Scope, Function Calls and Storage Management

Lecture 12

Example: Return fctn with private state

ML

```
fun mk_counter (init : int) =
    let   val count = ref init
          fun counter(inc:int) =
              (count := !count + inc; !count)
    in
          counter
    end;
val c = mk_counter(1);
c(2) + c(2);
```

- Function to “make counter” returns a closure
- How is correct value of count determined in c(2) ?

Example: Return fctn with private state

JS

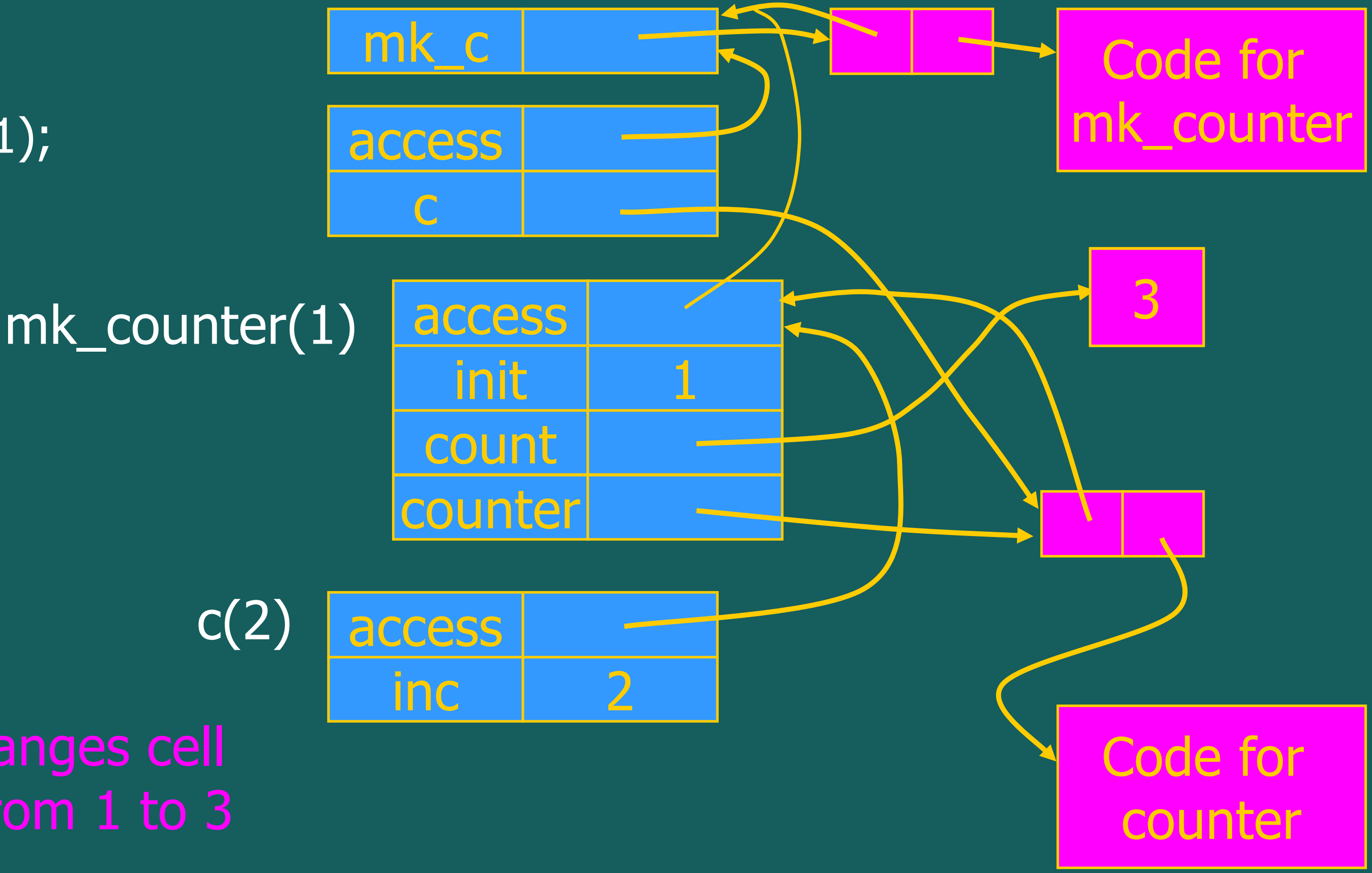
```
function mk_counter (init) {
    var count = init;
    function counter(inc) {count=count+inc; return count};
    return counter};
var c  = mk_counter(1);
c(2) + c(2);
```

Function to “make counter” returns a closure

How is correct value of count determined in call c(2) ?

