**CS 210** **Sept. 13, 2016**
**Day 6**

***More Loops***

## General Conditional Loop

1. Initialize loop control variable.
2. As long as exit condition hasn't been met
   3. Continue processing

## Loop Control Components

- initialization of the loop control variable
- test of the loop repetition condition
- change (update) of the loop control variable

- the for loop supplies a designated place for each of these three components

## The for Statement Syntax

```
for   (initialization expression;
          loop repetition condition;
          update expression)
      statement;
```

```
/* Display N asterisks. */
for  (count_star = 0;
      count_star < N;
      count_star += 1)
   printf("*");
```

Example Countable loop

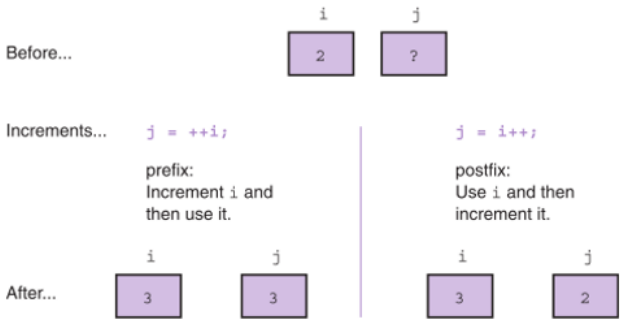
**TABLE 5.3** Compound Assignment Operators

| Statement with Simple Assignment Operator | Equivalent Statement with Compound Assignment Operator |
|---|---|
| `count_emp = count_emp + 1;` | `count_emp += 1;` |
| `time = time - 1;` | `time -= 1;` |
| `total_time = total_time + times;` | `total time += time;` |
| `product = product * item;` | `product *= item;` |
| `n = n * (x + 1);` | `n *= x + 1;` |

## Increment and Decrement Operators

- counter = counter + 1
  count += 1
  counter++

- counter = counter - 1
  count -= 1
  counter--




Example of a for loop

```c
#include<stdio.h>
int main()
  {int i, f, fn1, fn2;
   fn1 = 1;
   fn2 = 0;
   printf("Fibonacci 0 = %d\n", fn2);
   printf("Fibonacci 1 = %d\n", fn1);
   for(i=2;i<20;i++)
     {f = fn1 + fn2;
      printf("Fibonacci %d = %d\n", i, f);
      fn2 = fn1;
      fn1 = f;
     }
   return 0;
  }
```

```c
int main()
  {int i, f, fn1, fn2;
   fn1 = 1;
   fn2 = 0;
   i = 2;
   printf("Fibonacci 0 = %d\n", fn2);
   printf("Fibonacci 1 = %d\n", fn1);
   while(i < 20)
     {f = fn1 + fn2;
      printf("Fibonacci %d = %d\n", i, f);
      fn2 = fn1;
      fn1 = f;
      i++;
     }
   return 0;
  }
```

Example Factorial n! = n*(n-1)*…*1

```c
#include<stdio.h>
#define FACTMAX 20
int Factorial(int n);
int main()
{
    int i;
    for(i=0;i < FACTMAX;i++)
       printf("Factorial of %d = %d\n", i, Factorial(i));
    return 0;
}
int Factorial(int n)
{
    int i, fact;
    if(n == 0)
       return 1;
    fact = 1;
    for(i=1;i<=n;i++)
       fact *= i;
    return fact;
}
```

Example Print a table of Fahrenheit to Celsius conversion

```c
#include<stdio.h>
double ConvertToCelsius(double tF);
int main()
{
    int i;
    double tC, tF;
    tF = -40;
    printf("Fahrenheit   Celsius\n");
    for(i=0;i < 15;i++)
    {
        tC = ConvertToCelsius(tF);
        printf("%8.2f   %8.2f\n", tF, tC);
        tF = tF + 10;
    }
}
double ConvertToCelsius(double tF)
{
    return 5.0*(tF - 32)/9.0;
}
```

```
Fahrenheit   Celsius
  -40.00     -40.00
  -30.00     -34.44
  -20.00     -28.89
  -10.00     -23.33
    0.00     -17.78
   10.00     -12.22
   20.00      -6.67
   30.00      -1.11
   40.00       4.44
   50.00      10.00
   60.00      15.56
   70.00      21.11
   80.00      26.67
   90.00      32.22
  100.00      37.78
Press any key to continue . . .
```

# Conditional Loops

- used when there are programming conditions when you will not be able to determine the exact number of loop repetitions before loop execution begins

