

Assignment & Mathematical Expressions

What can you do with a variable?

iClicker Attendance

Please click on A if you are here:

A. I am here today.

Exercise Review: Mass of Atoms

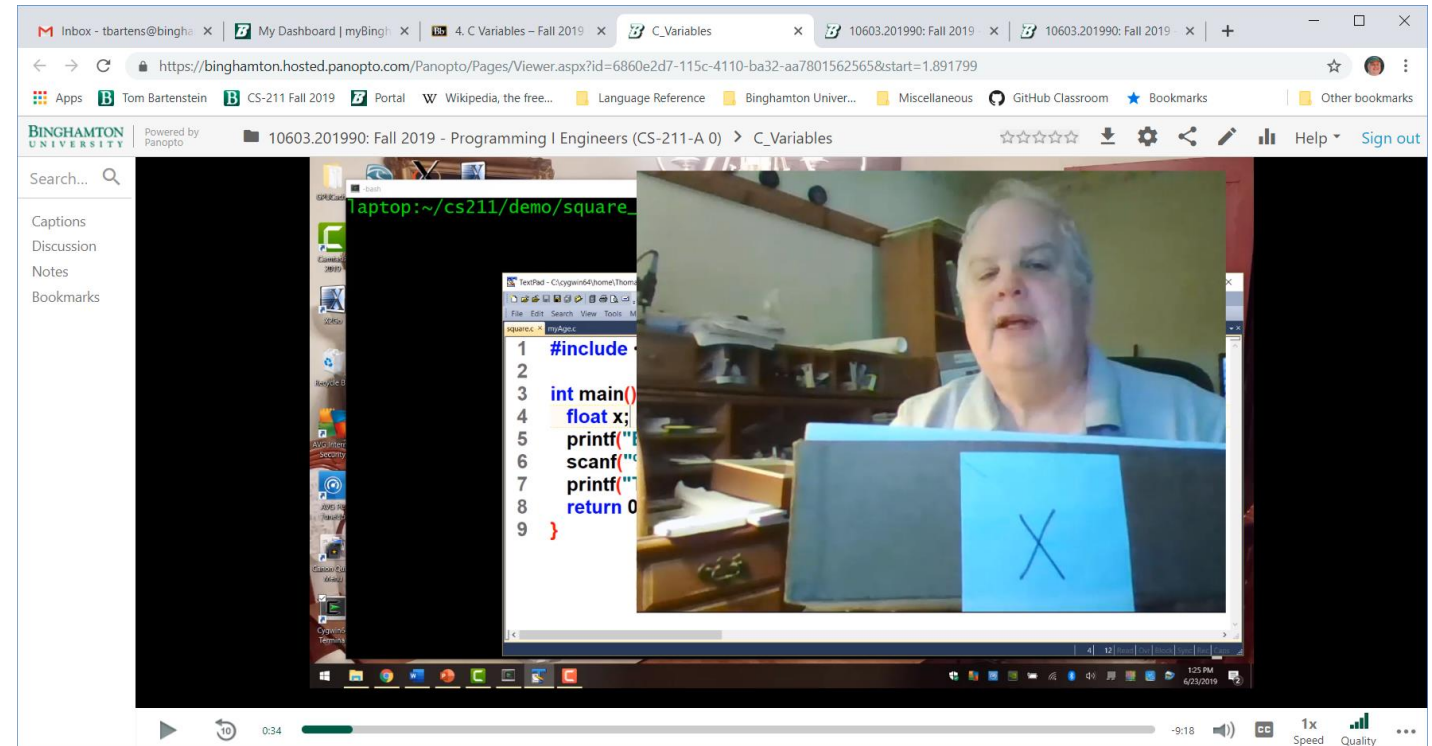
A hydrogen atom consists of one proton and one electron. If the mass of an electron is approximately 9.1×10^{-31} kilograms, and the mass of a proton or a neutron is approximately 1.7×10^{-27} kilograms, what is the mass of a hydrogen atom?

A typical carbon atom contains 6 electrons, 6 protons, and 6 neutrons. What is the mass of a carbon atom?

Write a generalized program that asks for the element name, number of protons, neutrons, and electrons, and prints it's mass.

