Lecture 2: Variables and I/O

PIC 10A

Todd Wittman

Typing Up Your Program

- Thursday in the PIC Lab, I hope you noticed a couple things about typing programs.

1. C++ let's you put spaces and returns wherever you want.

```
cout<<2<<3;          cout     <<     2
                                     << 3;
```

2. If you want a space or return to appear in the output, you have to put it there with “ “ or “\n”.

```
cout << 2 << 3;          cout << 2 << “ “ << 3;
```

3. The last line return 0; wasn't necessary. But it's good style to have it.

```
int main( )     {
        cout << “hello”;
}
```

Sec 2.1: Variables & Number Types

- A variable is a storage space for a “value”, which could be a number, string (word), or even a list of numbers.
- Today we’ll look at number data types.
- The code

```
x = 2;
cout << “The value of x is “ << x;
```

will output: `The value of x is 2`

- The first line assigns the value 2 to x.
- Referring to x will look up the value 2.

Numbers and Memory

- Everything is 1’s and 0’s to a computer.
- Numbers are represented in binary as strings of 1’s and 0’s, called bits.
  - 2 = 00000010
  - 131 = 01000011
- We can also represent decimals.
- The “longer” the number, the more bits it takes to represent.
- So number 0.0002 takes more bits to store than the number 2.
  - 8 bits = 1 byte 1,000 bytes = 1 kilobyte
  - 1,000,000 bytes = 1 megabyte

Data Types

- To save memory we want to use as few bits as possible to represent our number.

2 Most Common

Type	Description	#Bits
bool	Boolean (true/false)	1
char	Character (letter, e.g. 'T')	8
short	short integer	16
int	integer (e.g. 2, 0, -42)	32
float	floating point (e.g. 3.14)	32
double	long float	64
long double	very long float	80

Declaring Variables

- Before we can use a variable, we need to declare it. Choose a name and a type.
- This action allocates memory for the variable, based on the type we assigned.

```
int x;                double y;
x = -100;             y = 42.378;
```

- Initialization: We can assign the variable a value in the declaration.

```
int x = -100;         double y = 42.378;
```

Number Types

- Be careful what values you write into a variable, depending on its type.

```
int x = 10.8;
double y = 42;
```

Will not give an error, but not good form. The integer x will be stored as x=10. Does not round!

- Integers can have no decimal (fractional) part.
- If you give an int a number with decimals, the decimal will be chopped off, not rounded off!
- Doubles should have a decimal part.

```
int x = 10;
double y = 42.0;
```

BETTER!

Rules for Choosing Variable Names

- You can choose any name you want, as long as it's not a reserved word, like cout or return.
- Must start with a letter, but could contain numbers and underscore _.

  x3 , way2cool , revenge_of_the_nerds_part_3
- No spaces, but we can change capitalization or use the underscore:

  myVariable , my_variable
- Case-sensitive. 3 different variables:

  percent vs. PERCENT vs. Percent

Guidelines for Choosing Variable Names

- Usually start with lower-case letter. Generally capitalize classes and constants. (e.g. PI)
- Better to use descriptive variables names.

  numberOfOrcs vs. n
- Rule of Thumb: Except for counters, all variable names should be at least 5 characters long.

How much is 8 pennies, 4 dimes, and 3 quarters?

```
int main() {
   int x = 8;
   int y = 4;
   int z = 3;

   double q = 0.01*x + 0.10*y + 0.25*z;
   cout << "Total value = $" << q;
   return 0;
}
```

Hard to keep track of the variables with these names.

How much is 8 pennies, 4 dimes, and 3 quarters?

```
int main() {
   int pennies = 8;
   int dimes = 4;
   int quarters = 3;

   double totalValue = 0.01*pennies +
        0.10*dimes + 0.25*quarters;
   cout << “Total value = $”
         << totalValue;
   return 0;
}
```

Much easier to follow with these names.
But it takes a little more typing.

By the way, it’s OK to continue a statement to the next line.
The semi-colon ends the statement, not the carriage return.
Best to indent if you do so.

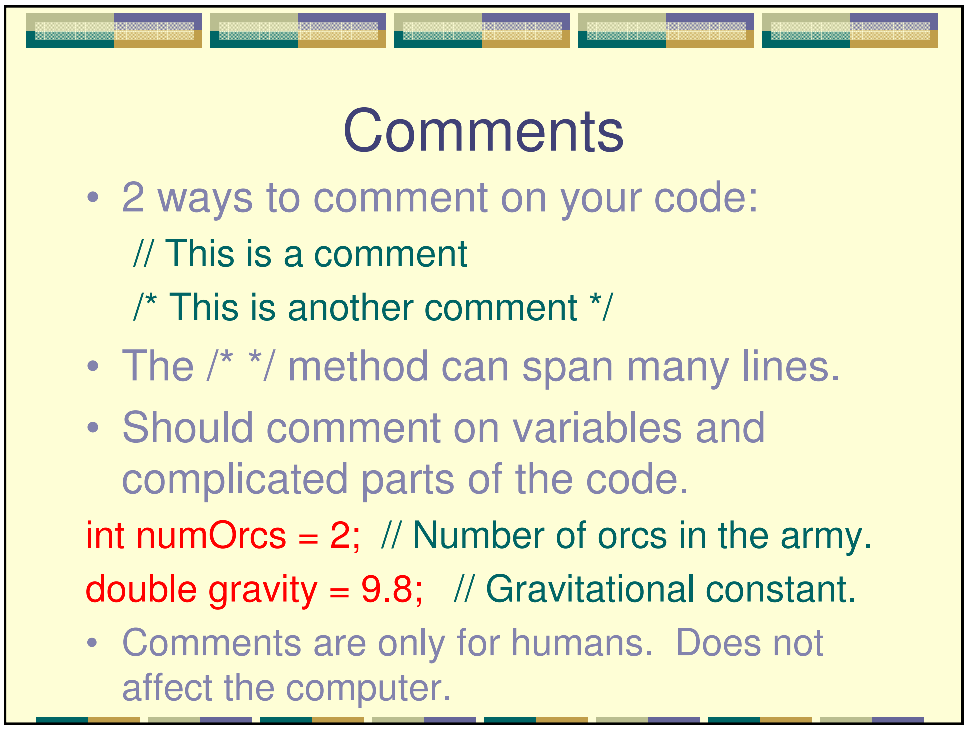
Declare Variables Right Away

- Standard to declare all variables first.

```
int main () {
    int numOrcs ;
    double gravity = 9.8;

    **STATEMENTS**
    return 0;
}
```

- As you need variables in your program, go back to the beginning and declare them.