Example of a conditional loop

The trigonometric sin(x) function can be written as an infinite sum as:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

where x is a real number in radians. Write a program in C which will prompt the user for a value of x in degrees and an accuracy number and print the value of sin(x) to the required accuracy. Take $\pi = 3.141592653589793$

```c
#include<stdio.h>
#include<math.h>
#define PI 3.141592653589793
double Factorial(int n);
int main()
{
    int i;
    double degrees, x, term, acc, sinx;
    printf("This program computes the value of sin(x)\n");
    printf("Enter a value for x in degrees ... ");
    scanf_s("%lf", &degrees);
    printf("Enter a value for the accuracy ... ");
    scanf_s("%lf", &acc);
    x = degrees * PI/180;
    i = 0;
    term = pow(x, (2*i+1))/Factorial(2*i+1);
    sinx = term;
    while(term > acc)
    {
        i++;
        term = pow(x, (2*i+1))/Factorial(2*i+1);
        sinx += pow(-1, i)*term;
    }
    printf("Sin(%5.2f) = %10.8f\n", degrees, sinx);
}

double Factorial(int n)
{
   double fact;
   if(n == 0)
      return 1.0;
   fact = 1;
   while(n > 0)
   {
       fact = fact*n;
       n--;
   }
   return fact;
}
```

# Loop Design

- Sentinel-Controlled Loops
  - sentinel value: an end marker that follows the last item in a list of data
- Endfile-Controlled Loops
- Infinite Loops on Faulty Data

# Sentinel Loop Design

- Correct Sentinel Loop
  1. Initialize sum to zero.
  2. Get first score.
  3. while score is not the sentinel
     4. Add score to sum.
     5. Get next score

# Sentinel Loop Design

- Incorrect Sentinel Loop
  1. Initialize sum to zero.
  2. while score is not the sentinel
     3. Get score
     4. Add score to sum.

Example: find the average of numbers input from the keyboard.

```
#include<stdio.h>
int main()
{
      int n, sum, count;
      double avg;
      count = 0;
      sum = 0;
      printf("Enter numbers to find the average.\n");
      printf("Enter 0 to end.\n");
      printf("Enter the first number ... ");
      scanf_s("%d", &n);
      while(n != 0)
      {
          count++;
          sum += n;
          printf("Enter the next number ... ");
          scanf_s("%d", &n);
      }
      if(count > 0)
      {
            avg = (double)sum/count;
            printf("The average is %3.4f\n", avg);
      }
    return(0);
}
```

```
Enter numbers to find the average.
Enter 0 to end.
Enter the first number ... 1
Enter the next number ... 2
Enter the next number ... 3
Enter the next number ... 0
The average is 2.0000
Press any key to continue . . .
```

End Of File (EOF) loops are similar to sentinel loops except the program reads a file until it comes to an EOF marker. We will do this later.

# Nested Loops

- Loops may be nested just like other control structures
- Nested loops consist of an outer loop with one or more inner loops
- Each time the outer loop is repeated, the inner loops are reentered, their loop control expressions are reevaluated, and all required iterations are performed

Example of nested loops:
Print the sequence
000
001
010
011
100
101
110
111
Notice that the least significant digit changes on every line, the next on every other line, and the final only on every fourth line.

```
#include<stdio.h>
int main()
{
	int i, j, k;
	for(i=0;i<2;i++)
	{
		for(j=0;j<2;j++)
		{
			for(k=0;k<2;k++)
				printf("%d%d%d\n", i, j, k);
		}
	}
}
```

```
000
001
010
011
100
101
110
111
Press any key to continue . . .
```

How many line of print do each of the following produce? The variables i, j, and k are integers.

```
i = 5;
while(i>1)
 {printf("1\n";
  j = 3;
  while(j < 5)
   {printf("2\n";
    j++;
   }
  i--;
 }
```

Lines of Print _____

```
B)int i, j, k;
   i = -9;j = 1;k = 7;
   while(i < 0)
    {while(j > -2)
      {printf("%d\n", k);
       j--;
       k = k - 2;
      }
      i++;
     }
```

Lines of Print_____

**Engr 123** **September 12, 2016**
**More Loops**

The following function takes two arguments x and y and produces one returned variable z. All inputs and outputs are integers. Write a program that contains a nested loop which will use the function to calculate values of z as x goes from 0 to 5 and y goes from 0 to 3. Print each value of z on a new line.

```
int FindZ(int x, int y)
{
    if(x >= 0 && y >= 0)
       return x*x + y*y
    return 0;
}
```