Video Review: C Variables

- Variable Attributes: name, type, value, location
- Declaring a variable
- Uninitialized Variables
- Declare with Initializations
- Questions???



Class Exercise: Declares

- Write a declare statement for a variable to contain the number of electrons in an atom of a specific element.
- In a separate statement, assign the number of electrons in a helium atom to your variable.

Video Review: C Built-In Types

- “int” integer type
- “float” floating point type
- “char” ASCII type
- Questions???

The screenshot shows a web browser window displaying a video player. The video player interface includes a search bar, a list of video controls (Captions, Discussion, Notes, Bookmarks), and a video player with a progress bar. The video content is a slide titled "C Built-In Types : int" from Binghamton University, CS-211 Fall 2019. The slide lists the following points:

- Whole Numbers (e.g. 32, -16, 0, 45021, ...)
 - Positive or negative or zero
- Limits on largest (and smallest) number that “fits”
 - -2,147,483,648 up to 2,147,483,647
- 32 bits (4 bytes) - other built-in integer types (e.g. short, long, char) have different sizes and different limits

Below the text, there is a number line diagram illustrating the range of integers:

Number line: -4, -3, -2, -1, 0, 1, 2, 3, 4

Arrows point from the line to three boxes:

- negative integers (covering -4 to -1)
- zero (covering 0)
- positive integers (covering 1 to 4)

Class Exercise : Integer Literals

- What are the decimal values and internal representations of the following variables:

```
int x=73;
```

```
int y=0b11111111111111111111111111111111; // 32 bits of “1”.
```

```
int z=0x000000AD;
```

```
char w=0377;
```

Class Exercise: Float Literals

- How would you declare a float variable in C with the initial value of:
 - The mass of a proton: $\sim 1.7 \times 10^{-27}$ Kg ?
 - The number of sides of a pentagram?
 - Earth's Gravity (metric) : ~ 9.81 meters/sec² ?
 - Avagadro's number : $\sim 6.022 \times 10^{23}$ atoms/mole?

Class Exercise : Integer Representation

- What is the internal representation of the following variable:

```
int x=73;
```

Class Exercise: Float Representation

- What is the internal representation of the following variable:

```
float fx=19.75;
```

Class Exercise: Char Representation

- What is the internal representation of the following variable:

```
char letter='q';
```

iClicker Question

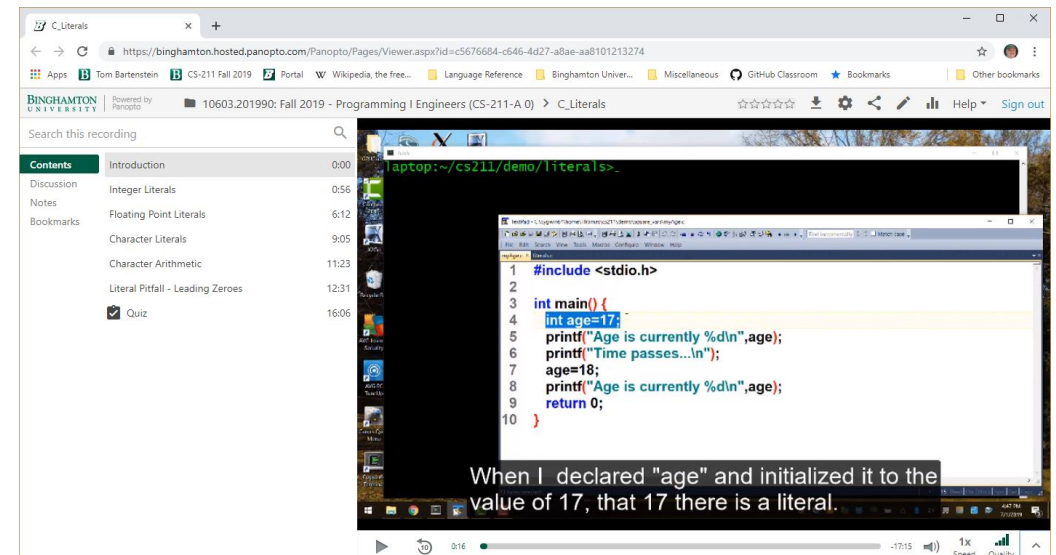
- What is the decimal value of num?

