**1. Input and Output**

**Purpose:** To interact with the user by receiving input and displaying output.

using System; // Namespace for basic input and output operations.

class Program

{

static void Main()

{

Console.WriteLine("Enter your name:"); // Displays a message to the user.

string name = Console.ReadLine(); // Reads a line of text from the user.

Console.WriteLine($"Hello, {name}!"); // Uses interpolation to display a personalized message.

}

}

**Explanation:**

* Console.WriteLine: Outputs text to the console.
* Console.ReadLine: Reads user input from the console.
* $"{variable}": String interpolation for formatting output.

**2. Conditional Statements**

**Purpose:** To make decisions based on conditions.

using System;

class Program

{

static void Main()

{

int age = 18; // Declares a variable.

if (age >= 18)

{

Console.WriteLine("You are an adult."); // Executes if the condition is true.

}

else

{

Console.WriteLine("You are not an adult."); // Executes if the condition is false.

}

}

}

**Explanation:**

* if: Evaluates a condition.
* else: Executes when the if condition is false.
* Relational operators (>=): Used to compare values.

**3. Loops**

**Purpose:** To execute a block of code multiple times. **For Loop Example:**

for (int i = 0; i < 5; i++)

{

Console.WriteLine($"Iteration {i + 1}");

}

* **Explanation:** The for loop initializes a counter (i = 0), checks a condition (i < 5), and increments the counter (i++) in each iteration.

**While Loop Example:**

int counter = 0;

while (counter < 5)

{

Console.WriteLine($"Count: {counter}");

counter++;

}

* **Explanation:** The while loop executes as long as the condition (counter < 5) is true.

**4. Methods**

**Purpose:** To encapsulate reusable code into functions.

using System;

class Program

{

static void Main()

{

int result = AddNumbers(5, 10); // Calls the method with arguments.

Console.WriteLine($"Sum: {result}"); // Outputs the result.

}

static int AddNumbers(int a, int b)

{

return a + b; // Returns the sum of two numbers.

}

}

**Explanation:**

* static: Makes the method accessible without creating an object.
* int AddNumbers(int a, int b): Defines a method that takes two integers and returns an integer.

**5. Arrays**

**Purpose:** To store multiple values in a single variable.

using System;

class Program

{

static void Main()

{

int[] numbers = { 1, 2, 3, 4, 5 }; // Initializes an array.

foreach (int number in numbers)

{

Console.WriteLine(number); // Iterates through the array and prints each value.

}

}

}

**Explanation:**

* Arrays are collections of elements of the same type.
* foreach: Loops through each element in a collection.

**6. Classes and Objects**

**Purpose:** To implement object-oriented programming principles.

using System;

class Person

{

public string Name { get; set; } // Auto-implemented property.

public int Age { get; set; }

public void Introduce()

{

Console.WriteLine($"Hi, I'm {Name} and I'm {Age} years old.");

}

}

class Program

{

static void Main()

{

Person person = new Person { Name = "John", Age = 30 }; // Creates an object.

person.Introduce(); // Calls the method of the object.

}

}

**Explanation:**

* class: Defines a blueprint for objects.
* public: Specifies access modifiers.
* Name { get; set; }: Defines properties with getter and setter.

**7. File Handling**

**Purpose:** To read and write files.

using System;

using System.IO;

class Program

{

static void Main()

{

string filePath = "example.txt";

// Writing to a file

File.WriteAllText(filePath, "Hello, File!");

// Reading from a file

string content = File.ReadAllText(filePath);

Console.WriteLine(content);

}

}

**Explanation:**

* File.WriteAllText: Writes text to a file.
* File.ReadAllText: Reads all text from a file.

**8. Exception Handling**

**Purpose:** To handle runtime errors gracefully.

using System;

class Program

{

static void Main()

{

try

{

int number = int.Parse("NotANumber"); // Will throw a FormatException.

}

catch (FormatException ex)

{

Console.WriteLine($"Error: {ex.Message}"); // Catches and handles the error.

}

}

}

**Explanation:**

* try: Encapsulates code that may throw exceptions.
* catch: Defines how to handle specific exceptions.

**9. Basic LINQ Query**

**Purpose:** To perform queries on collections.

using System;

using System.Linq;

class Program

{

static void Main()

{

int[] numbers = { 1, 2, 3, 4, 5 };

var evenNumbers = numbers.Where(n => n % 2 == 0); // Filters even numbers.

foreach (var num in evenNumbers)

{

Console.WriteLine(num); // Prints each even number.

}

}

}

**Explanation:**

* Where: Filters elements based on a condition.
* Lambda expressions (n => n % 2 == 0): Compact way to define conditions.