Announcements!

- Homework 0a due today!
  - ➤Email it to your TA
  - ➤No Word documents
- Homework 0b due on Sunday/Monday
  - ➤Electronic turnin by 10pm Sunday
  - ➤Paper turnin during lecture Monday

Announcements!

- We need a note taker who is enrolled in section AE
  - ➤Very easy—just need to copy the notes you already take
  - ➤See Isaac about this if you're interested

CSE 142
Computer Programming I

Arithmetic Expressions

Outline for Today

- Integer expressions
- Expressions with doubles
- Mixing types

Assignment Statement: Review

```
int length, width, area;
…
area = length * width;
```

assignment statement

expression

What is an Expression?

- Something that has a value
  - ➤A variable: `length`
  - ➤A constant: `1067`
  - ➤These can be put together with operators to make new expressions

Operator Overview

- Operators put expressions together to make more complicated expressions
  - ➤Operators take the values of the expressions and compute a new value
  - ➤The results depend on the types of the expressions
  - ➤Unary and binary operators

Unary and Binary Operators

- Unary operators take *one* operand

```
-1067
```

- Binary operators take *two* operands

```
3 * 17
12 - 15
```

- Most C operands are unary or binary

Multiple Operators

- What if we have something like

```
4 + 16 / 4
```

- These are still binary operators
- Need Order of Operations

Order of Operations

- Sometimes called Operator Precedence
- A rule that says what operators get executed first
- Disambiguates expressions
  - ➤What is the value of

```
4 + 16 / 4
```


Order of Operations

- Do ( )'s first
- Then do unary - (negation)
- Then do "multiplicative" ops: * / %
- Then do "additive" ops: + -
- What is the value of

```
4 + 16 / 4
```


Associativity

- What if things have the same level?

```
5 + 6 - 7 + 32
```

- C executes from left to right within the same precedence level
- Called left associativity

Use Parenthesis!

- C has about 50 operators, and roughly 15 precedence levels
- (almost) nobody remembers them all
  - ➤Isaac doesn't
  - ➤Your TAs don't
  - ➤Dennis Ritchie probably does
- Parenthesis force the order of operations, but also make code clearer, so use extra ones!

Use Parenthesis

- Which is clearer?

```
a * b + c * d + e * f
```

or

```
(a * b) + (c * d) + (e * f)
```

- Both mean the same thing

Integer Division

- This makes sense
  20 / 4
- What does this mean?
  17 / 4
- This is called integer division

Integer Division

- Remember—Integers cannot hold fractional numbers!
- If you do an integer division you get the integer part of the division
  - ➤`16 / 4` ➜ `4`
  - ➤`11 / 4` ➜ `2`
  - ➤`17 / 4` ➜ ???
  - ➤`199 / 200` ➜ ???

The Modulus Operator

- When we do an integer division we only get the integer part of the result
- The `%` operator gets the remainder
  - ➤`16 % 4` ➜ `0`
  - ➤`11 % 4` ➜ `3`
  - ➤`17 % 4` ➜ ???
  - ➤`199 % 200` ➜ ???

Doubles

- We use doubles to represent floating point values
- All of the things we just learned about precedence and associativity still apply
- Most operators still work
  - ➤ `+ - / *`

Floating Point Division

- Floating point division does not drop the remainder
  - `16 / 4 ➜ 4.0`
  - `11 / 4 ➜ 2.75`
  - `17 / 4 ➜ 4.25`
  - `199 / 200 ➜ 0.995`
- Cannot use the modulus operator
  - `11 % 4` is an error!

Why Use ints?

- Doubles seem to be better than ints…
  - They can store fractional numbers
  - They can store larger numbers
- Doubles may store numbers imprecisely
  - `3.0*15.0*(1.0/3.0)` might be 14.999999
- Rule: If we want an integer, use an int

Assigning Ints to Doubles

- What happens when we try to put an int into a variable of type double?

```
double value;
…
value = 6;
```

- What's in **`value`**?

Assigning Doubles To Ints

- What happens when we try to put a double into a variable of type int?

```
int value;
…
value = 3.14159;
```

- What's in **`value`**?

Truncation and Promotion

- When we make a double into an integer, the value is truncated
- When we make an integer into a double we say that the type has been promoted
- This can happen within expressions as well!

Mixing Doubles and Ints

- What happens if we have doubles and ints in the same expression?
- Always promote if there's a problem

```
double foo;
foo = 3.0 + (3 / 1.5);
```

- What value is in **`foo`**?

Mixing Doubles and Ints

- When we mix integers and doubles, the integers are promoted *at the last possible moment!*
- Examples
  - ➤4.0 * (3 / 4.0) =
  - ➤4.0 * (3 / 4) =

An Example

- What's wrong here:

```
int length, area;
double width;
...
width = area / length;
```


Explicit Conversions

- We need to explicitly convert the types
- This conversion is called a cast
- Format:

```
(type) expression
```

- Examples:

```
(double) myage;
(int) area;
```


The Example Revisited

- We fix the example by using a cast:

```
int length, area;
double width;
...
width = ((double) area)  /  ((double) length);
```

- Could we use less casts?


```
int foo, answer;
double bar;
foo = 2;
bar = 4.5;
answer =
     (foo*foo + bar*2) + 3/(double)foo;
```


Types Matter in C

- The result of computation can differ depending on the type
- Always know what types you have
- There are cases in which you MUST use the right types!

Important Lessons

- Write clearly
- Break up complicated expressions
- Use parenthesis to make execution happen the way you want *and* to make things easier to read
- Use explicit casts when you need them
- Be aware of types