VISUAL PROGRAMMING LAB 03

WORKBOOK

Instructor

Md. Rashedul Islam

SUBMIT WITHIN: 12:35 PM (TODAY, 28 September 2022)

SEND PROGRAMS TO

rashed.class.official17@gmail.com

Email Subject: LAB03_CSC440_ID_FALL2022

1. C# program to demonstrate the use of DivRem() method of Math class

Here, we will learn about the DivRem() method of Math class. This method is used to calculate the quotient of two given numbers and it also returns the remainder in the third output parameter.

Syntax:

int Math.DivRem(int dividend, int divisor, out int remainder);

Parameter(s):

dividend: The dividend.
divisor: The divisor.
remainder: The remainder.
Return value:

It returns quotient according to passed dividend and divisor.

Program:

```
namespace CLASS02
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            int REM = 0;
            int QUOTIENT = 0;

            QUOTIENT = Math.DivRem(21, 4, out REM);

            Console.WriteLine("QUOTIENT is: " + QUOTIENT);
            Console.WriteLine("REMAINDER is:" + REM);

            Console.ReadKey();
        }
    }}
```

2. C# program to demonstrate the use of BigMul() method of Math class

Here, we will learn about the BigMul() method of Math class. This method is used to return long values after the multiplication of the maximum value of an integer. Normally we multiply two integers with maximum value then it does not produce actual output. So we need to BigMul() method of Math class.

Syntax:

long Math.BigMul(int Val1, int Val2);

Parameter(s):

Val1: The first number to multiply.
Val2: The second number to multiply.
Return value:
It returns a long integer value.

Program:

```
using System;
namespace CLASS02
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            int IntVal1 = 0;
            int IntVal2 = 0;
            long BigMultResult = 0;
            long SimpleMultResult = 0;

            IntVal1 = Int32.MaxValue;
            IntVal2 = Int32.MaxValue;

            SimpleMultResult = IntVal1 * IntVal1;
            BigMultResult = Math.BigMul(IntVal1, IntVal2);

            Console.WriteLine("Result of Simple product of MaxValues: " + SimpleMultResult);
            Console.WriteLine("Result of product of MaxValues using BigMul(): " + BigMultResult);

            Console.ReadKey();
        }}}
```

Difference between Int16, Int32 and Int64 in C#

Int16: This Struct is used to represents 16-bit signed integer. The Int16 can store both types of values including negative and positive between the ranges of -32768 to +32767. It takes 2-bytes space in the memory.

Example :

```
// C# program to show the
// Int16 Struct usage

using System;
using System.Text;

public
class GFG {

 // Main Method
 static void Main(string[] args) {

   // printing minimum & maximum values
   Console.WriteLine("Minimum value of Int16: "
                + Int16.MinValue);
   Console.WriteLine("Maximum value of Int16: "
                + Int16.MaxValue);
   Console.WriteLine();

   // Int16 array
   Int16[] arr1 = {-3, 0, 1, 3, 7};

   foreach (Int16 i in arr1)
   {
     Console.WriteLine(i);
   }
  }
}
```

Int32: This Struct is used to represents 32-bit signed integer. The Int32 can store both types of values including negative and positive between the ranges of -2147483648 to +2147483647. It takes 4-bytes space in the memory.

Example :

```
// C# program to show the
// Int32 struct usage

using System;
using System.Text;

public
class GFG {
```

```
 // Main Method
 static void Main(string[] args) {

  // printing minimum & maximum values
  Console.WriteLine("Minimum value of Int32: "
            + Int32.MinValue);
  Console.WriteLine("Maximum value of Int32: "
            + Int32.MaxValue);
  Console.WriteLine();

  // Int32 array
  Int32[] arr1 = {-3, 0, 1, 3, 7};

  foreach (Int32 i in arr1)
  {
    Console.WriteLine(i);
  }
 }
}
```

Int64: This Struct is used to represents 64-bit signed integer. The Int64 can store both types of values including negative and positive between the ranges of -9,223,372,036,854,775,808 to +9, 223,372,036,854,775,807. It takes 8-bytes space in the memory.

