2025-03-06

## Warm-up Exercises

1. Explain the concept of loops with sentinel, and give a small example.

Solution

Loops check for a condition to determine whether or not to repeat execution. The value that is checked in this condition is known as the sentinel value.

Example:

 int i = 0;
 while (i <= 5) { i++; } // this code will loop until it becomes greater than 5 (the sentinel value)

1. Write a program containing a **while** loop that would display the numbers between -100 and 100 (both included) with a space between them when executed.

Solution

int counter = -100;
while(counter <= 100)
 {
 Console.Write(counter++ + " ");
 }

1. Write a **for** loop that displays on the screen the sequence “1, 2, 3, 4, 5, 6, 7, 8, 9, 10,”.

Solution

for (int x = 1; x <= 10; x++)
 Console.Write(x + ", ");

1. Write a **for** loop that displays on the screen the sequence “1, 2, 3, 4, 5, 6, 7, 8, 9, 10”. Notice that the final number is not followed by a comma.

Solution

for (int x = 1; x <= 10; x++)
 Console.Write(x + ", ");
 if (x < 10) Console.Write(" ,");

1. Write a **for** loop that displays on the screen the sequence “1 3 5 7 9”.

Solution

for (int x = 1; x <= 10; x += 2)
 {
 Console.Write(x + " ");
 }

## Questions

1. A while statement can cause logic errors where the body never stops executing. This is known as a(n)
* Syntax error
* Fatal error
* Infinite loop
* None of the above.
1. A \_\_\_\_\_ can be used in a repetition structure (a loop) to control the number of times a set of statements will execute.
* Declaration
* Counter
* Controller
* None of the above.
1. How many times is the body of the loop below executed?

int counter = 10;
while (counter >= 0)
{
 counter--;
}//End while

* 9
* 10
* 11
* 0
1. How many times is the while statement checked in the code below?

int counter = 10;
while (counter >= 0)
{
 counter--;
}//End while

* 9
* 12
* 11
* 0
1. Which of the following increments the variable a by one?
* ++a
* a++
* a+=1
* All of the above.
1. Counting loops should be controlled with \_\_\_\_\_ values.
* double
* int
* char
* None of the above.
1. A common logic error known as a(n) \_\_\_\_\_ occurs when the programmer incorrectly specifies a conditional operator, such as < instead of <=.
* Fatal error
* Off-by-one error
* Syntax error
* None of the above.
1. The header for(int i = 0; i <= 10; ++i) will cause i to be incremented:
* Before the body begins execution
* After the body begins to execute, but before it finishes
* After the entire body executes
* None of the above.
1. The \_\_\_\_\_ statement, when executed in a while loop, will skip the remaining statements in the loop body and proceed with the next iteration of the loop.
* continue
* break
* next
* None of the above.
1. Consider the code segment below.

if (gender == 1)
{
 if (age >= 65)
 {
 ++seniorFemales;
 }
}

This segment is equivalent to which of the following? - [ ] if (gender == 1 || age >= 65) { ++seniorFemales; } - [x] if (gender == 1 && age >= 65) { ++seniorFemales; } - [ ] if (gender == 1 AND age >= 65) { ++seniorFemales; } - [ ] if (gender == 1 OR age >= 65) { ++seniorFemales; }

1. Methods that call themselves are known as \_\_\_\_\_ methods.
* Reiterative
* Self-calling
* Repeat-calling
* Recursive
1. What will be displayed on the screen by the following program?

for (int num = 3; num <= 5 ; num++)
 Console.Write(num + " ");

Solution

3 4 5

1. Given an **int** variable counter, write three statements to decrement its value by 1.

Solution

Four possible ways:

counter = counter - 1;
counter -= 1;
counter--;
-- counter;

1. What will be displayed on the screen?

int x = 3, y = 7;
Console.WriteLine(x++ +" and "+ --y);

Solution

“3 and 6”

1. What will be displayed on the screen by the following program?

int counter = 2;
while (counter != 5)
{
 Console.Write(counter + "\n");
 counter++;
}

Solution

2 3 4

1. What will be displayed on the screen by the following program?

int counter = 10;
while (counter != 5);
Console.Write(counter + "\n");
counter--;

Solution

Nothing, and the program will loop indefinitely.

1. What will be displayed on the screen by the following program?

int counter = 7;
while (counter != 2);
Console.Write(counter + "\n");
counter--;

Solution

7 will be displayed infinitely many times.

1. What do we name a variable that is incremented at every iteration of a loop, i.e., that keeps the running total?

Solution

An accumulator.

