1)

//int x = 10;

/\*int y = 20;

int sum = x + y;

Console.WriteLine(sum);\*/

2) ctrl + /

3)

int x = "10";

console.WriteLine(x + y);

ERROR: 1- cannot convert from int to string

2- y variable not declared

3- console doesn’t exist

Correct Code:

int x = 10 , y = 5;

Console.WriteLine(x + y);

4)

A **runtime error** occurs during program execution, such as dividing by zero or accessing an invalid array index (e.g., int x = 1 / 0).  
A **logical error** produces incorrect results due to flawed logic, like using the wrong formula (e.g., area = length + width; instead of length \* width).

5)

string Name = "Ahmed Mostafa";

int Age = 20;

double MonthlySalary = 5000.12;

bool IsStudent = true;

Console.WriteLine("Name: " + Name + "\n" + "Age: " + Age + "\n" + "MonthlySalary: " + MonthlySalary);

Console.WriteLine("Is student: " + IsStudent);

6)

Following naming conventions like **PascalCase** in C# is important because:

1. **Readability**: Consistent naming makes code easier to read and understand for you and other developers.
2. **Maintainability**: Adhering to conventions helps maintain a uniform style, simplifying collaboration and debugging.

For example, EmployeeName is clearer and more professional than employeename or EMPLOYEENAME.

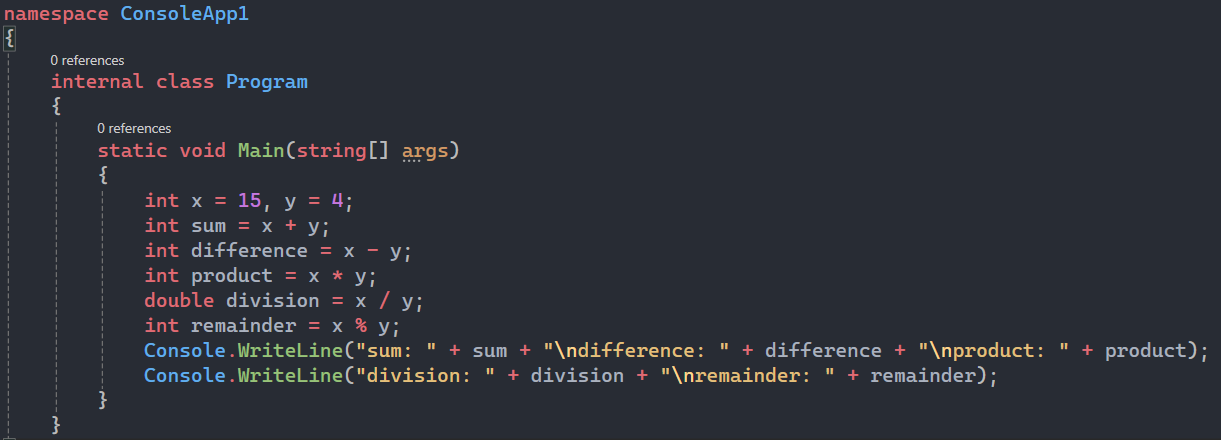
A computer screen shot of a program

Description automatically generated7)

8)

**Value types** are stored in the stack, where the actual data is kept, and each variable has its own copy (e.g., int, bool).  
 **Reference types** are stored in the heap, with variables holding a reference (or pointer) to the memory location of the actual data (e.g., class, string).  
 Modifying a reference type through one variable affects all references to the same object, unlike value types.

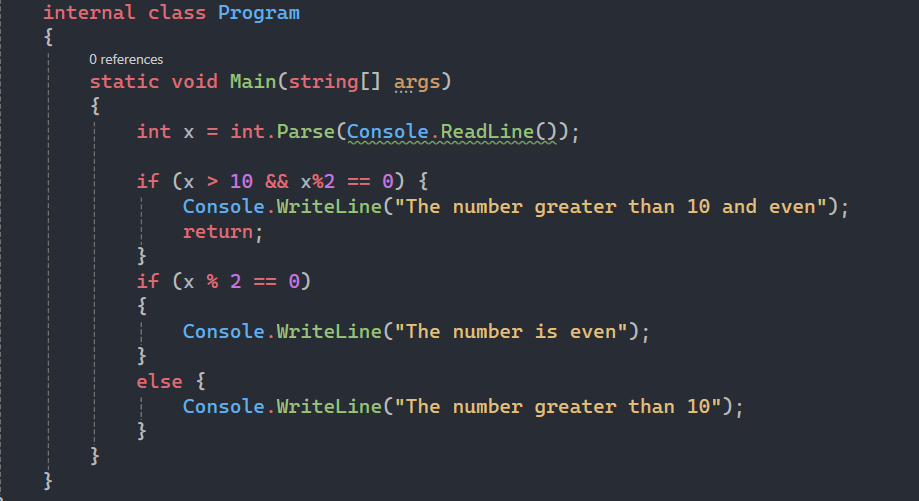
9)



