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:

```
int x = 0;
```

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

```
(define x 0)
```

- What does `x` mean in?

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

```
(set! x (+ x 1))
```

R-Mode and L-Mode

- Consider

```
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, take the address of l-values

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

Not L-Values

- L-mode evaluation sometimes disallowed
- In C

```
int x = 5;
int y = 6;
int *p = &(x + y); /* not allowed */
(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.

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

- Field access in Java, Scala, etc.

```
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*