Example :

```
// C# program to show
// Int64 struct usage
using System;
using System.Text;

public class GFG {

 // Main Method
 static void Main(string[] args) {

  // printing minimum & maximum values
  Console.WriteLine("Minimum value of Int64: "
            + Int64.MinValue);
  Console.WriteLine("Maximum value of Int64: "
            + Int64.MaxValue);
  Console.WriteLine();

  // Int64 array
  Int64[] arr1 = {-3, 0, 1, 3, 7};
```

```
foreach (Int64 i in arr1)
{
  Console.WriteLine(i);}
```

3. C# program to demonstrate the use of Sin() method of Math class

Here, we will learn about the Sin() method of Math class. This method is used to return the Sine of the specified angle.

Syntax:

double Math.Sin(double angle);

Parameter:

angle : Given angle, for this we got sine value.

Return value:

It returns sine value of specified angle.

Program:

```
1   using System;
2   namespace CLASS02
3   {
        0 references
4       class Program
5       {
            0 references
6           static void Main(string[] args)
7           {
8               double SINE = 0.0;
9
10              SINE = Math.Sin(30);
11              System.Console.WriteLine("SINE of 30 is : " + SINE);
12
13              Console.ReadKey();
14          }}}
15
```

4. C# program to demonstrate the use of Log() method of Math class

Here, we will learn about the Log() method of Math class. This method is used to get the logarithm of a specified number with a given base. This method is overload two times. If we pass a single parameter then it will return the logarithm of the given number using base e. if we pass two parameters then the first parameter is used for number and the second parameter is for the base.

Syntax:

double Math.Log(double number);
double Math.Log(double number, double base);
Parameter(s):

number: The number whose logarithm is to be found.
base: The base of the logarithm.
Return value:

It returns the logarithm of a given number.

Program:

```
1     using System;
2     namespace CLASS02
3     {
          0 references
4         class Program
5         {
              0 references
6             static void Main(string[] args)
7             {
8                 double logarithm1 = 0.0;
9                 double logarithm2 = 0.0;
10
11                //Here we get logarithm of base e for 90.
12                logarithm1 = Math.Log(90);
13
14                //Here we get logarithm of base 10 for 90.
15                logarithm2 = Math.Log(90, 10);
16
17                Console.WriteLine("Logarithm of Base E:  " + logarithm1);
18                Console.WriteLine("Logarithm of Base 10: " + logarithm2);
19
20                Console.ReadKey();
21            }}}
22
```

5. C# program to print backslash (\)

In C#, \ is a special character (sign) – that is used for escape sequences like to print a new line – we use \n, to print a tab – we use \t etc.

In this case, if we write \ within the message – it will throw an error "Unrecognized escape sequence".

To print a backslash (\), we have to use a double backslash (\\).

```
using System;
namespace CLASS02
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            // printing "\"
            Console.WriteLine("\\");

            // printing between string "\"
            Console.WriteLine("Hello\\World");

            // printing "\"n and "\"t
            Console.WriteLine("\\n\\t");

            //hit ENTER to exit the program
            Console.ReadLine();

            Console.ReadKey();
        }}}
```

6. C# program to demonstrate example of bitwise operators

Bitwise operators are used to perform calculations on the bits.

Here is the list of bitwise operators,

"&" (Bitwise AND) – returns 1 (sets bit), if both bits are set
"|" (Bitwise OR) – returns 1 (sets bit), if any or all bits are set
"^" (Bitwise XOR) – returns 1 (sets bit), if only one bit is set (not both bits are set)
"~" (Bitwise NOT) – returns one's compliment of the operand, it's a unary operator
"<<" (Bitwise Left Shift) – moves the number of bits to the left
">>" (Bitwise Right Shift) – moves the number of bits to the right

Bitwise AND

This is one of the most commonly used logical bitwise operators. It is represented by a single ampersand sign (&). Two integer expressions are written on each side of the (&) operator.

