Count-controlled Loops – the for Loop and Increment Operators

Larry Caretto
Computer Science 106

Computing in Engineering and Science

March 9, 2006

Outline

- Review last class
  - While and do while loops give test-before and test-after implementation of loops
  - Getting the right count and avoiding off-by-one errors
- Count controlled loops and the for statement
- Operators like += and ++ in for loops
- Nested for loops

Review Basic Loop Structures

```
while ( <condition> )
{
    // Repeated statements
}
do
{
    // Repeated statements
}
while ( <condition> );
```

Review Tracing Loops

```
count = 0
while (count < 3 ) {
    inFile << x << y;
    cout << x + y << endl;
    count = count + 1
}
```

- What is printed to screen for file data as follows: 1 2 3 4 5 6 7 8 9 10 11 12

3
7
11

Review Correct Counts

- Watch out for off-by-one errors caused by bad initial or final count values or by incorrect condition (< *vs.* <=)
- Look at loop counter settings

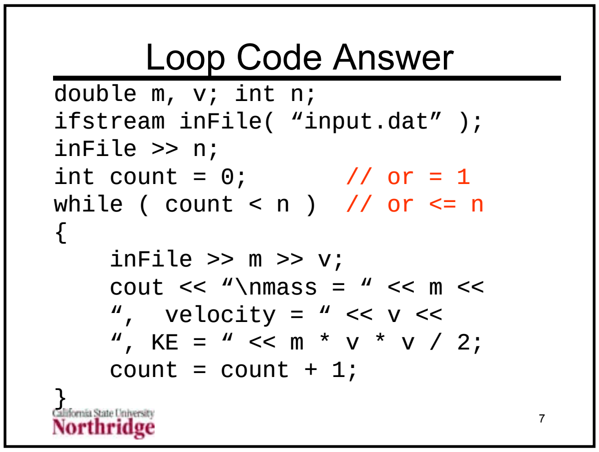
```
count = 0; // count = 1;
do {
    // calculations and output
    count = count + 1
}while ( count < 4 ); // count <= 4
```

- How many times do we go through loop?

Four times (count = 0, 1, 2, 3)

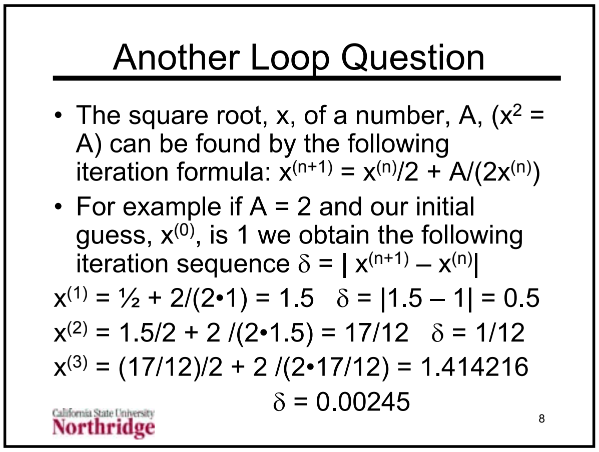
Loop Code Question

- A data file has n sets of data on mass and velocity
  - The first number on the data file is the number of data sets, n
  - This is followed by individual data sets with mass given before velocity
- Write the looping code that can read n and the data on mass and velocity data from the file and print the value of KE = $mV^2/2$ for each data set



Loop Code Answer

```
double m, v; int n;
ifstream inFile( “input.dat” );
inFile >> n;
int count = 0;       // or = 1
while ( count < n )  // or <= n
{
    inFile >> m >> v;
    cout << “\nmass = “ << m <<
    “,  velocity = “ << v <<
    “, KE = “ << m * v * v / 2;
    count = count + 1;
}
```



Another Loop Question

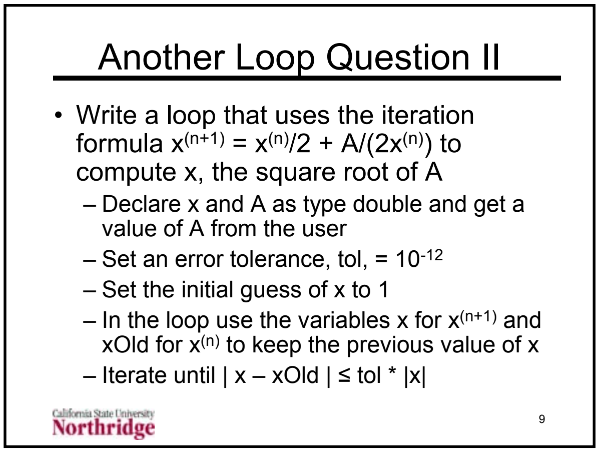
- The square root, x, of a number, A, (x^2 = A) can be found by the following iteration formula: $x^{(n+1)} = x^{(n)}/2 + A/(2x^{(n)})$
- For example if A = 2 and our initial guess, $x^{(0)}$, is 1 we obtain the following iteration sequence $\delta = | x^{(n+1)} - x^{(n)}|$