`char num='G' - 'A';`

- A. No value – compiler error.
- B. '@'
- C. 6
- D. 71

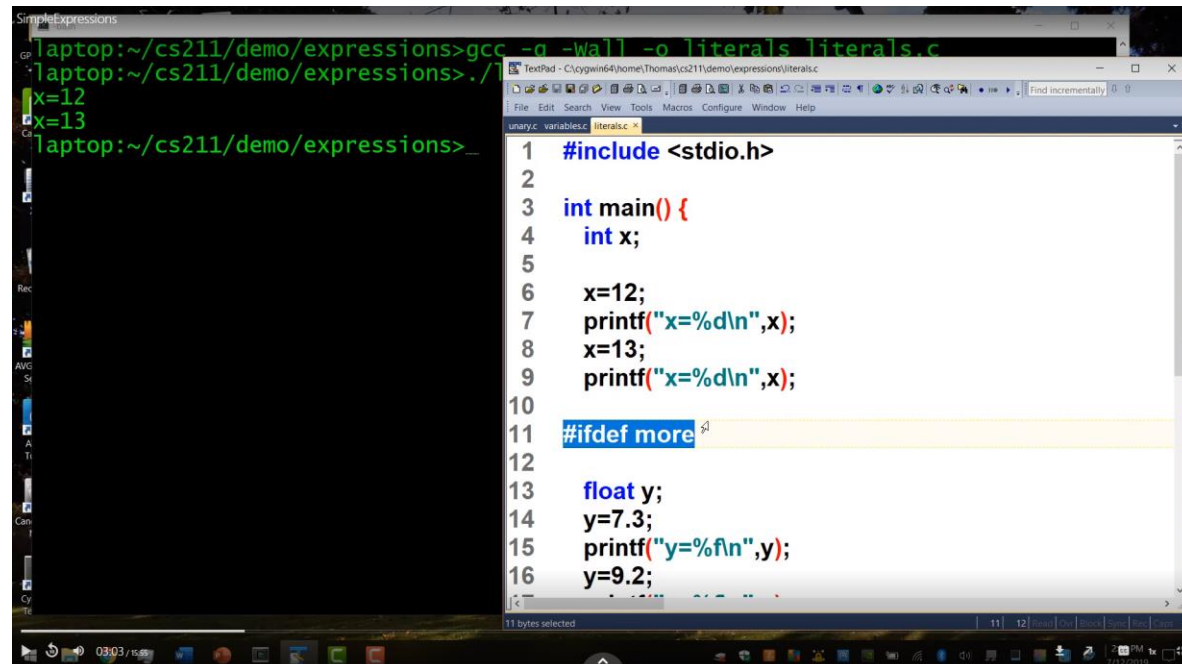
Video Review: C Literals

- Integer Literals
 - Decimal, Hexadecimal, Octal, and Binary specification
- Floating Point Literals
 - Decimal and Scientific Notation representation
- Character Literals
 - Numeric and Character representation



Simple Expressions

- Watch the video, available on myCourses
 - Content
 - Videos
 - 5. Mathematical (Numeric) Expressions