The result of the bitwise AND operation is 1 if both the bits have the value as 1; otherwise, the result is always 0.

Let us consider that we have 2 variables op1 and op2 with values as follows:

Op1 = 0000 1101
Op2 = 0001 1001

The result of the AND operation on variables op1 and op2 will be

Result = 0000 1001

As we can see, two variables are compared bit by bit. Whenever the value of a bit in both the variables is 1, then the result will be 1 or else 0.

Bitwise OR

It is represented by a single vertical bar sign (|). Two integer expressions are written on each side of the (|) operator.

The result of the bitwise OR operation is 1 if at least one of the expressions has the value as 1; otherwise, the result is always 0.

Let us consider that we have 2 variables op1 and op2 with values as follows:

Op1 = 0000 1101

Op2 = 0001 1001

The result of the OR operation on variables op1 and op2 will be

Result = 0001 1101

As we can see, two variables are compared bit by bit. Whenever the value of a bit in one of the variables is 1, then the result will be 1 or else 0.

Bitwise Exclusive OR

It is represented by a symbol (^). Two integer expressions are written on each side of the (^) operator.

The result of the bitwise Exclusive-OR operation is 1 if only one of the expressions has the value as 1; otherwise, the result is always 0.

Let us consider that we have 2 variables op1 and op2 with values as follows:

Op1 = 0000 1101
Op2 = 0001 1001

The result of the OR operation on variables op1 and op2 will be

Result = 0001 0100

As we can see, two variables are compared bit by bit. Whenever only one variable holds the value 1 then the result is 0 else 0 will be the result.

Bitwise shift operators

The bitwise shift operators are used to move/shift the bit patterns either to the left or right side. Left and right are two shift operators provided by 'C#' which are represented as follows:

Operand << n (Left Shift)
Operand >> n (Right Shift)

Here, an operand is an integer expression on which we have to perform the shift operation.
'n' is the total number of bit positions that we have to shift in the integer expression.

The left shift operation will shift the 'n' number of bits to the left side. The leftmost bits in the expression will be popped out, and n bits with the value 0 will be filled on the right side.

The right shift operation will shift the 'n' number of bits to the right side. The rightmost 'n' bits in the expression will be popped out, and the value 0 will be filled on the left side.

Example: x is an integer expression with data 1111. After performing shift operation, the result will be:

x << 2 (left shift) = 1111<<2 = 1100
x>>2 (right shift) = 1111>>2 = 0011

Bitwise complement operator (~)

The bitwise complement is also called as one's complement operator since it always takes only one value or an operand. It is a unary operator.

When we perform complement on any bits, all the 1's become 0's and vice versa.

If we have an integer expression that contains 0000 1111 then after performing bitwise complement operation the value will become 1111 0000.

```
class Program
{
    0 references
    static void Main(string[] args)
    {
        int a = 10;
        int b = 3;
        int result = 0;

        result = a & b;     //1010 & 0011 = 0010 = 2
        Console.WriteLine("a & b  : {0}", result);

        result = a | b;     //1010 | 0011 = 1011 = 11
        Console.WriteLine("a | b  : {0}", result);

        result = a ^ b;     //1010 ^ 0011 = 1001 =9
        Console.WriteLine("a ^ b  : {0}", result);

        result = ~a;        //ones compliment of 10
        Console.WriteLine("~a     : {0}", result);

        result = a << 2;    //1010<<2 = 101000 = 40
        Console.WriteLine("a << b : {0}", result);

        result = a >> 2;    //1010>>2 = 0010 = 2
        Console.WriteLine("a >> b : {0}", result);
```

7. C# program to read the grade of students and print the appropriate description of grade

Following are the grades and description,

Grade	Description
A	Excellent
B	Very Good
C	Good
D	Keep it up
E	Poor
F	Very Poor

Program:

