COIS1020H: Programming for Computing Systems

Chapter 5
Looping

Loops

- **Loop**
  - Structure that allows repeated execution of a block of statements
- **Loop body**
  - Block of statements within a looping structure
- C# types of loops
  - `while` loop
  - `for` loop
  - `do` loop (or `do-while` loop)

`while` Loop

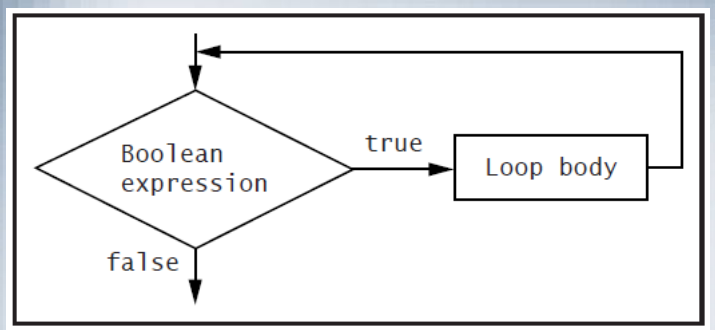


Figure 5-1 Flowchart of a loop structure

3

Using the `while` Loop

- **`while` loop**
  - Used to execute a body of statements continuously as long as some condition continues to be `true`
  - *Pretest loop*
- **Infinite loop**
  - A loop that never ends
- Making a `while` loop end correctly
  - Initialize the **loop control variable**
  - Test the control variable in the `while` expression
  - Alter the value of the control variable

4

Using the `while` Loop (cont'd.)

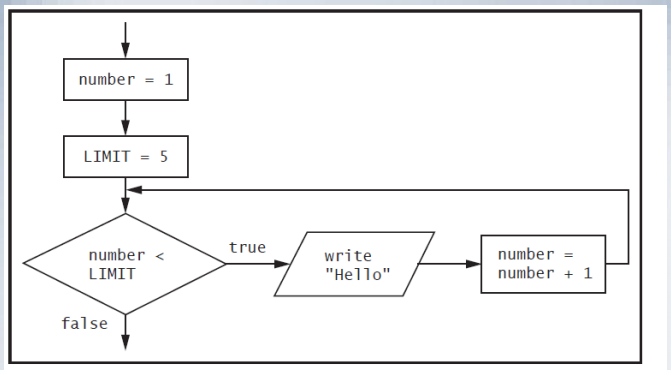


Figure 5-2 Flowchart for the logic of a `while` loop whose body executes four times

5

Using the `while` Loop (cont'd.)

```
using System;
public class FourHellos
{
   public static void Main()
   {
      int number = 1;
      const int LIMIT = 5;
      while(number < LIMIT)
      {
        Console.WriteLine("Hello");
        number = number + 1;
      }
   }
}
```

```
Hello
Hello
Hello
Hello
```

Figure 5-3 A program that contains a `while` loop whose body executes four times

6

Using the `while` Loop (cont'd.)

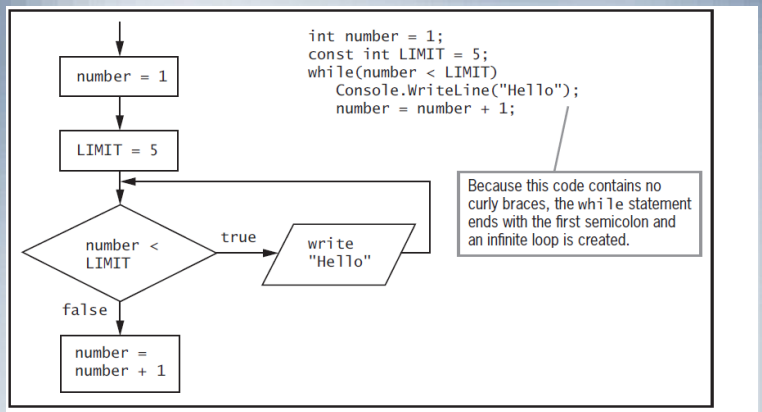


Figure 5-5 Incorrect logic when curly braces are omitted from the loop in the `FourHellos` program

7

Using the `while` Loop (cont'd.)

