Programming Basics - Detailed Guide

1.Data Types, Variables, Constants, and Input/Output

Data types define the type of data a variable can hold. C# is strongly typed, so you must declare the type.

**Common Data Types:**

- int: Whole numbers (e.g., int age = 25;)

- float: Decimal numbers (e.g., float pi = 3.14f;)

- char: Single character (e.g., char grade = 'A';)

- string: Text (e.g., string name = "Mohammed";)

- bool: True/False (e.g., bool isOnline = true;)

Variables store data that can change.

Constants are fixed values that cannot be modified.

Example:

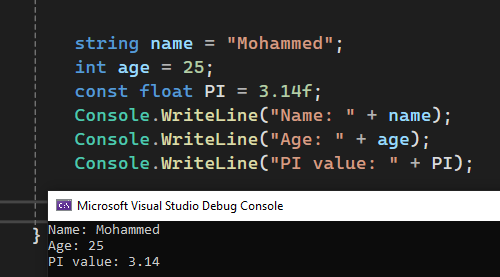
string name = "Mohammed";

int age = 25;

const float PI = 3.14f;

Console.WriteLine("Name: " + name);

Console.WriteLine("Age: " + age);

Console.WriteLine("PI value: " + PI);

Output:

Name: Mohammed

Age: 25

PI value: 3.14

To take input from the user and display output:

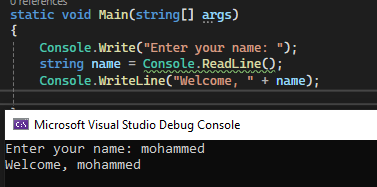
Console.Write("Enter your name: ");

string name = Console.ReadLine();

Console.WriteLine("Welcome, " + name);

Input: Mohammed

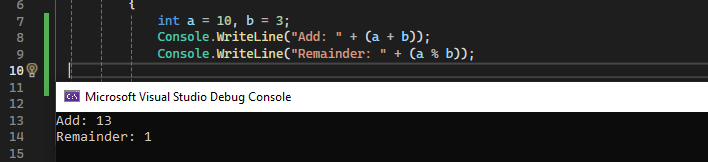
Output: Welcome, Mohammed



2. Arithmetic and Logical Operators

Arithmetic operators perform mathematical operations, while logical operators compare true/false conditions.

Arithmetic Example:

int a = 10, b = 3;

Console.WriteLine("Add: " + (a + b));

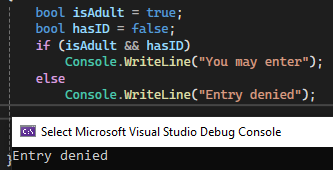
Console.WriteLine("Remainder: " + (a % b));

Output:

Add: 13

Remainder: 1

Logical Example:



bool isAdult = true;

bool hasID = false;

if (isAdult && hasID)

Console.WriteLine("You may enter");

else

Console.WriteLine("Entry denied");

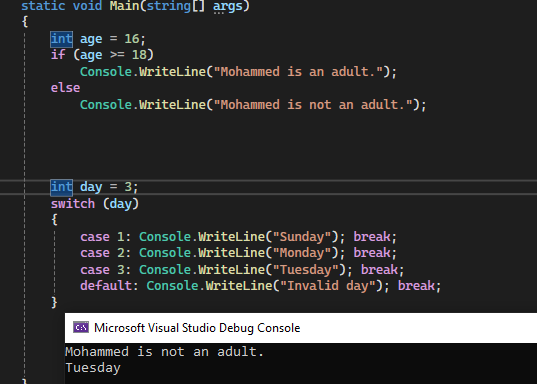
Output:

Entry denied

3. Conditional Statements

Conditional statements allow decision-making in code.

if-else Example:

int age = 16;

if (age >= 18)

Console.WriteLine("Mohammed is an adult.");

else

Console.WriteLine("Mohammed is not an adult.");

Output:

Mohammed is not an adult.

switch Example:

int day = 3;

switch (day)

{

case 1: Console.WriteLine("Sunday"); break;

case 2: Console.WriteLine("Monday"); break;

case 3: Console.WriteLine("Tuesday"); break;

default: Console.WriteLine("Invalid day"); break;

}

4. Looping (for, while, do-while)

Loops are used to repeat a block of code multiple times.

for loop Example:

for (int i = 1; i <= 3; i++)

{

Console.WriteLine("Hello Mohammed! Count: " + i);

}

Output:

Hello Mohammed! Count: 1

Hello Mohammed! Count: 2

Hello Mohammed! Count: 3

while loop Example:

int i = 1;

while (i <= 3)

{

Console.WriteLine("Step " + i);

i++;

}

Output:

Step 1

Step 2

Step 3

do-while loop Example:

int i = 5;

do

{

Console.WriteLine("This runs at least once");

i++;

} while (i < 5);

Output:

This runs at least once

14. Nested Operations

Nested operations mean placing one control structure (like if or loop) inside another.

Nested if Example:

int score = 90;

if (score > 60)

{

if (score > 80)

Console.WriteLine("Mohammed did excellent!");

}

Output:

Mohammed did excellent!

