Contents

[1. C# Program to Swap two Numbers 2](#_Toc158053000)

[2. C# program to find area of rectangle 3](#_Toc158053001)

[3. C# program to find area of circle 4](#_Toc158053002)

[4. C# program to convert a Celsius temperature into Fahrenheit 5](#_Toc158053003)

[5. C# code for days into years, weeks, and days conversion 6](#_Toc158053004)

[6. C# program to find the square root of a given number without using Math.Sqrt() method. 7](#_Toc158053005)

[7. Escape Sequence Program 8](#_Toc158053006)

# 1. C# Program to Swap two Numbers

*/\**

*\* C# Program to Swap two Numbers*

*\*/*

**using** System;

**using** System.Collections.Generic;

**using** System.Linq;

**using** System.Text;

**namespace** Program

{

**class** Program

{

**static** **void** Main(**string**[] args)

{

**int** num1, num2, temp;

Console.Write("**\n**Enter the First Number : ");

num1 = **int**.Parse(Console.ReadLine());

Console.Write("**\n**Enter the Second Number : ");

num2 = **int**.Parse(Console.ReadLine());

temp = num1;

num1 = num2;

num2 = temp;

Console.Write("**\n**After Swapping : ");

Console.Write("**\n**First Number : "+num1);

Console.Write("**\n**Second Number : "+num2);

Console.Read();

}

}

}

# 2. C# program to find area of rectangle

using System;

class RectangleProgram

{

static void Main()

{

// Ask the user to enter the length

Console.Write("Enter the length of the rectangle: ");

double length = double.Parse(Console.ReadLine());

// program uses double.Parse() to convert the user's input from a string to a double

// Ask the user to enter the breadth

Console.Write("Enter the breadth of the rectangle: ");

double breadth = double.Parse(Console.ReadLine());

// Calculate the area and perimeter

double area = length \* breadth;

double perimeter = 2 \* (length + breadth);

// Display the results

Console.WriteLine($"Area of the rectangle: {area:F2}");

Console.WriteLine($"Perimeter of the rectangle: {perimeter:F2}");

}

}

# 3. C# program to find area of circle

using System;

class CircleProgram

{

static void Main()

{

// Ask the user to enter the radius

Console.Write("Enter the radius of the circle: ");

// Read the radius from the user's input

double radius = double.Parse(Console.ReadLine());

// Calculate the perimeter and area of the circle

double perimeter = 2 \* Math.PI \* radius;

double area = Math.PI \* Math.Pow(radius, 2);

// Display the results

Console.WriteLine($"Perimeter of the circle: {perimeter:F2}");

Console.WriteLine($"Area of the circle: {area:F2}");

}

}

# 4. C# program to convert a Celsius temperature into Fahrenheit

using System;

class CelsiusToFahrenheitProgram

{

static void Main()

{

// Ask the user to enter the temperature in Celsius

Console.Write("Enter the temperature in Celsius: ");

double celsius = double.Parse(Console.ReadLine());

// Convert Celsius to Fahrenheit

double fahrenheit = (celsius \* 9 / 5) + 32;

// Display the result

Console.WriteLine($"Temperature in Fahrenheit: {fahrenheit:F2}°F");

}

}

# 5. C# code for days into years, weeks, and days conversion

using System;

class DaysConversionProgram

{

static void Main()

{

// Ask the user to enter the total number of days

Console.Write("Enter the total number of days: ");

int totalDays = int.Parse(Console.ReadLine());

// Calculate years, weeks, and remaining days

int years = totalDays / 365;

int weeks = (totalDays % 365) / 7;

int remainingDays = totalDays % 365 % 7;

// Display the conversion

Console.WriteLine($"Years: {years}");

Console.WriteLine($"Weeks: {weeks}");

Console.WriteLine($"Remaining Days: {remainingDays}");

}

}

# 6. C# program to find the square root of a given number without using Math.Sqrt() method.

using System;

class SquareRootProgram

{

static void Main()

{

// Declare variables

int number = 0;

double squareRoot = 0;

// Ask the user to enter the value of the number

Console.Write("Enter the value of number: ");

number = Convert.ToInt32(Console.ReadLine());

// Calculate square root using Math.Sqrt()

//squareRoot = Math.Sqrt(number);

// Calculate square root using Math.Pow() and Math.Ceiling()

squareRoot = Math.Ceiling(Math.Pow(number, 0.5));

// Display the result

Console.WriteLine("Square Root is: " + squareRoot);

}

}

**Math.Pow() (Power Function):**

Purpose: Raises a number to the power of another number.

Usage: **Math.Pow(base, exponent**) returns base raised to the power of exponent.

In the Program: Math.Pow(number, 0.5) calculates the square root by raising number to the power of 0.5.

**Math.Ceiling() (Ceiling Function):**

Purpose: Rounds a number to the smallest integral value greater than or equal to it.

Usage: **Math.Ceiling(value)** returns the smallest integer greater than or equal to value.

In the Program: Math.Ceiling (result) is used to round up the calculated square root to the nearest integer.

# 7. Escape Sequence Program

using System;

class EscapeSequenceProgram

{

static void Main()

{

// Using escape sequences

Console.WriteLine("Escape Sequence Examples:");

Console.WriteLine("1. New Line: Hello\nWorld");

Console.WriteLine("2. Tab: Hello\tWorld");

Console.WriteLine("3. Backslash: C:\\Windows\\System32");

Console.WriteLine("4. Single Quote: She said, 'Hello'");

Console.WriteLine("5. Double Quote: He said, \"Goodbye\"");

// Using verbatim string to avoid escape sequences

Console.WriteLine("\nUsing Verbatim String:");

Console.WriteLine(@"C:\Users\John\Documents");

Console.WriteLine(@"This is a verbatim string.

It preserves new lines and escape sequences.");

// Waiting for user input before closing the console

Console.ReadLine();

}

}

Here

* \n is used for a new line.
* \t is used for a tab.
* \\ is used to display a backslash.
* \' is used to display a single quote.
* \" is used to display a double quote.

**Verbatim string**

A verbatim string in C# is a special kind of string that starts with the @ symbol. Verbatim strings are useful when you want to include backslashes, newline characters, or other escape sequences in a string without interpreting them.

Example

* string regularString = "This is a new line:\nSecond line";

-In this case, the \n is interpreted as a new line.

* string verbatimString = @"This is a new line:

Second line";

-In this case, \n is treated as the characters \ and n, not as a new line.

When to Use Verbatim Strings:

Paths: Verbatim strings are often used for file paths because they can contain backslashes without needing to double them.

Regular Expressions: When working with regular expressions, verbatim strings can make patterns more readable.

string filePath = @"C:\Users\DP\Documents";

string regexPattern = @"^\d{3}-\d{2}-\d{4}$";