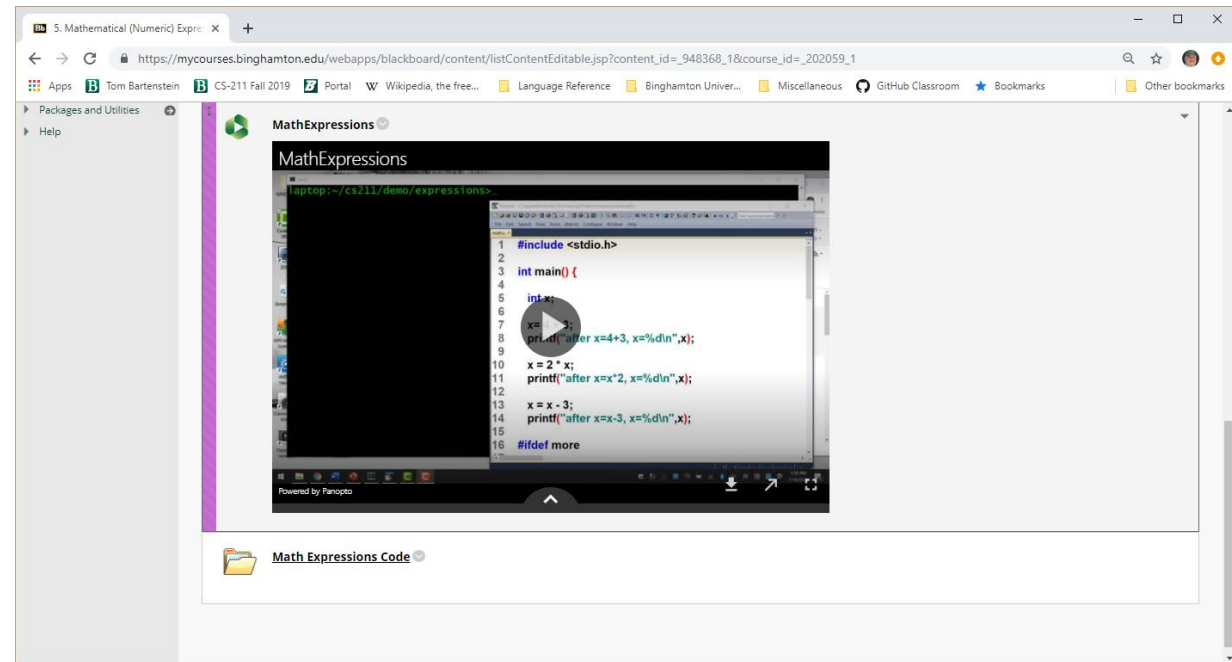


```
laptop:~/cs211/demo/expressions>gcc -g -Wall -o literals literals.c
laptop:~/cs211/demo/expressions>./literals
x=12
x=13
laptop:~/cs211/demo/expressions>_
```

```
1 #include <stdio.h>
2
3 int main() {
4     int x;
5
6     x=12;
7     printf("x=%d\n",x);
8     x=13;
9     printf("x=%d\n",x);
10
11 #ifdef more
12
13     float y;
14     y=7.3;
15     printf("y=%f\n",y);
16     y=9.2;
```

Math Expressions

- Watch the video, available on myCourses
 - Content
 - Videos
 - 5. Mathematical (Numeric) Expressions



Exercise: Ball Motion

Assume you throw a (frictionless) ball straight up in the air at some initial velocity. With gravity acting at 9.81 m/s^2 , after time t , what is the height and velocity of the ball?