Comments

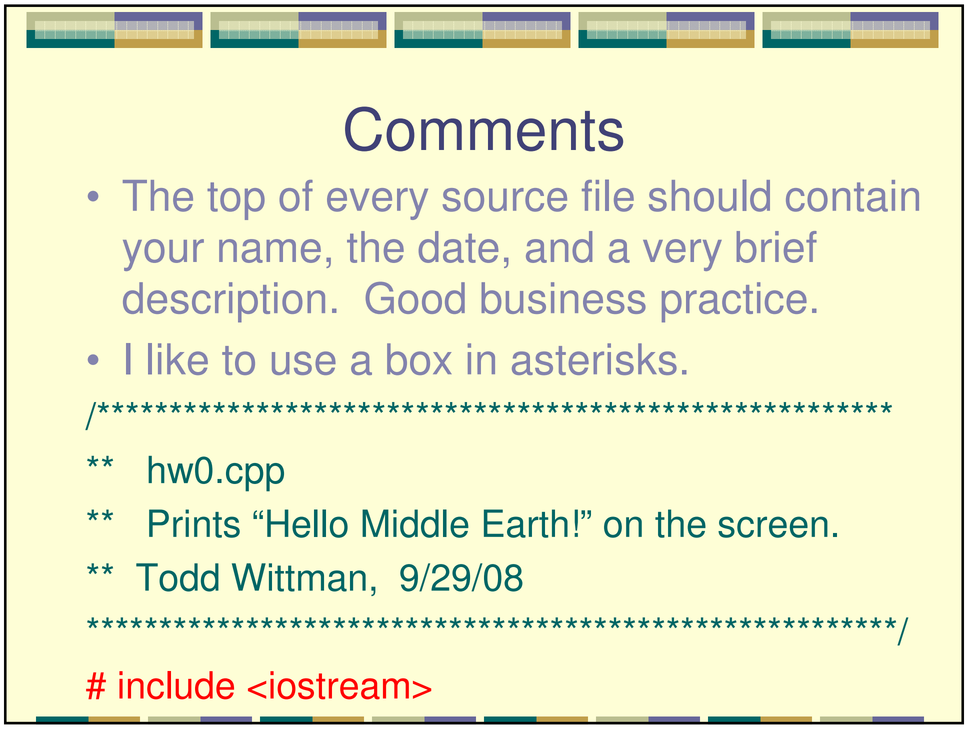
- 2 ways to comment on your code:

```
// This is a comment
/* This is another comment */
```

- The /* */ method can span many lines.
- Should comment on variables and complicated parts of the code.

```
int numOrcs = 2;  // Number of orcs in the army.
double gravity = 9.8;   // Gravitational constant.
```

- Comments are only for humans. Does not affect the computer.



Comments

- The top of every source file should contain your name, the date, and a very brief description. Good business practice.
- I like to use a box in asterisks.

```
/*****************************************************
**   hw0.cpp
**   Prints “Hello Middle Earth!” on the screen.
**  Todd Wittman,  9/29/08
******************************************************/
# include <iostream>
```

Sec 2.2: Input / Output (I/0)

- The <iostream> library includes the functions:
  - cout : Print out to the console.
  - cin : Grab data from the console.

```
int x = 2;
int y = 3;
cout << “x and y equals “ << x + y << “.\n”;
```

Print to screen

Push what follows to the screen.

Text in quotes.

Variables/ arithmetic

Carriage return

Escape Characters

- Special characters need a backslash.
- Should appear in quotes. “\n”

\n	New line
\\	Backslash \
\”	Double quote “
\t	Horizontal tab
\v	Vertical tab
\a	Alert beep (sound)

```
cout << “hi\n\n\t\a\athere”;
```

SHOWS:

```
hi


        there
```

AND BEEPS TWICE

Getting Input

- cin works the same way, with >> pulling data from the screen.
- Can input multiple values, separated by a space or line break (return).
- Ex Add two integers, separated by a space.

```
int x, y;
cout << “Gimme two numbers: “;
cin >> x >> y;
cout << “Their sum is “ << x + y << “.\n”;
```

You can create several variables of the same type, separated by a comma.

```
Gimme two numbers: 2 3
Their sum is 5.
```

Multiple values should be separated by a space.

Mind the Types

- Be careful you’re grabbing the right data type.
- If you’re expecting decimals, use a double.
- You probably won’t get an error for the wrong number type, but it will mess up your inputs.

```
int x;
cin >> x;
```

User inputs 3.14

Computer sets x=3.

Assumes 0.14 is for next input.

Buffered Input

- The input is buffered, meaning it's stored in memory until it's needed.

```
cout << “Enter first #: “;
cin >> x;
cout << “Enter second #: ”;
cin >> y;
cout << “x = “ << x << “, y = “ << y;
```

```
Enter first #: 2 8
Enter second #:
x = 2, y = 8
```

Immediately assigns y=8. Doesn't give user a chance on second line.