$x^{(1)} = \frac{1}{2} + 2/(2\cdot 1) = 1.5 \quad \delta = |1.5 - 1| = 0.5$

$x^{(2)} = 1.5/2 + 2/(2\cdot 1.5) = 17/12 \quad \delta = 1/12$

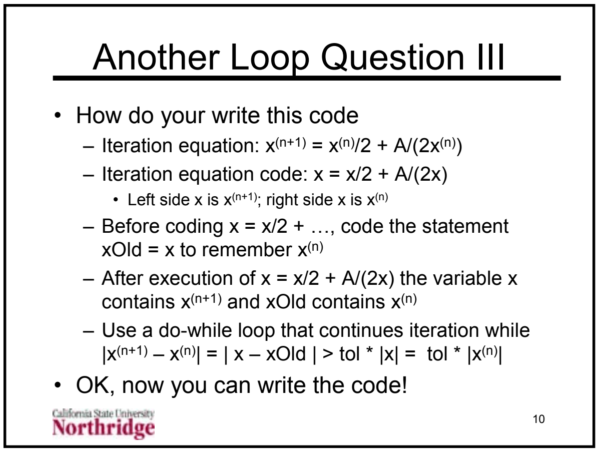
$x^{(3)} = (17/12)/2 + 2/(2\cdot 17/12) = 1.414216$

$\delta = 0.00245$



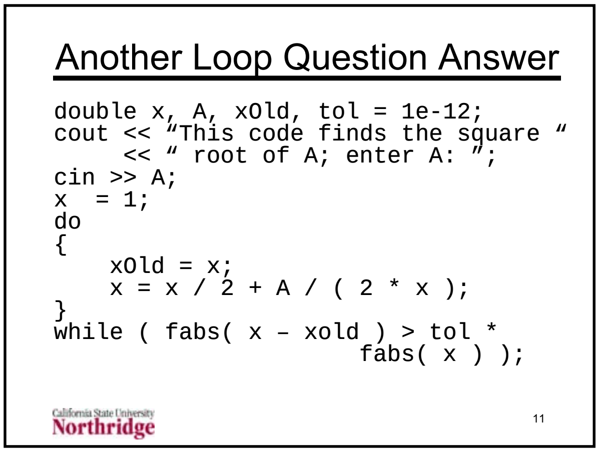
Another Loop Question II

- Write a loop that uses the iteration formula $x^{(n+1)} = x^{(n)}/2 + A/(2x^{(n)})$ to compute x, the square root of A
  - Declare x and A as type double and get a value of A from the user
  - Set an error tolerance, tol, = 10^{-12}
  - Set the initial guess of x to 1
  - In the loop use the variables x for $x^{(n+1)}$ and xOld for $x^{(n)}$ to keep the previous value of x
  - Iterate until | x – xOld | ≤ tol * |x|