Physics: $v(t) = v_0 - gt$ and $h(t) = v_0 t - \frac{gt^2}{2}$

Q: What do you need to know? v_0 and t

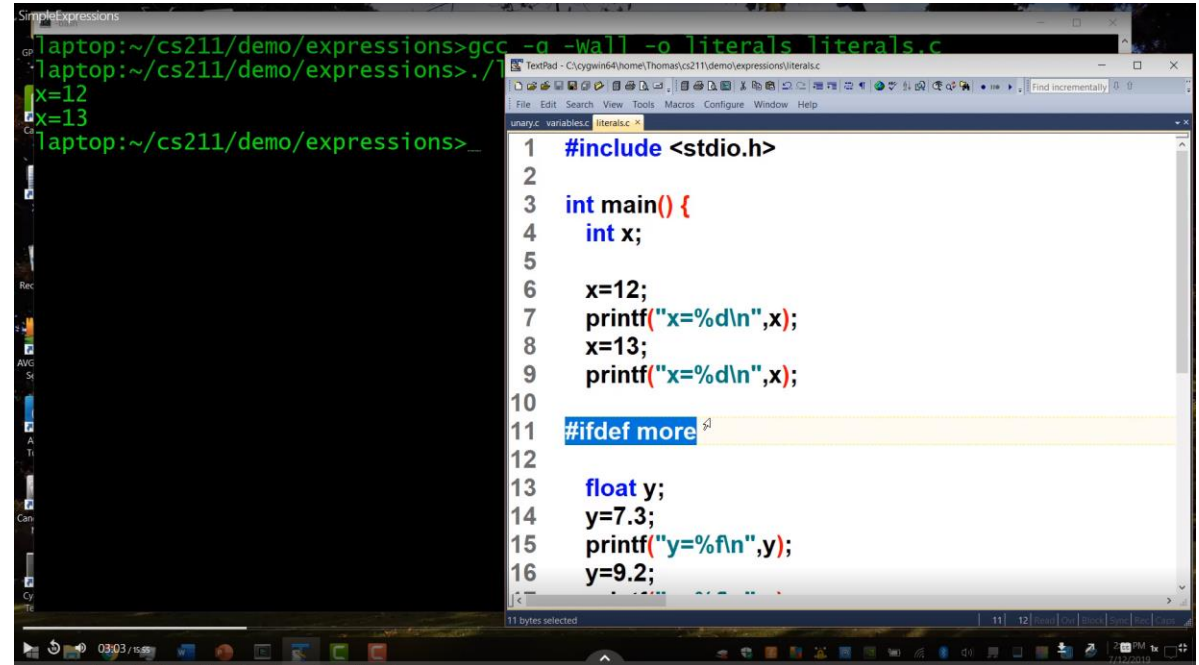


Exercise assumptions

- Express velocity in meters per second, with + being up, and – being down.
- Express location (height) in meters, where the ball starts at zero meters.

Resources

- Programming in C, Chapter 3
- Wikipedia: Operators in C and C++
([https://en.wikipedia.org/wiki/Operators in C and C%2B%2B](https://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B))
- GNU C Tutorial, Expressions and Operators
(<http://www.crasseux.com/books/ctutorial/Expressions-and-operators.html#Expressions%20and%20operators>)



The screenshot shows a Windows desktop environment. On the left, a terminal window titled 'SimpleExpressions' displays the following commands and output:

```
laptop:~/cs211/demo/expressions>gcc -g -Wall -o literals literals.c
laptop:~/cs211/demo/expressions>./literals
x=12
x=13
laptop:~/cs211/demo/expressions>_
```

On the right, a text editor window titled 'literals.c' shows the source code:

```
1 #include <stdio.h>
2
3 int main() {
4     int x;
5
6     x=12;
7     printf("x=%d\n",x);
8     x=13;
9     printf("x=%d\n",x);
10
11 #ifdef more
12
13     float y;
14     y=7.3;
15     printf("y=%f\n",y);
16     y=9.2;
```

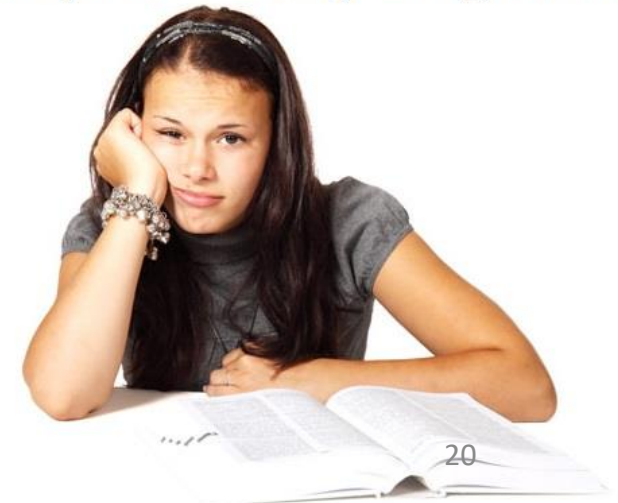
Simple Expressions

Summary Notes

Assignment Statement

Changing a Variable Value

Someone please do my assignment!