Nested loop Example:

for (int i = 1; i <= 2; i++)

{

for (int j = 1; j <= 2; j++)

Console.WriteLine($"i = {i}, j = {j}");

}

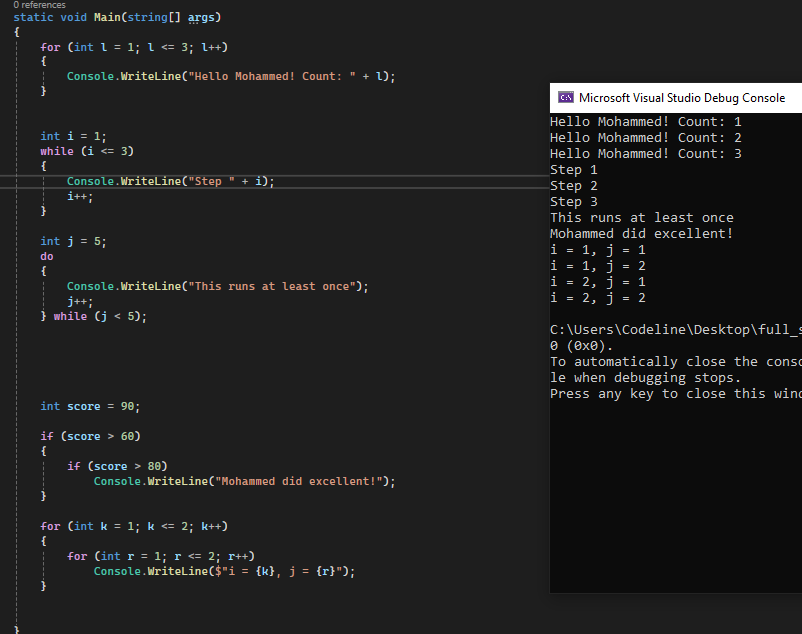
Output:

i = 1, j = 1

i = 1, j = 2

i = 2, j = 1

i = 2, j = 2



5. Array Data Structure

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

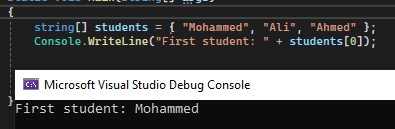
Array Example:

string[] students = { "Mohammed", "Ali", "Sara" };

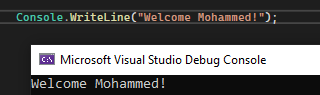
Console.WriteLine("First student: " + students[0]);

Output:

First student: Mohammed



6. Functions in C# (Built-in and User-defined)

Functions are blocks of code that perform a specific task. You can use built-in functions or create your own.

**Built-in function Example:**

It is download and add to the compiler file sdk

So you can use it any time each language has it ons

built -in functions.

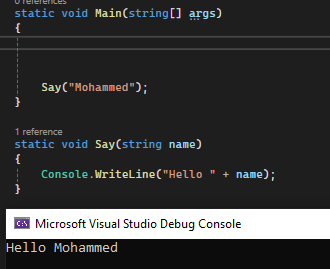
Console.WriteLine("Welcome Mohammed!");

Output:

Welcome Mohammed!

**User-defined function Example:**

After defined out the maine you can call

It anytime and from any place in the 

code .

void Say(string name)

{

Console.WriteLine("Hello " + name);

}

say("Mohammed");

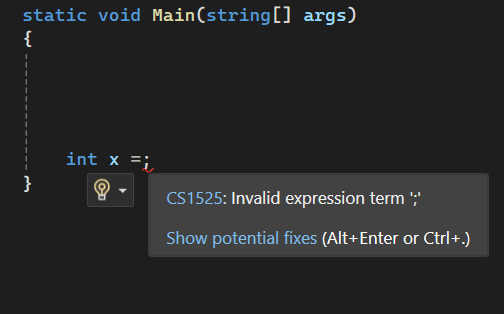
Output:

Hello Mohammed

7. Errors in Programming

Errors are problems in code that prevent the program from running correctly.

These categories are 3 categories :

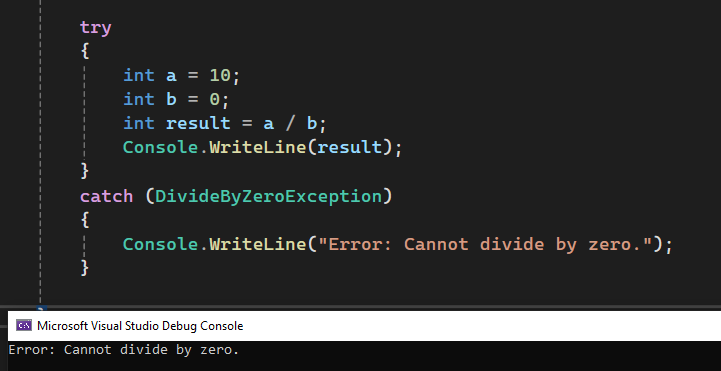
**1. Syntax Error :** it can be found by the compiler .

Example and Fix:

int x = ; // ❌ Syntax Error

Fixed:

int x = 5;

**2. Runtime Error** : it can be found after running the program and can be fix by the try built in functions .

Example and Fix:

try

{

int a = 10;

int b = 0;

int result = a / b;

Console.WriteLine(result);

}

catch (DivideByZeroException)

{

Console.WriteLine("Error: Cannot divide by zero.");

}

Output:

Error: Cannot divide by zero.

**3. Logic Error**: hard to be found only after testing the program and do it manually

Example and Fix:

int length = 5;

int width = 4;

int area = length + width;

wrong logic

Fixed:

area = length \* width;

Console.WriteLine("Area: " + area);

Output:

Area: 20