10)

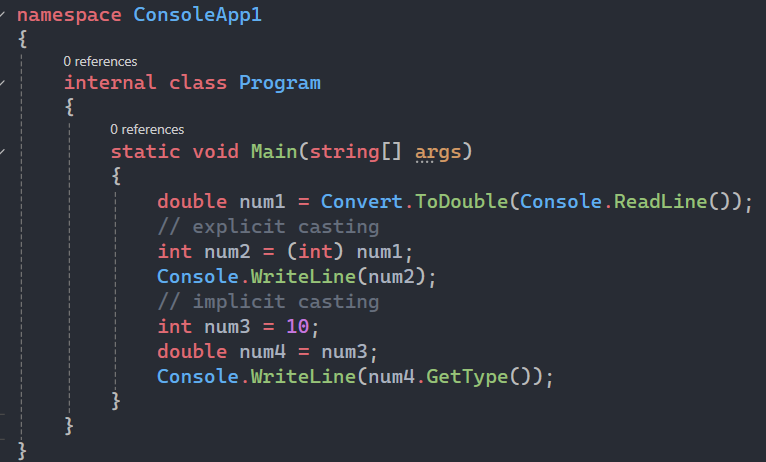
Since 2 divided by 7 results in a quotient of 0 (because 7 is larger than 2), the remainder is 2.

11)



12) && compare between numbers, but & compare between number in binary

13)



14)

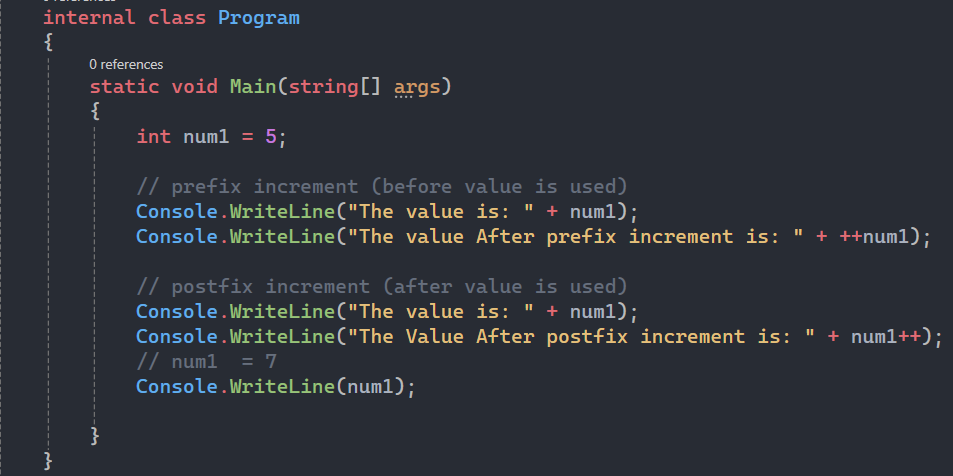
Explicit casting is required when converting a double to an int because the double type can hold decimal values and converting it to an int would truncate the fractional part. This ensures type safety by making the programmer explicitly acknowledge the potential data loss.

15)

A computer screen shot of a program

Description automatically generated

16) FormatException

17) 

18)

Question: Given the code below, what is the value of x after execution? Explain why int x = 5;

int y =++x + x++;

value of x is 7 when make ++x the value of x is 6 then after postfix increment value of x is 7

19)

Compiled languages are converted directly into machine code by a compiler, making them faster to execute. Interpreted languages are executed line-by-line by an interpreter, which can make them slower. C# is a **compiled** language, but it is first compiled into Intermediate Language (IL) and then interpreted by the Common Language Runtime (CLR).

20)

**Implicit casting** is automatically performed by the compiler when there is no loss of data (e.g., from int to double).

**Explicit casting** requires the developer to manually cast between types when there's a potential for data loss (e.g., from double to int).

**Convert** methods provide a safe way to change types with built-in handling for different formats, while **Parse** methods convert strings to types but throw exceptions if the conversion fails.