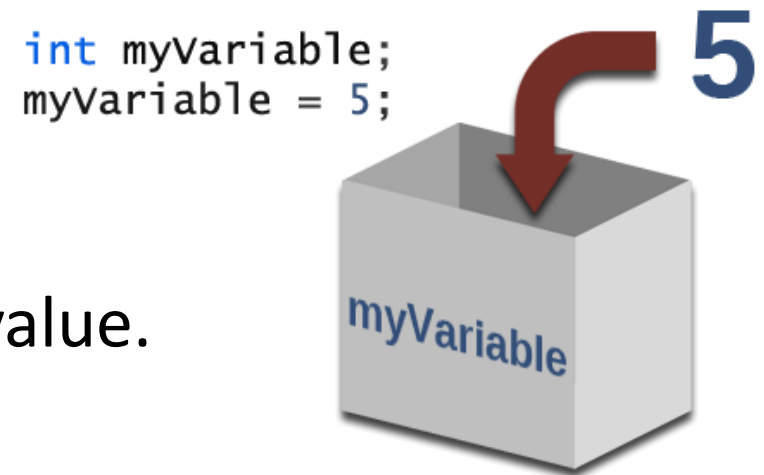
Anatomy of an Assignment Statement

$$lhs = rhs;$$

lhs : Left-Hand-Side – reference to memory, e.g. variable name
(we will learn other ways to reference memory)

rhs : C Expression

Expression is evaluated to a value, and the memory (variable) at *lhs* is updated with that value.



Literal Expressions

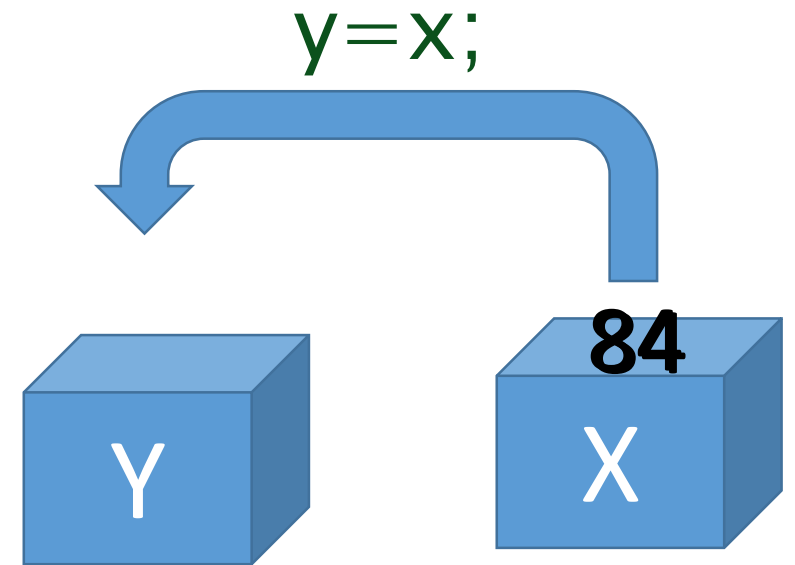
```
int x;  
float y;  
char first_init;
```

```
x=13;  
y=7.3;  
first_init='T';
```

Variable Expressions

```
int x=84; int y=17;  
float fx=7.3; float fy=9.2;  
char first_init='W';
```

```
y=x;  
fy=fx;  
first_init=y;
```



What is y? fy? first_init?

Unary Operator Expressions

```
int x=5; int y;
```

```
y=-x;
```

```
x=+30;
```



What is x? y?