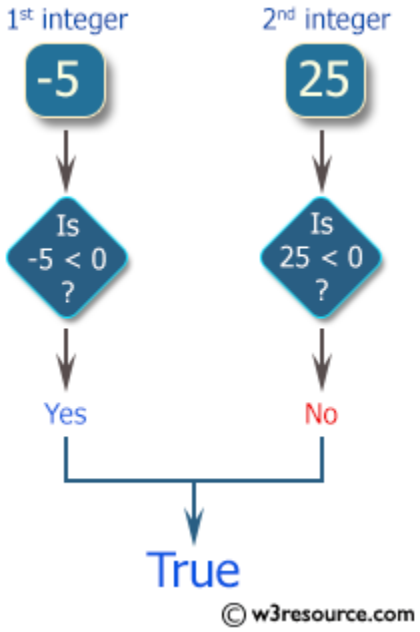
```
using System;
namespace CLASS02
{
    class Program
    {
        static void Main(string[] args)
        {
            char student_grade;

            Console.Write("Enter the student grade: ");
            student_grade = Convert.ToChar(Console.ReadLine());

            switch (student_grade)
            {
                case 'A':
                    Console.WriteLine("Excellent");
                    break;
                case 'B':
                    Console.WriteLine("Very Good");
                    break;
                case 'C':
                    Console.WriteLine("Good");
                    break;
                case 'D':
                    Console.WriteLine("Keep it up");
                    break;
                case 'E':
                    Console.WriteLine("Poor");
                    break;
                case 'F':
                    Console.WriteLine("Very Poor");
                    break;
```

```
                default:
                    Console.WriteLine("Invalid GRADE");
                    break;
            }
            Console.ReadKey();
        }}}
```

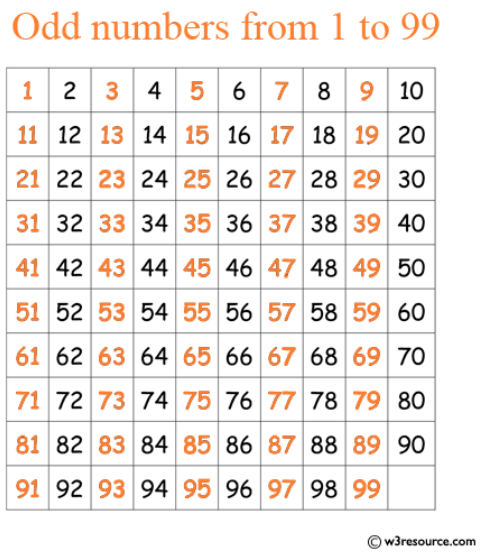
8. **Write a C# program to check two given integers and return true if one is negative and, one is positive.**




```
using System;
namespace CLASS02
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("\nInput first integer:");
            int x = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Input second integer:");
            int y = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Check if one is negative and one is
positive:");
            Console.WriteLine((x < 0 && y > 0) || (x > 0 && y < 0));
            Console.ReadKey();
```

```
        }}}
```

9. C# Sharp Exercises: Print the odd numbers from 1 to 99




```
using System;
namespace CLASS02
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Odd numbers from 1 to 99. Prints one
number per line.");
            for (int n = 1; n < (99 + 1); n++)
            {
                if (n % 2 != 0)
                {
```

```
                Console.WriteLine(n.ToString());
            }
        }
        Console.ReadKey();
    }}}
```

10.Sum of digits program in C#

Sum of digits algorithm
To get sum of each digit by C# program, use the following algorithm:

Step 1: Get number by user
Step 2: Get the modulus/remainder of the number
Step 3: sum the remainder of the number
Step 4: Divide the number by 10
Step 5: Repeat the step 2 while number is greater than 0.

```
using System;
namespace CLASS02
{
    class Program
    {
        static void Main(string[] args)
        {
            int n, sum = 0, m;
            Console.Write("Enter a number: ");
            n = int.Parse(Console.ReadLine());
            while (n > 0)
            {
                m = n % 10;
                sum = sum + m;
                n = n / 10;
            }
```

```
Console.Write("Sum is= " + sum);
Console.ReadKey();
  }}}
```