

# Expressions

Last updated 8/20/20

# Expressions

- These slides introduce expressions
- Upon completion: You should be able to interpret and code using these expression types

# Expressions

- Expression: Sequence of Operators and Operands that reduce to a single value
  - Simple and Complex Expressions
  - Subject to Precedence and Associativity
  - Six categories
    - Primary
    - Postfix
    - Prefix
    - Unary
    - Binary
    - Ternary

# Expressions

- Simple Expressions
  - Only 1 operator
  - $a + b$
- Complex Expressions
  - Multiple operators
  - $2 * 3 / 6$

# Expressions

- Primary Expressions

- One operand and no operators

- Name

a            interest\_rate            RATE            initial1

- Literal

2    123.456                    'a'            "hello ee1910"

- Parenthetical

Anything in parentheses reduces to a single value

(2 + 3 \* 4)            (a = b + c)

# Expressions

- Postfix Expressions
  - One operand followed by one operator
    - Operand must be a variable
  - Function Call

Function name is an operand (named entity)  
Parenthesis are the operator  
printf(...)
  - Postfix increment/decrement
    - $i++ \rightarrow i = i + 1$
    - $j-- \rightarrow j = j - 1$

# Expressions

- Postfix Expressions
  - Some expressions have a Value and a Side Effect

```
int j;
```

```
int x;
```

```
j = 5;
```

```
x = j++;
```

Value: `x = 5`

Side Effect: `j = 6`

Consider `printf("%d", j++);`

5 or 6 ?

- **Postfix indicates to operate after the evaluation**

# Expressions

- Prefix Expressions
  - One operator followed by one operand
    - Operand must be a variable
  - Only 2 examples
  - Prefix increment/decrement
    - $++j \rightarrow j = j + 1$
    - $--k \rightarrow k = k - 1$



# Expressions

- Prefix Expressions
  - Some expressions have a Value and a Side Effect

```
int j;
```

```
int x;
```

```
j = 5;
```

```
x = ++j;
```

Value: x = 6

Side Effect: j = 6

Consider printf(“%d”, ++j);

5 or 6 ?

- Prefix indicates to operate before the evaluation

# Expressions

- Unary Expressions
  - One operator followed by one operand
    - Operand can be any expression

+, -

cast

sizeof(int). ...

a = 5

+a -> +5

-a -> -5

Note: the expression is modified  
not the variable, a = 5 in both cases

# Expressions

- Binary Expressions
  - Operand operator operand
    - Familiar to us: +, -, \*, /
    - New: % - modulus (remainder of a division)
    - Subject to type limitations

$10 * 2 \rightarrow 20$

$10 / 2 \rightarrow 5$

$\text{true} * 2 \rightarrow 2$

$\text{true} / 2 \rightarrow 0$  ???

$'A' * 2 \rightarrow 130$

$'A' / 2 \rightarrow 32$  ???

$15.6 * 2 \rightarrow 31.2$

$15.6 / 2 \rightarrow 7.8$

$3 / 5 \rightarrow 0$

$12 / 5 \rightarrow 2$

$3 \% 5 \rightarrow 3$

$12 \% 5 \rightarrow 2$

Modulo only operates on integers

# Expressions

- Binary Expressions
  - Special binary expression - assignment
    - `variable = expression`
    - Has both a value - result of right side
    - And a side effect – places value into the variable on the left side

- Simple

`a = b + c`                      `j = j * 2`

- Compound

`*=, /=, +=, -=, %=`

`a *= b` → `a = a * b`

`a += 10` → `a = a + 10`

`a -= b + c` → `a = a - (b + c)`

expression is evaluated first

# Expressions

- Ternary Expressions