Unary expressions with side effects

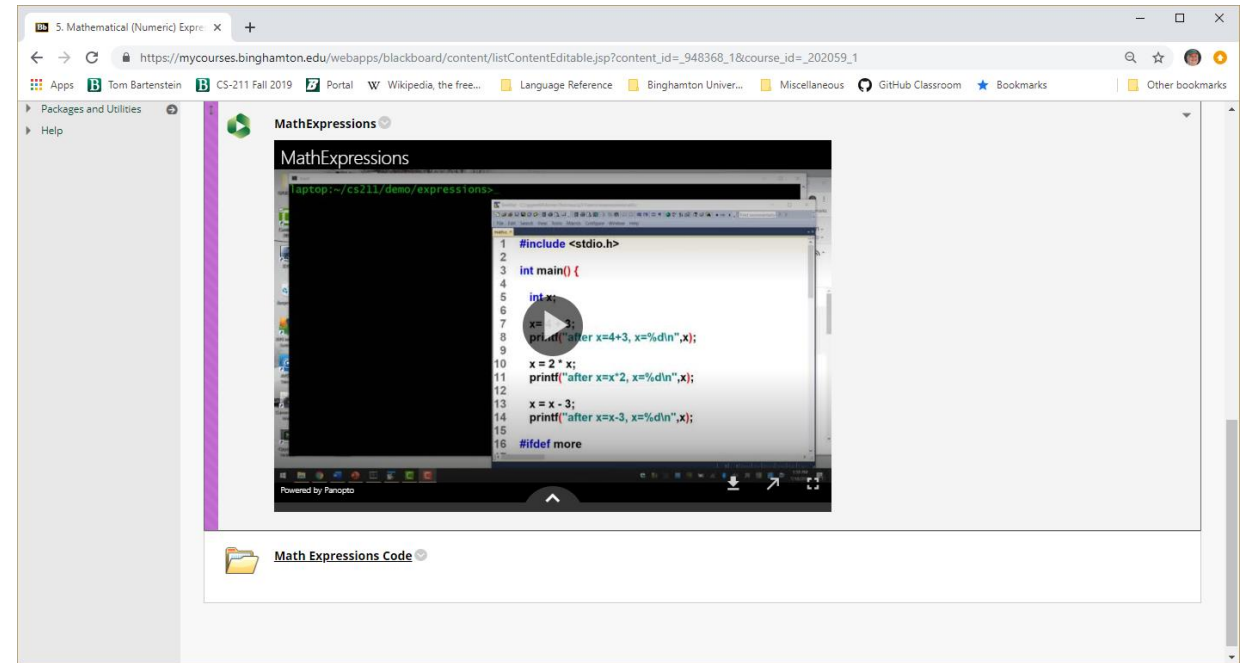
Expression	Interpretation	Example
<code>++X</code>	<code>x=x+1</code> , value is <code>x+1</code>	<code>x=12;</code> <code>y=++x;</code> // x is 13, y is 13
<code>X++</code>	value is <code>x</code> , <code>x=x+1</code>	<code>x=12;</code> <code>y=x++;</code> // x is 13, y is 12
<code>--X</code>	<code>x=x-1</code> , value is <code>x-1</code>	<code>x=12;</code> <code>y=--x;</code> // x is 11, y is 11
<code>X--</code>	value is <code>x</code> , <code>x=x-1</code>	<code>x=12;</code> <code>y=x--;</code> // x is 11, y is 12

Pitfall : Don't mix assign and increment

- `x=x++;` is redundant and ambiguous!
 - Which assignment happens first – `x=` or the side effect of `x++`?
 - On some machines, if `x` starts out at 12, it ends up 13
 - On other machines, if `x` starts out at 12, it ends up as 12!
- Typically, we increment or decrement without an assignment:
 - `++x;`
- Sometimes, we get more complex:
 - `array[i++]=next();`
 - But even this can cause a compiler warning or bug

Math Expressions

Summary Notes



Binary Math Expressions: +, -, *, /, %

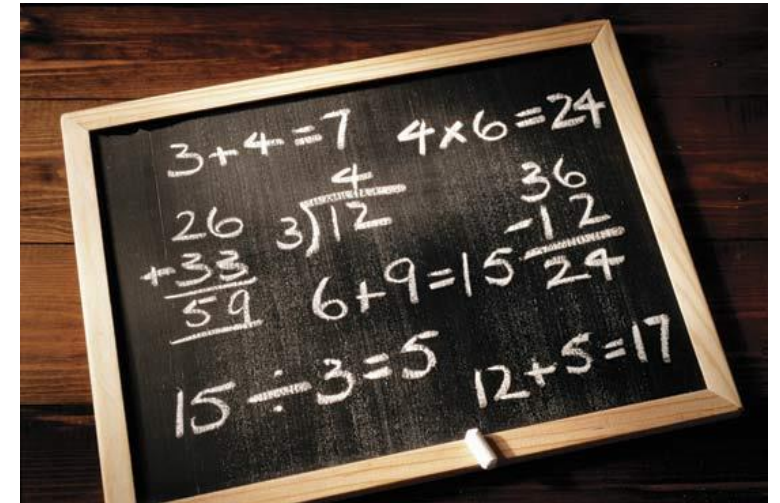
```
int x,d,r; float fx;
```

```
x = 13 * 3; /* x=39 */
```

```
d = x / 4; /* d = 9 */
```

```
r = x % 4; /* r = 3 */
```

