcType ex) {
    ...
} catch (ExcType' ex) {...} ...
```
- If not caught, must declare. E.g.

```
public E pop() throws EmptyStackException {
    ... throw new EmptyStackException(); ...
}
```

RuntimeException

- Exceptions w subclasses of RuntimeException need not be declared in method headers
- Ex.:
 - NullPointerException,
 - ArrayIndexOutOfBoundsException,
 - IllegalArgumentException,
 - NumberFormatException, and ArithmeticException
- Unfortunately, also includes EmptyStackException

If Exception Not Handled

- Pop off activation records while searching for handler.
- What if allocated memory in unit being popped?
- OK if garbage collection, but ...
- Closing files also problems

Java try-catch-finally

```
try {
    ...
} catch (ExcType ex) {
    ...
} catch (Exc'Type ex) {
    ...
} finally {... }
```