CSC 447 - Concepts of Programming Languages

L-Values

Instructor: Eric J. Fredericks

Learning Objectives

❓ When do we read and when write to a memory location?

- Distinguish l-values from r-values

Variables

- Given declaration:

```
1 var x:Int = 0
```

- What does `x` mean below?

```
1 x = x + 1
```

- Right-hand `x` denotes
  - value *read* from storage location
- Left-hand `x` denotes
  - the storage location (address)
- Goes back to Strachey in the 1960s
  - Fundamental Concepts in Programming Languages, Christopher Strachey (1967)

L-Values

- Expression for which l-mode evaluation succeeds
- Effectively, *has an address*
- In C, l-value has address:

```c
int x = 5;
int y = 6;
int *p = &x;
int *q = &y;
int **r = &p;
r = &q;
q = p;
**r = 7;
```

- In Scala, we can assign to l-values

```scala
val x = 5
val l = List(1, 2)
val (a, b) = (1, 2)
val y :: z :: Nil = l
```

Not L-Values

- L-mode evaluation sometimes disallowed
- In C, cannot take address of temporary expressions

```
1 int x = 5;
2 int y = 6;
3 int *p = &(x + y); /* not allowed */
4 (x + y) = 7;       /* not allowed */
```

- `(x + y)` not an l-value
  - although it might be stored temporarily

Further L-Values

- L-values may require r-mode evaluation
- Array access in C, Java, etc.

```
1 arr[n + 2] = 7;
```

- Field access in Java, Scala, etc.

```
1 obj.f1 = 7
```

Summary

L-values

- Write to a memory location
- C: take address `&x`
- Scala, Java: assignable

```
1 val x = 5
2 val y :: tail = List(1, 2, 3)
```

- May need r-value `arr[1+2]`

R-values

- Read value at a memory location
- C, Scala, Java: expressions `x`, `1`, `x+y`, ...

- Specialized uses of *l-value / r-value*