Loops

Sections 5.1, 5.6

Increment and Decrement Operators (5.1)

- C++ provides a shortcut to increment or decrement a variable by 1
  - ++ is the unary increment operator
  - -- is the unary decrement operator

```cpp
int x = 99, y = 90;
x++; // this is equivalent to x += 1
x--; // this is equivalent to x -= 1
```

In a Loop

- Often, this is used to increment a loop counter

```cpp
int x = 1;
while(x < 5)
{
    cout << " x : " << x << endl;
    x++;
}
```

Examples

- This can be used in an expression:

```
y = x++ + 9;
```

This is equivalent to:

```
y = x + 9;
x += 1;
```

- This can also be used in a conditional

```
( x-- > 9 )
```

is equivalent to:

```
( x > 9); x -= 1;
```

Q.1. Practice

- Write one statement of code to do each of the following:

```
int x = 0, y = 1;
```

- Add x + 9 to y and increment x by 1

- Add x * 4 to y and increment x by 1

- Add y – 13 to x and decrement y by 1

Prefix vs Postfix

- ++x is *prefix*
  - The x += 1 happens **before** the expression is evaluated
- x++ is *postfix*
  - the x += 1 happens **after** the expression is evaluated

```
int y = 0, x = 0;
x = y++ + 1; // x = y + 1; y += 1;
y = ++x + 1; // x += 1; y = x + 1;
```

Q.2. What is the Output?

```cpp
int x = 0, y = 0;

x = y++ * 2;
y = ++x / 2;

cout << "x: " << x << "y: "
     << y << endl;

x = x++ + 1;
x = ++x + 1;

cout << "x: " << x << "y: "
     << y << endl;
```

Q.2. Continued

```cpp
y = (y+ x++) * 2;
x = y++ + ++x;

cout << "x: " << x << "y: "
     << y << endl;
```

Q.3. Practice

What is the output if `i = 2`?

```cpp
cout << "Value of x is" << i;

cout << "Value of i++ is" << i++;

cout << "Value of ++i is" << ++i;

cout << "Value of --i is" << --i;

cout << "Value of i-- is" << i--;
```

Q.4. Practice

- Write a single C++ statement to do each of the following:
```
int y = 0, x = 0, z = 0;
```
  - Decrement x by 1 then add 2x to y
  - Add 2y to x then increment y by 1
  - Increment x and y each by 1 then add x+y to z

for loops (5.6)

- 3 main steps for loops:
  - Initialize, Test, Update
- **for** loops provide a concise way to do this

```
//  initialize    test       update
for (count = 0; count < 5; count++)
{
  cout << count << endl;
}
```

for vs While

- This for loop
```
for(count = 0; count < 5; count++)
{
   cout << count << endl;
}
```
- is equivalent to
```
count = 0;
while(count < 5)
{
  cout << count << endl;
  count ++;        // update happens at the end
}
```

Q.5. Example

- Write a **for** loop that outputs odd numbers less than 10

Q.6. Practice

- What does this output?

```
for(i = 5; i < 10; i += 2)
{
   cout << i;
}
```

- Rewrite the **for** loop as a **while** loop

Q.7. Problem

- Write the code that will print the sum of the odd integers between 1 and 50 inclusive.

- Do this once with a **while** loop, and again using a **for** loop

Q.8. Practice

- Write a program that computes the factorial of a number. The factorial of a number is given by the formula

- The user will input N

  o N! = N*(N-1)*...*2*1

    ▪ where 0! = 1, 1! = 1, 2! = 2, 3! = 6, ...

Localized Declarations

```
for (int i = 0; i < n; i++)
{
   cout << i << endl;
}
cout << i << endl; // This will cause an error
```

- `i` is declared ONLY in the loop
- Convert this to a `while` loop

Q.9. Potential Pitfalls

- What is the output of the following loop

```
for(count = 0; count < 5; count++)
{
   cout << count << endl;
   count++;
}
```

Q.10. Practice

- What is the output of the following loop

```
for(count = 0; count < 10; count += 2)
{
  cout << count << endl;
}
```

Q.11. Problem

- Write a program that allows the user to enter 20 integers, you should then print out the following:
  - ○ The sum of all integers inputted
  - ○ The average of all integers inputted
  - ○ The largest integer of all integers inputted