- **Empty body**
  - Body with no statements in it
- Alter the control variable by:
  - **Incrementing**, or adding to it
  - **Decrementing** it
- **Definite loop** or **counted loop**
  - Loop for which the number of iterations is predetermined
- **Indefinite loop**
  - Value of a loop control variable is not altered by arithmetic, but instead, is altered by user input

8

Using the `while` Loop (cont'd.)

```
using System;
public class LoopingBankBal
{
   public static void Main()
   {
      double bankBal = 1000;
      const double INT_RATE = 0.04;
      string inputString;
      char response;
      Console.Write("Do you want to see your balance? Y or N ...");
      inputString = Console.ReadLine();
      response = Convert.ToChar(inputString);
      while(response == 'Y')
      {
          Console.WriteLine("Bank balance is {0}", bankBal.ToString("C"));
          bankBal = bankBal + bankBal * INT_RATE;
          Console.Write("Do you want to see next year's balance? Y or N ...");
          inputString = Console.ReadLine();
          response = Convert.ToChar(inputString);
      }
      Console.WriteLine("Have a nice day!");
   }
}
```

```
Do you want to see your balance? Y or N …Y
Bank balance is $1,000.00
Do you want to see next year's balance? Y or N …Y
Bank balance is $1,040.00
Do you want to see next year's balance? Y or N …N
Have a nice day!
```

Figure 5-7 LoopingBankBal program

9

Using the `for` Loop

- **`for` loop**
  - Shorthand way to create definite loops
- Sections of the loop
  - Control variable initialization
  - Control variable testing
  - Control variable updating
- Other tasks
  - Initialize more than one variable
  - Declare a new variable
  - Perform more than one test

10

Using the `for` Loop (cont'd.)

- Other tasks (cont'd.)
  - Decrement or perform some other task at the end of the loop's execution
  - Perform multiple tasks at the end of the loop's execution
  - Leave one or more portions of the `for` expression empty
    - Be careful with `for(;;)`
      - It is an intentional infinite loop (POOR STYLE)
  - Also a *pretest loop*

11

Using the `for` Loop (cont'd.)

```
// Declare loop control variable and limit
int x;
const int LIMIT = 10
// Using a while loop to display 1 through 10
x = 1;
while(x <= LIMIT)
{
   Console.WriteLine(x);
   ++x;
}
// Using a for loop to display 1 through 10
for(x = 1; x <= LIMIT; ++x)
   Console.WriteLine(x);
```

Figure 5-9 Displaying integers 1 through 10 with `while` and `for` loops

12

Using the do Loop

- **do loop**
  - Checks at the bottom of the loop after one repetition has occurred (*posttest loop*)
- Convenient when you know you want to perform some task at least one time

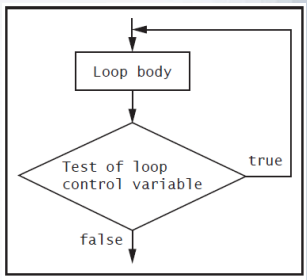
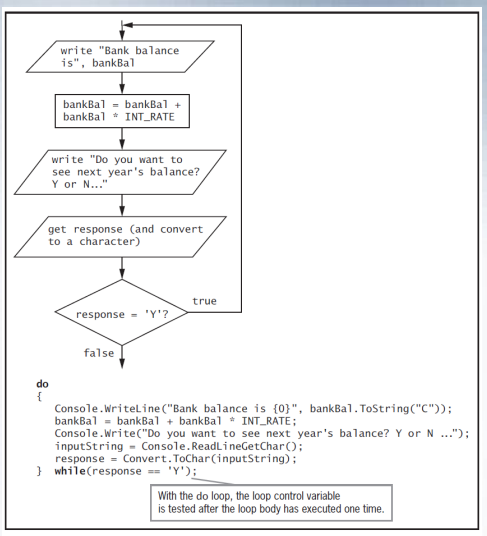


Figure 5-11 Flowchart of a do loop

13

Using the do Loop (cont'd.)




```
do
{
   Console.WriteLine("Bank balance is {0}", bankBal.ToString("C"));
   bankBal = bankBal + bankBal * INT_RATE;
   Console.Write("Do you want to see next year's balance? Y or N ...");
   inputString = Console.ReadLineGetChar();
   response = Convert.ToChar(inputString);
} while(response == 'Y');
```

With the do loop, the loop control variable is tested after the loop body has executed one time.

Figure 5-12 Part of the bank balance program using a do loop

14

Using the `do` Loop (cont'd)

- Applications of `do loop`
  - Validating data

```
Console.Write("Enter a number between 1 and 10");
num = Convert.ToInt32(Console.ReadLine());
```

becomes

```
do
{
    Console.Write("Enter a number between 1 and 10");
    num = Convert.ToInt32(Console.ReadLine());
} while (num < 1 && num > 10);
```

15

Using Nested Loops

- When loops are nested, each pair contains an **inner loop** and an **outer loop**
  - The inner loop must be entirely contained within the outer loop
  - Loops can never overlap

16

Using Nested Loops (cont'd.)

