

CSCI 1380

JANUARY 25, 2016

A BRIEF REFRESHER...

C++ OPERATORS

To tell the computer to do something, we can use something called an **operator**.

There are a number of operators, with each specifying a particular operation...

Addition operator (+): *add two things.*

Subtraction operator (-): *subtract one thing from another.*

Multiplication operator (*): *multiply two things.*

...

THE PRINT STATEMENT

A handy operator to know: <<

This is the **insertion operator**. This instructs the computer to print something, somewhere. For example:

```
cout << "Hello!";
```

EXPRESSIONS

Alternatively, instead of "cout << 5;", we could write:

```
cout << 2 + 3
```

COMBINING OPERATIONS

Given the following statement, what will the computer do?

```
cout << 1 + 2 + 3 - 4
```

VARIABLES

Before a program stores a piece of data, it must allocate space in memory. This involves reserving space at some location in memory and giving that location a *name*.

A **variable** is just that – a named, reserved address in memory.

VARIABLE DECLARATIONS

To reserve space in memory for a piece of data, we make a **variable declaration** statement.

Variable declarations are comprised of a **type** and a name (**identifier**).

For example:

```
int x
```

DATA TYPES

In C++, every variable must have a type (it tells the computer how much space to reserve and how to process the data).

Some C++ data types:

`int`: integer (e.g., 1)

`float`: real number (e.g., 1.0)

`char`: character (e.g., `h`)

`string`: string of characters (e.g., "hello")

IDENTIFIERS (VARIABLE NAMES)

There are some rules about naming our variables...Identifiers can consist of letters, digits, and the underscore character (`_`). However, they must begin with a letter or underscore.

For example:

`first`, `First`, `_1st`, and `f_1_r_s_t` are all valid identifiers

THE READ STATEMENT

Another handy operator to know: >>

This is the **extraction operator**. This instructs the computer to extract some data from somewhere and to save it in memory. For example:

```
string s;  
cin >> s;
```

A SIMPLE PROGRAM

```
string s;
```

```
cout << "Hello! What is your name?";
```

```
cin >> s;
```

```
cout << "Nice to meet you, " << s;
```

VARIABLES (CONT.)

ASSIGNMENT

Another handy operator: =

This is the **assignment operator**. This tells the computer to store some data somewhere. For example:

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

ASSIGNMENT (SYNTAX INTERLUDE)

Another handy operator: =

This is the **assignment operator**. This tells the computer to store some data somewhere. For example:

```
int x = 5
```

OPERATORS (TERMINOLOGY INTERLUDE)

Another handy operator `=`

This is the **assignment operator**. This tells the computer to store some data somewhere. For example:

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

left hand side (LHS) `x` `=` `5` right hand side (RHS)
arguments

ASSIGNMENT (CONT.)

An **operand** can be any expression that evaluates to the right type of data. For example, the RHS of an assignment can be an expression:

```
int x;  
x = 5 + 10;
```

So what gets saved in memory at x's location? 15

A SIMPLE PROGRAM

```
int x;
```

```
cout << "Hello! Please enter a number";
```

```
cin >> x;
```

```
cout << "The value of x is: " << x;
```

ANOTHER SIMPLE PROGRAM

```
int x;
```

```
x = 5;
```

```
cout << "The value of x is: " << x;
```

USING STORED DATA

Note: **variables are reusable**. Each assignment stores a new value and overwrites any prior data. For example:

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

```
x = 1;  
cout << x;
```

```
x = x + 10;  
cout << x;
```

EXERCISE

With a partner, determine what gets printed to the console when the following code is run.

```
int x;  
int y;  
cout << "Hello";  
x = 1 + 4 - 3;  
cout << x;  
y = x - 1;  
y = y + 5;  
cout << y;
```

EXERCISE (FORMATTING INTERLUDE)

With a partner, determine what gets printed to the console when the following code is run.

```
int x;  
int y;  
cout << "Hello\n";  
x = 1 + 4 - 3;  
cout << x << endl;  
y = x - 1;  
y = y + 5;  
cout << y;
```

EXERCISE

With a partner, write C++ statements to do the following and determine what happens when they are run.

Declare an integer variable called `banana`.

Declare an integer variable called `hamster`.

Assign `banana` the value `29`.

Assign `hamster` the value of: `17` added to the value of `banana`.

Print out the value of `hamster`.

On a new line, print out the value of: `banana - hamster`.