Structs and Vectors in DSSL2

EECS 214, Fall 2017
Welcome to DSSL2

- A Racket-based language, like BSL and ISL from EECS 111
- But made especially for you
DSSL2 expressions

3 + 5
DSSL2 expressions

3 + 5

6 * (3 + 5)

1 + strlen('hello')
DSSL2 statements

let x = 5

8 * x
DSSL2 statements

let x = 5

8 * x

if condition:
    do_some_stuff()
else:
    do_other_stuff(x, y, z)
DSSL2 functions

# hypotenuse: Number Number -> Number
# Finds the length of the hypotenuse.
def hypotenuse(a, b):
    sqrt(a * a + b * b)
DSSL2 functions

# hypotenuse: Number Number -> Number
# Finds the length of the hypotenuse.
def hypotenuse(a, b):
    sqrt(a * a + b * b)

# fact: Natural -> Natural
# Computes the factorial of `n`.
def fact(n):
    if n == 0: 1
    else: n * fact(n - 1)

assert_eq fact(5), 120
Vectors

\[
\begin{array}{cccccccccc}
0 & 1 & 1 & 2 & 4 & 7 & 13 & 24 & 44 & 81 \\
\end{array}
\]

\[
\left[ 0, 1, 1, 2, 4, 7, 13, 24, 44, 82 \right]
\]
Vector operations

let v = [ 0, 1, 1, 2, 4, 7, 13, 24, 44, 82 ]
Vector operations

```rust
define vector v = [ 0, 1, 1, 2, 4, 7, 13, 24, 44, 82 ]

test 'vector basics':
    assert_eq v[3], 2
    assert_eq v[6], 13
```
Vector operations

let v = [ 0, 1, 1, 2, 4, 7, 13, 24, 44, 82 ]

test 'vector basics':
  assert_eq v[3], 2
  assert_eq v[6], 13

test 'vector set':
  v[6] = 23
  assert_eq v[6], 23
What if I want a really big vector?

[ 0; 1000000 ]
Example: average

# average: Vector<Number> -> Number
# Averages the elements of a non-empty vector.
def average(vec):
    sum(vec) / len(vec)
Example: average

# average: Vector<Number> -> Number
# Averages the elements of a non-empty vector.
def average(vec):
    sum(vec) / len(vec)

# sum: Vector<Number> -> Number
# Sums the elements of a non-empty vector.
def sum(vec):
    let result = 0
    for v in vec:
        result = result + v
    return result
**Structs**

\[
\begin{aligned}
&x \quad 3 \\
&y \quad 4
\end{aligned}
\]

\[
\text{defstruct posn}(x, y)
\]
Structs

\[
\begin{array}{ccc}
  x & 12 & x \quad 3 \\
  y & -5 & y \quad 4 \\
\end{array}
\]

\[
\begin{array}{ccc}
  x & 0 & x \quad 0 \\
  y & 0 & y \quad 0 \\
\end{array}
\]

defstruct posn(x, y)

posn { x: 12, y: -5 }
posn { x: 0, y: 0 }
posn(3, 4)
Working with structs

defstruct posn(x, y)

let p = posn(3, 4)
assert posn?(p)
assert_eq p.x, 3
assert_eq p.y, 4

p.x = 6
assert_eq p.x, 6
assert_eq p.y, 4
defstruct employee(id, name, position)

let employees = [ employee(928, "Alice", 4),
                     employee(1089, "Bob", 6),
                     employee(14, "Carol", 6),
                     employee(546, "Dave", 6) ]
defstruct employee(id, name, position)

let employees = [
  employee( 928, "Alice", 4),
  employee(1089, "Bob", 6),
  employee( 14, "Carol", 6),
  employee( 546, "Dave", 6),
]

Suppose we want to find out Carol’s position:
defstruct employee(id, name, position)

let employees = [ 
    employee(928, "Alice", 4),
    employee(1089, "Bob", 6),
    employee(14, "Carol", 6),
    employee(546, "Dave", 6),
]

Suppose we want to find out Carol’s position:

employees[2].position

How can we give her a promotion (from 6 to 5)?
defstruct employee(id, name, position)

let employees = [
    employee(928, "Alice", 4),
    employee(1089, "Bob", 6),
    employee(14, "Carol", 6),
    employee(546, "Dave", 6),
]

Suppose we want to find out Carol’s position:

employees[2].position

How can we give her a promotion (from 6 to 5)?

employees[2].position = 5
Generalizing

```python
# promote-employee : Vector<Employee> Natural ->
# Decrements the position of the `index`th employee.
def promote_employee(employees, index):
    let emp = employees[index]
    emp.position = emp.position - 1
```
For more DSSL2 information

See the DSSL2 reference (or help desk)
Next time: The lowly linked list