

CSC 347 - Concepts of Programming Languages

L-Values

Instructor: Stefan Mitsch



Learning Objectives

- ❓ When do we read and when write to a memory location?
 - Understand the difference between l-values and r-values



Variables

- Given declarations such as:

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

- What does `x` mean in?

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



R-Mode and L-Mode

- Consider

```
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, we can take the address of l-values

```
1 int x = 5;  
2 int y = 6;  
3 int *p = &x;  
4 int *q = &y;  
5 int **r = &p;  
6 r = &q;  
7 q = p;  
8 **r = 7;
```

- In Scala, we can assign to l-values

```
1 val x = 5  
2 val l = List(1, 2)  
3 val (a, b) = (1, 2)  
4 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: refers to a memory location
- R-value: value of a memory location
- Specialized uses of *l-value* / *r-value*