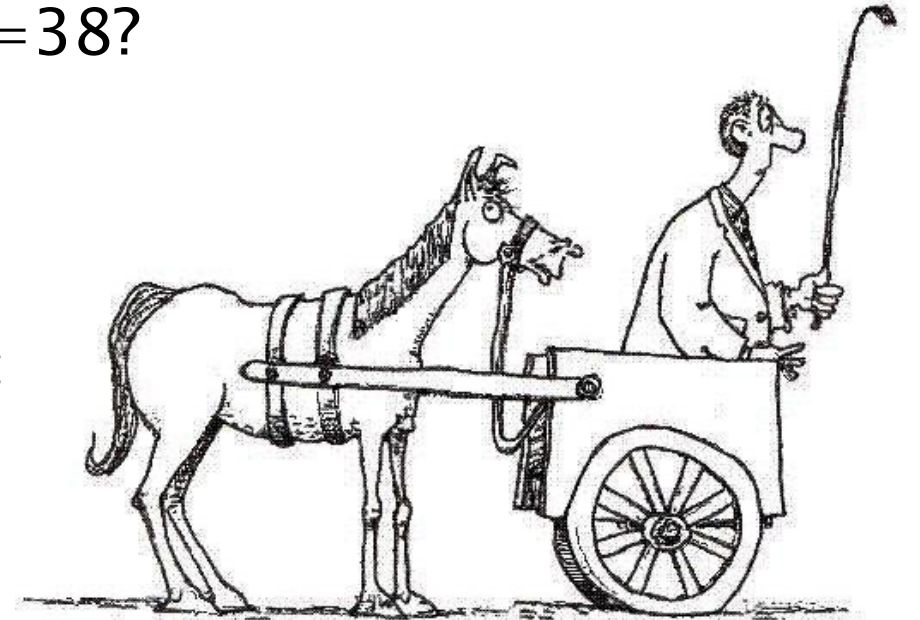
```
fx = x / 4.0; /* fx = 9.750 */
```



What is x? y? fx?

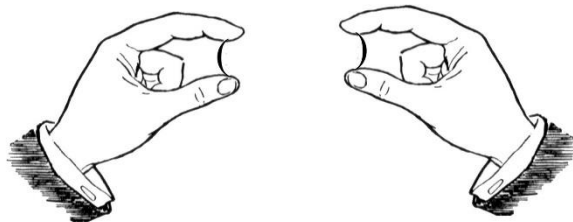
Operator Precedence

- Binary expressions are in the form *expr op expr*
- So, for instance, $3 + 5 * 7$ is a valid expression
 - Should this be evaluated as $3 + 5 = 8$, $8 * 7 = 56$?
 - Should this be evaluated as $5 * 7 = 35$, $3 + 35 = 38$?
- Rules in C: Operator Precedence
 - Always do multiplication/division/modulo first
 - Then do addition/subtraction



Parenthesis in Expressions

- Evaluate sub-expression in parenthesis first
 - e.g. $(3+5)*7$ is evaluated $3+5=8$, $8*7=56$
- Parenthesis can be nested
 - e.g. $((3+5)*(2+2))$ is evaluated $3+5=8$, $2+2=4$, $8*4=32$
- If you're not sure, use parenthesis
 - Extra parenthesis don't change the answer, $3+(5*7) = 36$
 - Missing parenthesis may result in the “wrong” answer, $3+5*7 = 36$



Assignment Expressions

- The value of an assignment is the value of the Left Hand Side



```
int x; int y; int z;
```

```
x=y=z=3;
```

What is x? y? z?

```
x=1+(y=1+(z=2));
```

What is x? y? z?

Assignment Operators

- Assignment of the form: $LHS \text{ op} = RHS$;
 - LHS : Memory Reference (variable) as in assignment
 - op : Binary operator such as $+$, $-$, $/$, $*$, $\%$, ...
 - RHS : Expression
- Shorthand for $LHS = LHS \text{ op } RHS$;

```
int x=6;
```

```
x +=2; /*x=8*/
```

```
x /=3; /*x=2*/
```

```
x*=5; /*x=10*/
```

```
int x=6;
```

```
x = x+2; /*x=8*/
```

```
x = x / 3; /*x=2*/
```

```
x = x*5; /*x=10*/
```

What is x?