

Loops

Repetition in Programs

- loop
 - a control structure that repeats a group of steps in a program
- loop body
 - the statements that are repeated in the loop

Comparison of Loop Kinds

- counting loop
 - we can determine before loop execution exactly how many loop repetitions will be needed to solve the problem
 - while, for
- sentinel-controlled loop
 - input of a list of data of any length ended by a special value
 - while, for
- endfile-controlled loop
 - input of a single list of data of any length from a data file
 - while, for
- input validation loop
 - repeated interactive input of a data value until a value within the valid range is entered
 - do-while
- general conditional loop
 - repeated processing of data until a desired condition is met
 - while, for

Counting Loops

- counter-controlled loop
 - a.k.a. counting loop
 - a loop whose required number of iterations can be determined before loop execution begins
- loop repetition condition
 - the condition that controls loop repetition
- loop control variable
 - the variable whose value controls loop repetition
- infinite loop
 - a loop that executes forever

while Statement Syntax

```
while (loop repetition condition)  
    statement;
```

```
/* display N asterisks. */  
count_star = 0  
while (count_star < N) {  
    printf("*");  
    count_star = count_star + 1;  
}
```

Computing a Sum or Product in a Loop

- accumulator
 - a variable used to store a value being computed in increments during the execution of a loop

Do Example of accumulator loop

```
#include<stdio.h>
int main()
{
    int i, sum;
    sum = 0;           //initialize accumulator value to zero
    i = 1;            //initialize loop control
    while(i < 100)
    {
        sum = sum + i;
        i = i + 2;    //update loop control
    }
    printf("sum of the odd numbers 1 to 100 is %d\n", sum);
}
```

Do Example of Fibonacci numbers

```
#include<stdio.h>
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;
}
```

Do Example to evaluate $y = 3x^3 - 12x^2 + 4x - 3$ for values of x starting at 0 and continuing in steps of 0.01 until y is greater than 1000. Print the first value of y and the corresponding value of x for which $y > 1000$.

```
#include<stdio.h>
#include<math.h>
int main()
{
    double x, y, xIncr;
    x = 0;
    xIncr = 0.01;
    y = 3*pow(x, 3) - 12*x*x +4*x - 3;
    while(y <= 1000)
    {
        x = x + xIncr;
        y = 3*pow(x, 3) - 12*x*x +4*x - 3;;
    }
    printf("x = %lf y = %lf\n", x, y);
    return 0;
}
```

Example: The value of π can be approximated from the series given by

$$\pi = 4 \times \left\{ 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - \frac{1}{15} \dots \right\}$$

```
#include<stdio.h>
//Pi = 4*{1 - 1/3 + 1/5 - 1/7 + 1/9 - ...}
// The user enters the number of terms to use.
int main()
{int terms, i;
  double pi;
  printf("Enter the number of terms to use... ");
  scanf_s("%d", &terms);
  i = 0;
  pi = 0;
  while(i < terms)
  {if(i % 2 == 0)
    pi = pi + 1.0/(2*i+1); //Add even term.
                               //Note that 2i+1 is always odd

    else
    pi = pi - 1.0/(2*i+1); //Subtract odd term
    i++;
  }
  pi = pi * 4;
  printf("For %d terms, pi = %lf\n", terms, pi);
  return 0;
}
```

Example: Write a program to find the integer square root of a number input from the user. The integer square root is the largest integer whose square is less than or equal to the number. Use a loop and do this program with an exhaustive search.

```
#include<stdio.h>
int main()
{
```

```
int i, n;
printf("Enter an integer greater than zero... ");
scanf_s("%d", &n);
i = 1;
while(i*i <= n)
{
    i = i + 1;
}
i = i - 1;
printf("%d is the integer square root of %d\n", i, n);
}
```

Engr 123

August 30, 2019

Loop equation evaluation

Write a console application that prompts the user for a value which can be used to increment the value of x . Evaluate and print the value of y where $y = x^4 - 3x^3 + 2x^2 + 1$. Use a loop to allow the value of x to go from 0 to 10 while x is incremented by the value from the user.

For example, if the user inputs a value of 1, your program should print values for y for $x = 0, 1, 2, 3, 4, \dots 10$.

Turn in a printed copy of your source file.

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
<code>count_emp = count_emp + 1;</code>	<code>count_emp += 1;</code>
<code>time = time - 1;</code>	<code>time -= 1;</code>
<code>total_time = total_time + times;</code>	<code>total_time += times;</code>
<code>product = product * item;</code>	<code>product *= item;</code>
<code>n = n * (x + 1);</code>	<code>n *= x + 1;</code>

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("*");
```