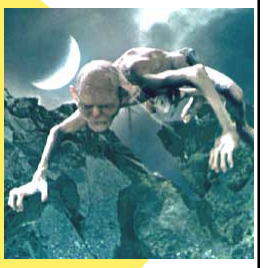


Lecture 3: Arithmetic and Casting

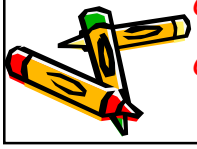
PIC 10A

Todd Wittman

Sec 2.4: Constants

- Sometimes we want a variable that does not vary.
- Declare it with the reserved word const.
- The value of a constant cannot change. You'll get an error if you try.
- Generally use capital letters for constants.
- Declare at the very top with the variables.

```
const double PI = 3.14159;
const int NUMBER_OF_HOBBITS = 4;
```



Sec 2.5: Basic Arithmetic

Symbol	Operation	Example Code
+	Addition	x = 2 + 3;
-	Subtraction	x = -2 – 5;
*	Multiplication	x = 3* -2;
/	Division	x = 3 / 2;
%	Mod	x = 9 % 2;
++	Add one	x++; or ++x;

Assignment

- In C++, the = sign means assignment not an equation.
- x = y + z;

  Add the values in y and z, then store the result in x.
- Sometimes we put a variable on both sides of the = sign.
- x = x + 2;

  Take the old value of x and add 2, then this becomes the new value of x.

Example: Circles

- Compute the area and circumference of a circle, given the radius.


```
const double PI = 3.14159;
double radius;
double area;
double circumference;
cout << "Enter the radius of the circle: ";
cin >> radius;
area = PI*radius*radius;
circumference = 2*PI*radius;
cout << "The area is " << area
     << ".\nThe circumference is " << circumference
     << ".\n\n";
```

Order of Operations

- * and / get preference over + and –
- To change the order use parantheses.

```
int x = 3 + 2*4;
```
Sets x = 3+8 = 11

```
int y = (3 + 2)*4;
```
Sets y = 5*4 = 20

- Balance your parentheses.
- You can insert white space wherever you want to make it easier to read.

```
int z = ( (3+2)*4 – (23-3)/5 ) * (1+1);
```
Sets z = (5*4 – 20/5)*2=(16)*2=32

Division & Data Types

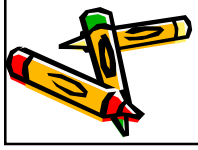
- If at least one number has a decimal, the division gives us decimals.

double x = 5/3.0; Sets x = 1.6667;

- Remember the .0 for doubles!

- For integers, the division cuts off the decimal part. Doesn't round!

int x = 5/3; Sets x = 1;



Grade school math: 5/3 = 1 R 2

So what happens to the remainder?

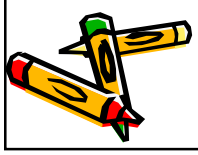
The mod operator %

- The mod operator % gives the remainder after integer division.
- 5 divided by 3 is 1 Remainder 2.

  int x = 5%3; Sets x = 2;

- Why a percent % sign? Because it kind of looks like division /.
- See example on p. 57.
- Given currency value in integer cents,

```
int dollars = value / 100;
int cents = value % 100;
```



For example, 312 cents = 3 dollars and 12 cents.

Shorthand

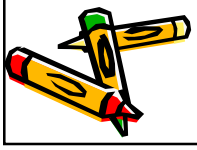
- Often we operate on the variable itself and there's a shorthand for it.

`x += 3;` is the same as `x = x+3;`

`x *= 5;` is the same as `x = x*5;`

```
int x = 20;
x /= 5;
cout << x;
```

Outputs 4



Funny looking, but it saves a little bit of typing for lazy lazy programmers.

The "add one" operator ++

- Adding one to a number is a very common operation, so there's a shorthand for that too.

`x++;` is the same as `x = x + 1;`

- We'll see this ++ again when we talk about for loops in a couple weeks.
- Subtle and complicated: There's a difference between x++ and ++x.

```
int x = 2;
int y = x++;
```

Sets y = 2.
Now x = 3.

```
int x = 2;
int y = ++x;
```

Sets y = 3.
Now x = 3.

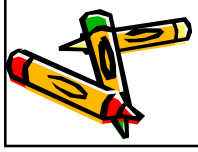
This is tricky. Just avoid using ++ in your calculations.

The cmath library

- For more complicated mathematical functions, look in the cmath library. # include <cmath>

Full list on p. 59

Function	Meaning	Equation
sqrt(x)	Square Root	$\sqrt{x}$
pow(x,y)	Power	x^y
exp(x)	Exponential	e^x
log(x)	Natural log	ln x
sin(x)	Sine	sin x
fabs(x)	Absolute value	|x|



```
int x = sqrt( pow(2,4) );
```

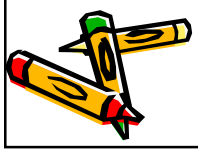
Sets $x = \sqrt{2^4} = \sqrt{16} = 4$

Data Types

- We have to watch the data types used in a calculation, especially with division.
- Our compiler will not give an error for mixing types. But we could lose information.

```
cout << 2/3;      //Shows 0
cout << 2.0/3;    //Shows 0.6667
cout << 2/3.0;    //Shows 0.6667
```

- Gets a little harder with variables. Watch the type when you combine different types.

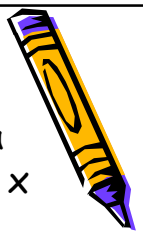


```
double x;  int y;
```

Different types!

Use a little trick called casting.

Casting



- We can temporarily change the type of a variable x in a calculation with (newtype) x

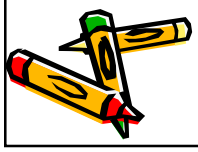
```
int x = 2;      int y = 10;
double d = 3.9;
int a = x / (int) d;
```

Sets a = 2/3 = 0

```
double b = (double) x / y;
```

Sets b = 2.0/10 = 0.2

What would the result be without casting?



- Doesn't actually change the variable, so we still have integer x=2 and double d=3.9. Just temporary adjustment for the calculation.
- You can also use the static_cast<newtype> format, but this doesn't work on all compilers (see p. 50)

Example: Time

- We want important fixed quantities, like those used for conversions, to be constants.
- Avoid "magic numbers" in your code.

```
const int SECONDS_PER_HOUR = 3600;
```

- Given an integer seconds, determine how many hours have passed *(want a whole number)*.

```
int hours = seconds / SECONDS_PER_HOUR;
```

- Given an integer seconds, convert to the number of hours *(with the decimal)*.

```
double hours = (double) seconds /
                        SECONDS_PER_HOUR;
```

Rounding & Not Rounding

- How do we round off a double x to the nearest integer?

```
double x = 3.6;
int y = (int) (x+0.5);
```

- Suppose we have integers x, y. How do we calculate x/y with the decimal?

```
int x = 5;   int y = 2;
double z = (double) x / y;
OR   double z = x / (double) y;
```

- If you have an operation with an int & a double, the result is a double.

Example: Dollars

- Given a double amount of money (e.g. $13.42), break it down into the appropriate amount of change.

```
double amount;
int dollars, quarters, dimes, nickels, pennies;
cin >> amount;
dollars = (int) amount;    //e.g. $13
quarters = (int)(100*(amount-dollars)) / 25;
```

- The last line first figures out the change (42 cents) and asks how many times 25 goes into 42.
  - How would you get the dimes?