Increment and Decrement

- Loops commonly have a counter variable
- Inside the loop body, counter variable is often
  - incremented: increased by one OR
  - decremented: decreased by one
- Example from last time:

```cpp
int number = 1;
while (number <= 3)
{
    cout << "Student" << number << endl;
    number = number + 1;
}
cout << "Done" << endl;
```
Increment/Decrement Operators

- C++ provides unary operators to increment and decrement.
  - Increment operator: `++`
  - Decrement operator: `--`

- Examples:

  ```
  int num = 10;
  num++;  // equivalent to: num = num + 1;
  num--;  // equivalent to: num = num - 1;
  ```

Postfix and Prefix

- The increment and decrement operators may be used in either postfix OR prefix mode:
  - Postfix: `num++`
  - Prefix: `++num`

- Examples:

  ```
  int num = 10;
  num++;  // equivalent to: num = num + 1;
  num--;  // equivalent to: num = num - 1;
  ++num;  // equivalent to: num = num + 1;
  --num;  // equivalent to: num = num - 1;
  ```
Postfix and Prefix: why?

- No difference between postfix and prefix UNLESS the variable is used in an expression:
  - Postfix, increments num AFTER it is used.
  - Prefix, increments num BEFORE it is used.

Examples:

```cpp
int num = 10;
cout << num++; //equivalent to: cout << num; num = num + 1;

cout << ++num; //equivalent to: num = num + 1; cout << num;
```

Watch out

- What is output in each case?

```cpp
int x = 13;
if (x++ > 13)
    cout << “x greater than 13” << endl;
cout << x << endl;

int x = 13;
if (++x > 13)
    cout << “x greater than 13” << endl;
cout << x << endl;
```

- I recommend NOT using ++ and -- in expressions.
Two kinds of loops

- Conditional loop: executes as long as a certain condition is true
  - input validation: loops as long as input is invalid

- Count-controlled loop: executes a specific number of times/iterations
  - count may be a literal, or stored in a variable.

- Count-controlled loop follows a pattern:
  - initialize counter to zero (or other start value).
  - test counter to make sure it is less than count.
  - update counter during each iteration.

for

- the for statement is used to easily implement a count-controlled loop.

  for (expr1; expr2; expr3)
  statement

- expr1 is evaluated (initialization).
- expr2 is evaluated (test)
  - If it is true, then statement is executed, then expr3 is executed (update), repeat.
  - If/when it is false, then statement is skipped, and the loop is exited.
for and while

• the for statement:

    for (expr1; expr2; expr3)  
    statement

• is equivalent to the following while statement:

    expr1;           // initialize
    while (expr2) {  // test
        statement
        expr3;        // update
    }

for example

• Example:

    int number;
    for (number = 1; number <= 3; number++)
    {
        cout << “Student” << number << endl;
    }
    cout << “Done” << endl;

• Output:

    Student1
    Student2
    Student3
    Done
Counters: Redo

- The example using while to output table of squares of ints 1 through 8:

```cpp
cout << "Number  Number Squared" << endl;
cout << "------  --------------" << endl;

int num = 1;
while (num <= 8) {
    cout << num << "           " << (num * num) << endl;
    num = num + 1; // increment the counter
}
```

- Rewritten using for:

```cpp
cout << "Number  Number Squared" << endl;
cout << "------  --------------" << endl;

int num;
for (num = 1; num <= 8; num++)
    cout << num << "           " << (num * num) << endl;
```

Watch out

- What is output?

```cpp
int x;

for (x=1; x <= 10; x++) {
    cout << "Repeat!" << endl;
    x++;
}

cout << "Done!" << endl;
```

- Do not update the loop variable in the body of a for loop.
Options

• What is output?

```cpp
int x;
for (x = 10; x > 0; x = x-2)
    cout << x << endl;
```

Note: no semicolon

• Can define the loop variable inside the for:

```cpp
for (int x = 10; x > 0; x=x-2)
    cout << x << endl;
```

```cpp
//ERROR, can’t use x here
```

• Do NOT try to access x outside the loop (the scope of x is the for loop only)

Non-deterministic count

• How many rows are output?

```cpp
int maxCount;
cout << “How many squares do you want?” << endl;
cin >> maxCount;

cout << “Number  Number Squared” << endl;
cout << “------  --------------” << endl;

int num;
for (num = 1; num <= maxCount; num++)
    cout << num << “ ” << (num * num) << endl;
```

• It depends . . .
  • It’s still a count controlled loop, even though the count is not known until run-time.
The exprs are optional

- You may omit any of the three exprs in the for loop header

```cpp
int value, incr;
cout << "Enter the starting value: ";
cin >> value;

for ( ; ; )
{
    cout << "Please enter the increment amount: ";
cin >> incr;
    value = value + incr;
cout << value << endl;
}
```

- Watchout:

```cpp
for ( ; ; )
    cout << "Hello!" << endl;
```