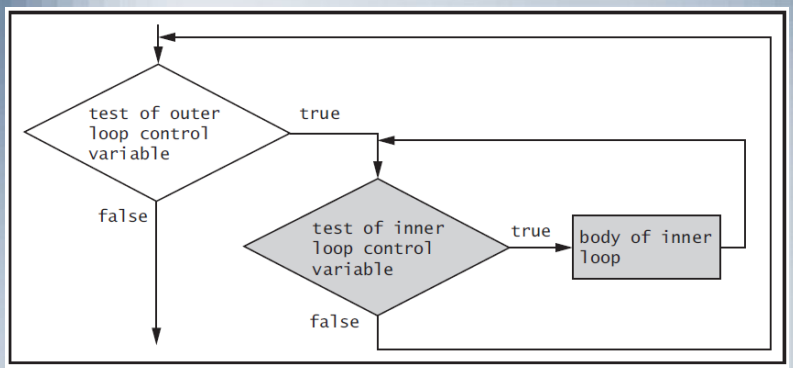


Figure 5-13 Nested loops

17

```
using System;
public class LoopingBankBal2
{
   public static void Main()
   {
      double bankBal;
      double rate;
      int year;
      const double START_BAL = 1000;
      const double START_INT = 0.04;
      const double INT_INCREASE = 0.02;
      const double LAST_INT = 0.08;
      const int END_YEAR = 5;
      for(rate = START_INT; rate <= LAST_INT; rate += INT_INCREAS
      {
         bankBal = START_BAL;
         Console.WriteLine("Starting bank balance is {0}",
            bankBal.ToString("C"));
         Console.WriteLine(" Interest Rate: {0}",
            rate.ToString("P"));
         for(year = 1; year <= END_YEAR; ++year)
         {
            bankBal = bankBal + bankBal * rate;
            Console.WriteLine(" After year {0}, bank balance is {
                year, bankBal.ToString("C"));
         }
      }
   }
}
```

```
Starting bank balance is $1,000.00
Interest Rate: 4.00%
Starting bank balance is $1,000.00
After year 1, bank balance is $1,040.00
After year 2, bank balance is $1,081.60
After year 3, bank balance is $1,124.86
After year 4, bank balance is $1,169.86
After year 5, bank balance is $1,216.65
Interest Rate: 6.00%
Starting bank balance is $1,000.00
After year 1, bank balance is $1,060.00
After year 2, bank balance is $1,123.60
After year 3, bank balance is $1,191.02
After year 4, bank balance is $1,262.48
After year 5, bank balance is $1,338.23
Interest Rate: 8.00%
Starting bank balance is $1,000.00
After year 1, bank balance is $1,080.00
After year 2, bank balance is $1,166.40
After year 3, bank balance is $1,259.71
After year 4, bank balance is $1,360.49
After year 5, bank balance is $1,469.33
```

Figure 5-14 The LoopingBankBal2 program

18

Accumulating Totals

- Totals are **accumulated**
  - Gathered together and added into a final sum
    - By processing individual records one at a time in a loop

```
sum = sum + mark;  // or
sum += mark;
```

- **Garbage**
  - Unknown value (initial value of `sum`)
  - C# compiler helps to prevent seeing an incorrect total
    - By requiring you to provide a starting value
  - C# will not use the garbage value that happens to be stored at an uninitialized memory location

19

Accumulating Totals

- Use a sentinel value to signal the end of data input
  - **A sentinel value is** a value of the same data type but not a legal data value

```
using System;
public class TotalPurchase
{
   public static void Main()
   {
      double purchase;
      double total = 0;
      string inputString;
      const double QUIT = 0;
      Console.WriteLine("Enter purchase amount ");
      inputString = Console.ReadLine();
      purchase = Convert.ToDouble(inputString);
      while(purchase != QUIT)
      {
         total += purchase;
         Console.WriteLine("Enter next purchase amount, or " +
           QUIT + " to quit ");
         inputString = Console.ReadLine();
         purchase = Convert.ToDouble(inputString);
      }
      Console.WriteLine("Your total is {0}", total.ToString("C"));
   }
}
```

```
Enter purchase amount
14.50
Enter purchase amount, or 0 to quit
5.19
Enter purchase amount, or 0 to quit
3.04
Enter purchase amount, or 0 to quit
0
Your total is $22.73
```

Figure 5-16 An application that accumulates total purchases entered by the user

20

Improving Loop Performance

- Make sure the loop does not include unnecessary operations or statements
- Example
  - A loop should execute while `x` is less than the sum of two integers, `a` and `b`
- Initial solution

```
while (x < a + b)
   // loop body
```

- Better solution

```
int sum = a + b;
while (x < sum)
   // loop body
```

21

Summary

- A loop is a structure that allows repeated execution of a block of statements
- Use a `while` loop to execute a body of statements continuously while some condition continues to be `true`
- When using a `for` statement, you can indicate in one place:
  - Starting value for the loop control variable
  - Test condition that controls loop entry
  - Expression that alters the loop control variable

22

Summary (cont'd.)

- The `do` loop checks the bottom of the loop after one repetition has occurred
- You can nest any combination of loops to achieve desired results
- In computer programs, totals frequently are accumulated
- Improve loop performance by making sure loop does not include unnecessary operations