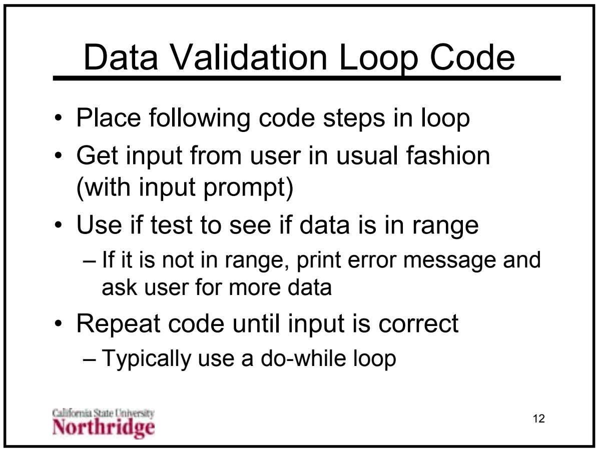
Another Loop Question III

- How do your write this code
  - Iteration equation: $x^{(n+1)} = x^{(n)}/2 + A/(2x^{(n)})$
  - Iteration equation code: x = x/2 + A/(2x)
    - Left side x is $x^{(n+1)}$; right side x is $x^{(n)}$
  - Before coding x = x/2 + …, code the statement xOld = x to remember $x^{(n)}$
  - After execution of x = x/2 + A/(2x) the variable x contains $x^{(n+1)}$ and xOld contains $x^{(n)}$
  - Use a do-while loop that continues iteration while $|x^{(n+1)} - x^{(n)}|$ = | x – xOld | > tol * |x| = tol * $|x^{(n)}|$
- OK, now you can write the code!



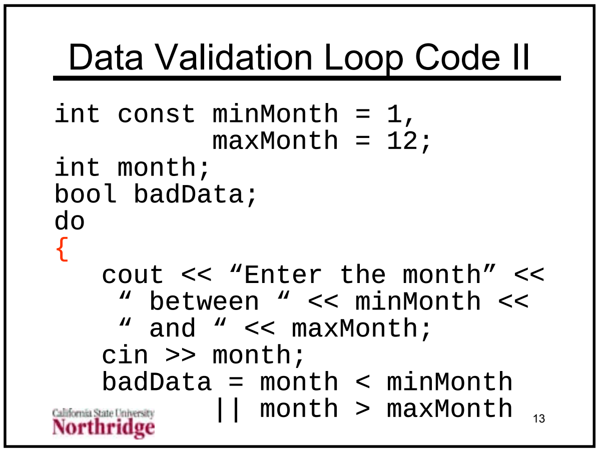
Another Loop Question Answer

```
double x, A, xOld, tol = 1e-12;
cout << “This code finds the square “
     << “ root of A; enter A: “;
cin >> A;
x  = 1;
do
{
    xOld = x;
    x = x / 2 + A / ( 2 * x );
}
while ( fabs( x – xold ) > tol *
                      fabs( x ) );
```



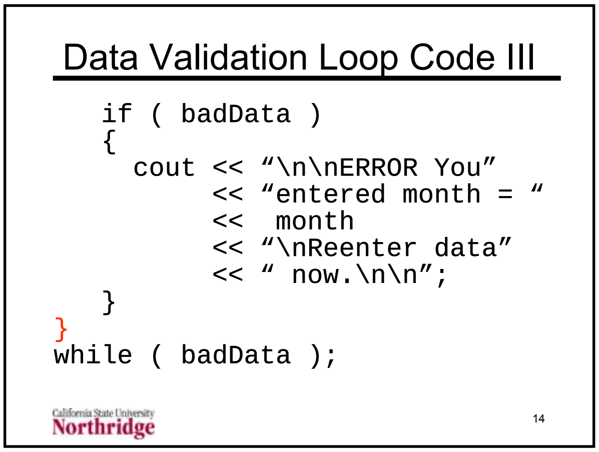
Data Validation Loop Code

- Place following code steps in loop
- Get input from user in usual fashion (with input prompt)
- Use if test to see if data is in range
  - If it is not in range, print error message and ask user for more data
- Repeat code until input is correct
  - Typically use a do-while loop



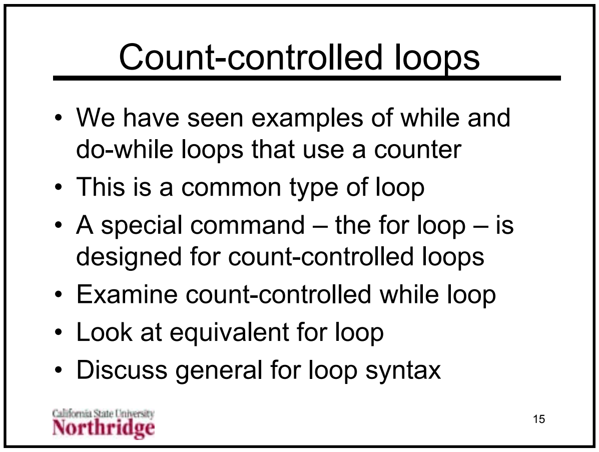
Data Validation Loop Code II

```
int const minMonth = 1,
          maxMonth = 12;
int month;
bool badData;
do
{
   cout << “Enter the month” <<
    “ between “ << minMonth <<
    “ and “ << maxMonth;
   cin >> month;
   badData = month < minMonth
          || month > maxMonth
```