Function Results and Closures

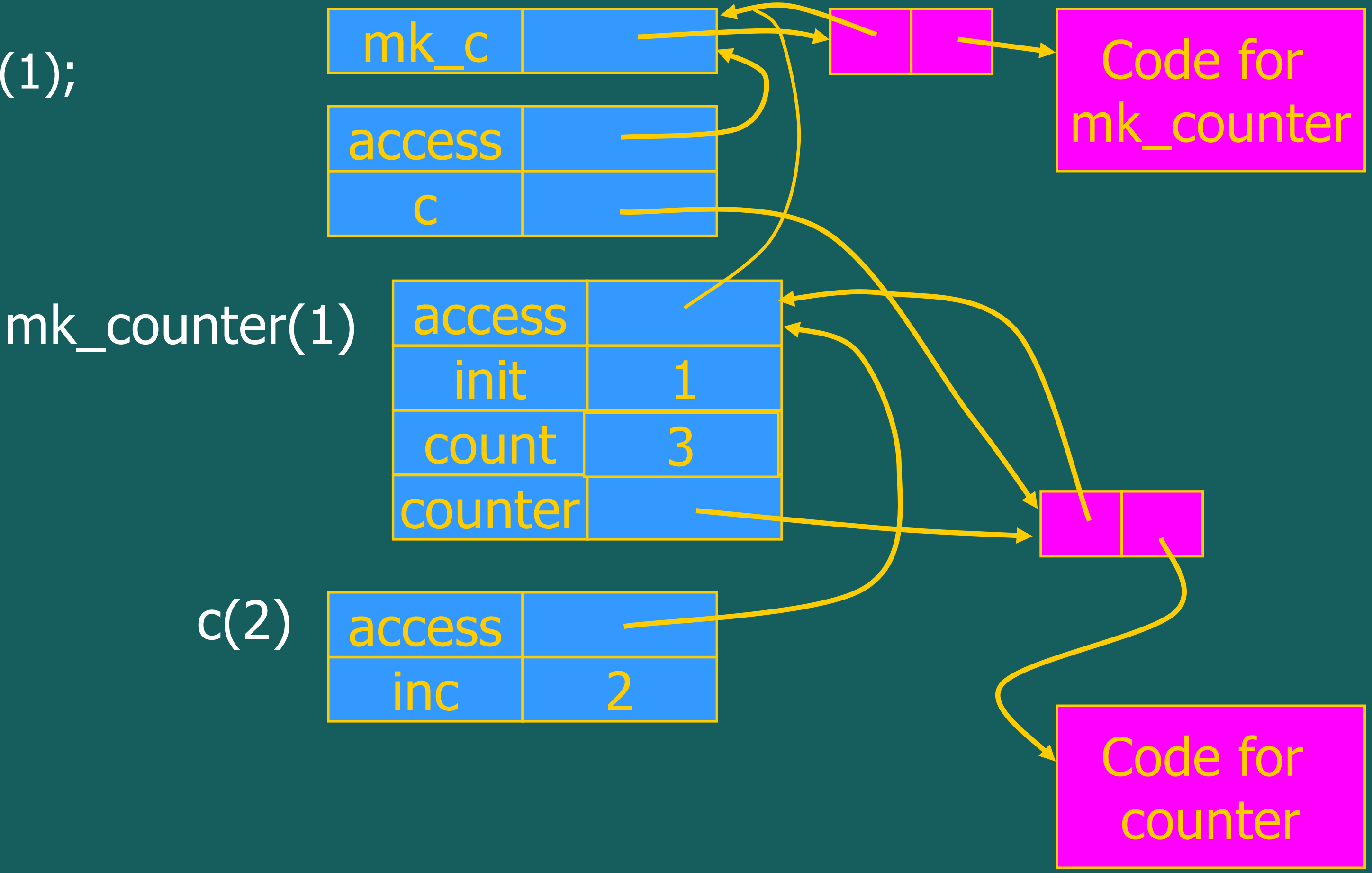
```
fun mk_counter (init : int) =
   let val count = ref init
       fun counter(inc:int) = (count := !count + inc; !count)
       in  counter end
   end;
val c = mk_counter(1);
c(2) + c(2);
```



Call changes cell value from 1 to 3

Function Results and Closures

```
function mk_counter (init) {
    var count = init;
    function counter(inc) {count=count+inc; return count};
    return counter};
var c  = mk_counter(1);
c(2) + c(2);
```


mk_c
access
c
mk_counter(1)
access
init
1
count
3
counter
c(2)
access
inc
2
Code for mk_counter
Code for counter