## Problems

1. Write an equivalent code replacing the while loop with a for loop, and provide short justification.

int A = 1;
while (A != 5)
{
 Console.WriteLine($"A= {A}");
 A = (A + 3) % 7;
}
Console.WriteLine($"A= {A}");

Solution

Example:

 for (int A = 1; A != 5; A = (A + 3) % 7)
 {
 Console.WriteLine($"A= {A}");
 }
 Console.WriteLine($"A= {A}");

The *for* loop contains the incrementing variable, the looping condition, and the incrementing statement needed for a loop.

1. Find all syntax errors in this code

using System;
namespace ConsoleApp
{
 class Program
 {
 static void Main(string[] args)
 {
 Console.Write("n= );
 int n= Int32.Parse(Console.ReadLine());
 Console.Write($"The value of odd factorial of n is equal to {OddFactorial(n)}");
 }
 static int OddFactorial(int n);
 {
 fi ((n % 2) == 0)
 return -1;
 else if (n == 1) return 1;
 else return (n \* OddFactorial(n - 2);
 }
 }
}

Solution

 using System;
 namespace ConsoleApp; // added missing semicolon
 {
 class Program
 {
 static void Main(string[] args)
 {
 Console.Write("n="); // added closing quotation marks
 int n= Int32.Parse(Console.Readline());
 Console.Write($"The value of odd factorial of n is equal to {OddFactorial(n)}");
 }
 static int OddFactorial(int n) // removed semicolon
 {
 if ((n % 2) == 0) // changed "fi" to "if"
 return -1;
 else if (n == 1) return 1;
 else return (n \* OddFactorial(n - 2)); // Added close parenthesis
 }
 }
 }

1. Write a C# program that takes a single-digit number as input and then, using a *for loop*, displays a rectangle of that digit that is 3 columns wide and 5 rows tall.

Solution

 int uInput;

 Console.Write("Please enter a single digit: ");
 uInput = int.Parse(Console.ReadLine());

 for (int i = 0; i < 15; i++)
 {
 Console.Write(uInput);

 if (i % 3 == 2) Console.WriteLine();
 }

1. Assume you are given an initialized **string** variable name, and a **string** variable field. Write a small program that assigns to field
	* “CS” if name is “Turing” or “Liskov”
	* “Math” if name is “Aryabhata” or “Noether”
	* “Unknown” otherwise.

Solution

string name;
name = "Turing"; // Value given as an example, change it to test.
string field;

switch(name)
{
 case("Turing"):
 case("Liskov"):
 field = "CS";
 break;
 case("Aryabhata"):
 case("Noether"):
 field = "Math";
 break;
 default:
 field = "Unknown";
 break;
}
Console.WriteLine(name + " worked in " + field + ".");

1. Assume you are given an un-assigned **string** variable letterGrade, and an already assigned **float** variable numberGrade. Write a small program that assigns “A” to letterGrade if numberGrade is between 100 and 90 (both included), “B” if numberGrade is between 90 (excluded) and 80 (included), etc., and “Invalid data” is strictly lower than 0 or strictly greater than 100. Should you use a **switch** statement or an **if…else if…else**?

Solution

An if…else if…else is the right structure for the task:

float numberGrade;
string letterGrade;
numberGrade = -60; // This is just an example, feel free to change it.

if(numberGrade > 100 || numberGrade < 0){
// It's actually easier to get rid of the "invalid" cases first.
 letterGrade = "Invalid Data";
}
else if (numberGrade >= 90){
 letterGrade = "A";
}
else if (numberGrade >= 80){
 letterGrade = "B";
}
else if (numberGrade >= 70){
 letterGrade = "C";
}
else if (numberGrade >= 60){
 letterGrade = "D";
}
else{
// We know the value is greater than 0 but strictly lower than 60.
 letterGrade = "F";
}

Console.WriteLine(numberGrade + " corresponds to " + letterGrade);

1. Write a loop that displays on the screen numbers between [0.0, 1.0], using one decimal place precision, i.e. 0.0, 0.1, 0.2, 0.3…

Solution

for (decimal i = 0m; i <= 1.0m; i += 0.1m)
{
 Console.WriteLine($"{i:N1}");
}

1. Write a loop that displays on the screen a value that decreases by 0.5 on each iteration. Start from 10 and iterate as long as the value remains positive.

Solution

for (decimal i = 10m; i > 0m; i -= 0.5m)
{
 Console.WriteLine($"{i:N1}");
}

1. Write a program that computes the sum of numbers [1, n]. You can choose any value you want for n, where n > 1. For example, if you choose n = 10, then program should compute and display the result for the following: 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 55.

Solution

int n = 12; // This value chosen arbitrarily for test purposes. You can use any positive integer value for n.
int sum = 0;
for (int i = 1; i <= n; i++)
{
 Console.Write(i + " ");
 sum += i;
}
Console.WriteLine("= " + sum);