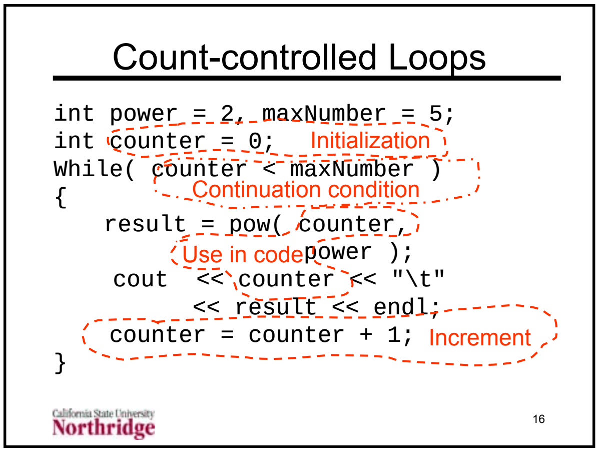
Data Validation Loop Code III

```
   if ( badData )
   {
     cout << “\n\nERROR You”
          << “entered month = “
          <<  month
          << “\nReenter data”
          << “ now.\n\n”;
   }
}
while ( badData );
```



Count-controlled loops

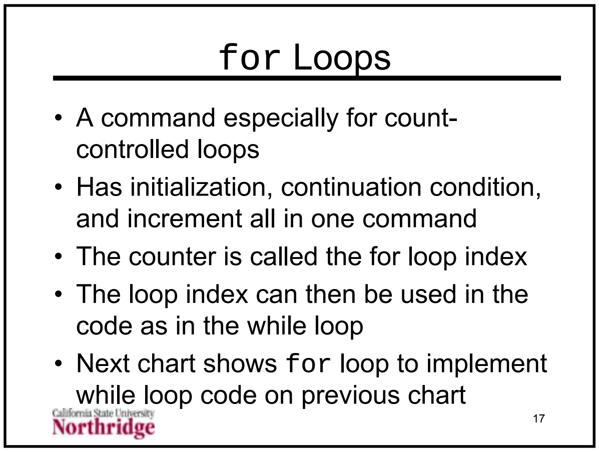
- We have seen examples of while and do-while loops that use a counter
- This is a common type of loop
- A special command – the for loop – is designed for count-controlled loops
- Examine count-controlled while loop
- Look at equivalent for loop
- Discuss general for loop syntax



Count-controlled Loops

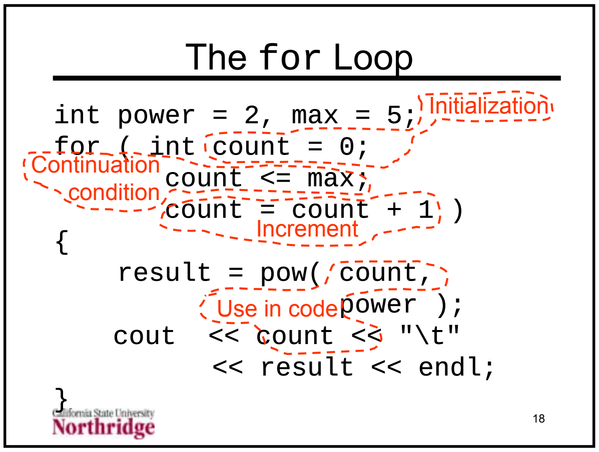
```
int power = 2, maxNumber = 5;
int counter = 0;
While( counter < maxNumber )
{
   result = pow( counter,
                 power );
   cout  << counter << "\t"
         << result << endl;
   counter = counter + 1;
}
```

Initialization
Continuation condition
Use in code
Increment



for Loops

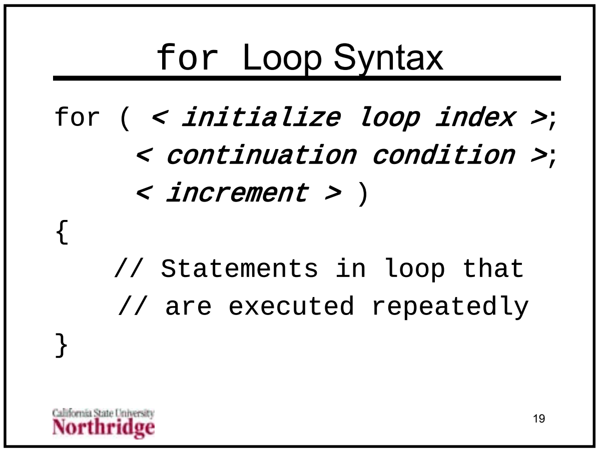
- A command especially for count-controlled loops
- Has initialization, continuation condition, and increment all in one command
- The counter is called the for loop index
- The loop index can then be used in the code as in the while loop
- Next chart shows for loop to implement while loop code on previous chart



The for Loop

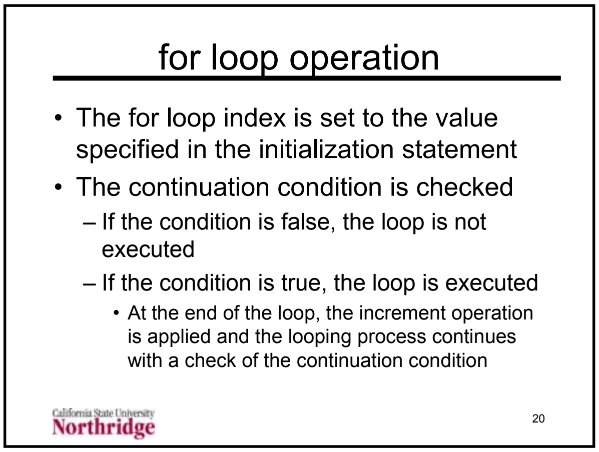
```
int power = 2, max = 5;
for ( int count = 0;
      count <= max;
      count = count + 1 )
{
   result = pow( count,
                 power );
   cout  << count << "\t"
         << result << endl;
}
```

Initialization
Continuation condition
Increment
Use in code



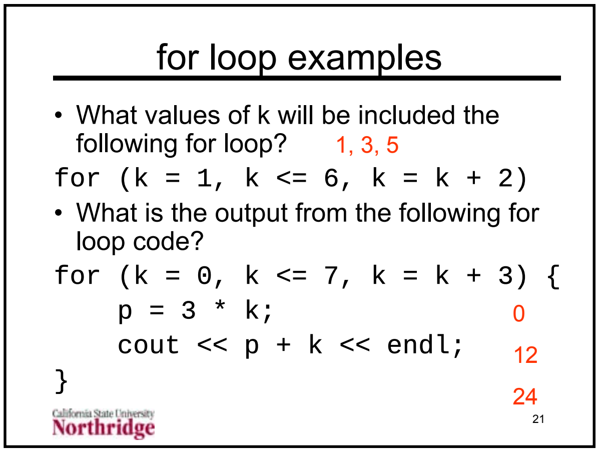
for Loop Syntax

```
for ( < initialize loop index >;
      < continuation condition >;
      < increment > )
{
    // Statements in loop that
    // are executed repeatedly
}
```



for loop operation

- The for loop index is set to the value specified in the initialization statement
- The continuation condition is checked
  - If the condition is false, the loop is not executed
  - If the condition is true, the loop is executed
    - At the end of the loop, the increment operation is applied and the looping process continues with a check of the continuation condition



for loop examples

- What values of k will be included the following for loop? 1, 3, 5

```
for (k = 1, k <= 6, k = k + 2)
```

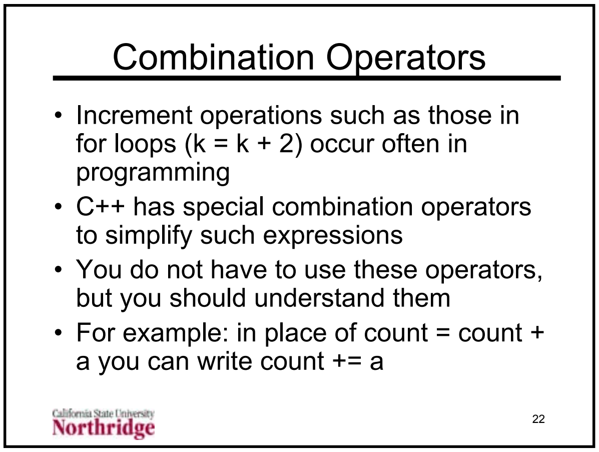
- What is the output from the following for loop code?

```
for (k = 0, k <= 7, k = k + 3) {
    p = 3 * k;
    cout << p + k << endl;
}
```

0

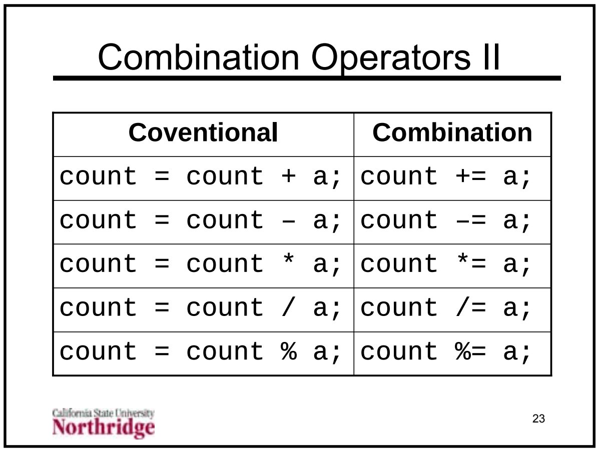
12

24



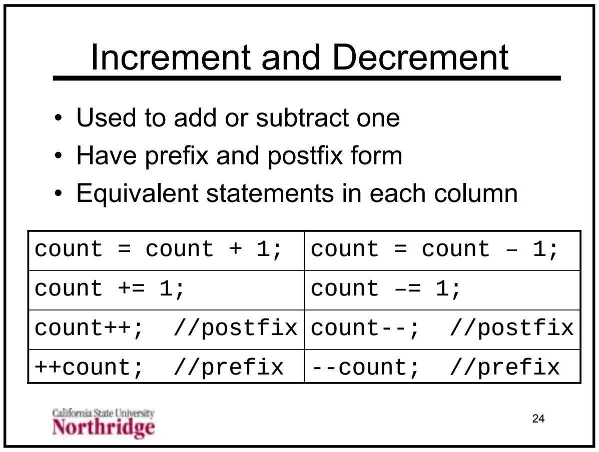
Combination Operators

- Increment operations such as those in for loops (k = k + 2) occur often in programming
- C++ has special combination operators to simplify such expressions
- You do not have to use these operators, but you should understand them
- For example: in place of count = count + a you can write count += a



Combination Operators II

Coventional	Combination
`count = count + a;`	`count += a;`
`count = count – a;`	`count –= a;`
`count = count * a;`	`count *= a;`
`count = count / a;`	`count /= a;`
`count = count % a;`	`count %= a;`



Increment and Decrement

- Used to add or subtract one
- Have prefix and postfix form
- Equivalent statements in each column

`count = count + 1;`	`count = count – 1;`
`count += 1;`	`count –= 1;`
`count++;  //postfix`	`count--;  //postfix`
`++count;  //prefix`	`--count;  //prefix`

Prefix and Postfix

- Increment and decrement can be used alone or as part of expression
- When used alone there is no difference between prefix (++j) and postifx (j++)
- When used in an expression such as k + (j++) or k + (++j)
  - Both expressions increase j by one
  - For prefix (++j) new j value is added to k
  - For postfix (j++) old j value is added to k

Prefix and Postfix II

- Equivalent code in each column below

Postfix	Prefix
`k = m + j;`	`j = j + 1;`
`j = j + 1;`	`k = m + j;`
`r = s / p;`	`p = p – 1;`
`p = p – 1;`	`r = s / p;`
`k = m + j++;`	`k = m +  ++j;`
`r = s / p--;`	`r = s / --p;`

for Loop Examples

```
for ( int i = 0; i < N; i++ )
for ( index = first; index <=
      last; index += 3 )
for ( int count = highest;
      count > 0; count-- )
for ( int current = 20;
 current <= 80; current += 10 )
```

for Loop Questions

- Write a for loop with a loop index, k, that starts at 2 and increments k by one so that the last value of k in the loop is 12

```
  for ( k = 2; k <= 12; k++ )
```

- What if we only want k = 2, 4, 6, 8, 10, 12?

```
 for ( k = 2; k <= 12; k += 2 )
```

- Write a for loop with 12 as the first value of k and decrease k by 2 with a final value of 4

```
 for ( k = 12; k >= 4; k -= 2 )
```

for Loop Exercise

- Write a for loop that computes and prints the squares of all even numbers from 2 through 12 inclusive

```
 for ( int n = 2; n <= 12; n += 2)
 {
     cout << n << “ squared is “ << n * n;
}
    // braces are optional
```