14. Which operator can be overloaded in C#?

A) `==`

B) `+=`

C) `++`

D) Both A and C

Answer: D) Both A and C

Explanation: Operators like `==`, `+`, `++`, and others can be overloaded in C#. However, `+=` cannot be directly overloaded.

15. What will the following code output?

```csharp
int a = 10;
int b = a << 2;
Console.WriteLine(b);
```

A) 20

B) 40

C) 10

D) Compilation Error

Answer: B) 40

Explanation: The left shift operator (`<<`) shifts the bits of `a` by 2 positions to the left, effectively multiplying by `2^2`. So, `10 << 2 = 40`.

3. **Which operator has the highest precedence in C#?**

A) `*`

B) `+`

C) `()`

D) `%`

Answer: C) `()`

Explanation: Parentheses `()` have the highest precedence and are used to explicitly group and prioritize parts of an expression.

6. **What is the correct precedence order for the following operators?**

A) `*`, `/`, `+`, `-`

B) `/`, `*`, `+`, `-`

C) `+`, `-`, `*`, `/`

D) `*`, `+`, `-`, `/`

Answer: A) `*`, `/`, `+`, `-`

Explanation: In C#, `*` and `/` have higher precedence than `+` and `-`. Operators of the same precedence are evaluated left to right.

8. What happens if we divide an integer by zero?

csharp   Copy code

```csharp
int result = 10 / 0;
Console.WriteLine(result);
```

A) Prints `Infinity`

B) Throws a `DivideByZeroException`

C) Compilation Error

D) None of the above

Answer: B) Throws a `DivideByZeroException`

Explanation: Division by zero for integers in C# results in a `DivideByZeroException`.

10. What is the output of this mixed operation?

csharp   Copy code

```csharp
double result = 10 / 4;
Console.WriteLine(result);
```

A) 2.5

B) 2

C) 2.0

D) Compilation Error

Answer: B) 2

Explanation: Since both operands (`10` and `4`) are integers, integer division is performed, and the result is truncated to `2`. For a floating-point result, use `10.0 / 4`.

12. **Which of the following expressions has the highest priority in C#?**

A) `a + b`

B) `a * b`

C) `a / b`

D) `a % b`

Answer: B) `a * b`, C) `a / b`, and D) `a % b`

Explanation: Multiplication, division, and modulus have the same precedence and are higher than addition or subtraction.

15. **Which operators have the lowest precedence in C#?**

A) `+` and `-`

B) `=`

C) `*` and `/`

D) `%`

Answer: B) `=`

Explanation: Assignment operators like `=` have the lowest precedence, evaluated last in expressions.

2. What is the output of this code?

csharp Copy code

```csharp
double result = 10 / 4 + 3.0;
Console.WriteLine(result);
```

5

A) 5.0

B) 5.5

C) 6.0

D) 6.5

4. What will this code produce?

csharp Copy code

```csharp
int a = 5;
int b = 2;
double result = a / b + 0.5;
Console.WriteLine(result);
```

2.5

A) 2.0

B) 2.5

C) 3.0

D) Compilation Error

Advanced Operator Priorities

For priority-based operations, the following rules are crucial:

1. **Parentheses** (`()`): Always evaluated first.
2. **Unary Operators** (`++`, `--`, `+`, `-`, `~`, `!`): Next in precedence.
3. **Multiplicative Operators** (`*`, `/`, `%`): Higher than additive.
4. **Additive Operators** (`+`, `-`): Lower than multiplicative.
5. **Relational and Equality Operators** (`<`, `>`, `==`, `!=`).
6. **Logical Operators** (`&&`, `||`): Evaluated after mathematical operations.
7. **Assignment Operators** (`=`, `+=`, `-=`, etc.): Lowest precedence.

15. What is the purpose of a `goto` statement in loops?

A) To terminate a loop

B) To jump to a labeled statement

C) To create infinite loops

D) None of the above

Answer: B) To jump to a labeled statement

Explanation: The `goto` statement transfers control to the labeled statement, but its use is discouraged for clarity and maintainability.

10. What will the following code output?

csharp Copy code

```csharp
int x = 0;
do
{
    Console.WriteLine(x);
    x++;
} while (x < 0);
Console.WriteLine(x);
```

A) 0

B) 0 1

C) Infinite loop

D) None of the above

Answer: B) 0 1

Explanation: The `do-while` loop executes once before checking the condition. It prints `0` and then increments `x` to `1`.

Summary of Common Runtime Errors

Error Type	Description	Example
NullReferenceException	Occurs when trying to access methods or properties of a null object.	`string name = null; Console.WriteLine(name.Length);`
DivideByZeroException	Occurs when attempting to divide a number by zero.	`int result = 10 / 0;`
IndexOutOfRangeException	Occurs when attempting to access an element of a collection using an invalid index.	`int[] arr = {1, 2, 3}; Console.WriteLine(arr[5]);`
FileNotFoundException	Occurs when attempting to access a file that does not exist.	`File.ReadAllText("nonexistentfile.txt");`

By handling exceptions properly and ensuring you account for possible edge cases (e.g., null values,

Error Type	Cause	Example
Syntax Errors	Code violates the grammar or syntax rules of C#	Missing semicolons, mismatched parentheses, etc.
Undeclared Variable	A variable is used without being declared first.	Using `x` without declaring it first.
Incompatible Types	Trying to assign incompatible data types.	Assigning an `int` to a `string` or vice versa.
Missing Method Arguments	Calling a method with the wrong number of arguments.	Not passing required arguments to a method.
Invalid Access Modifiers	Using inappropriate access modifiers for methods or fields.	Trying to access a `private` method from another class.
Method Overloading Ambiguity	Having multiple methods with similar or identical signatures, causing ambiguity.	Methods with similar parameters causing overload conflicts.
Incorrect Inheritance	A class does not implement all required methods of an interface or does not inherit properly.	Missing method implementations from an interface.
Ambiguous Namespace	Conflicting class names or namespaces.	Using `MyClass` when there are multiple classes named `MyClass`.
Type Cannot Be Used as a Type Parameter	Using a type as a type parameter where it's not allowed.	Using a non-generic type as a generic type parameter.

By understanding and addressing these comm compile-time errors, you can ensure your code

6. Which of the following is the main cause of a StackOverflowException in C#?

A) A logic error in the code
B) An unhandled exception in the try-catch block
C) Excessive recursion without a base case
D) Incorrect type casting

Correct Answer: C) Excessive recursion without a base case
Explanation: A `StackOverflowException` occurs when a method calls itself recursively without a termination condition, leading to excessive memory usage in the call stack.

14. What will happen if you attempt to cast a `string` to an `int` in C# without proper conversion methods?

A) A compile-time error will occur.
B) A `FormatException` will be thrown at runtime.
C) It will result in a `NullReferenceException`.
D) The program will automatically convert the string to an integer.

Correct Answer: B) A `FormatException` will be thrown at runtime.
Explanation: A `FormatException` occurs if you try to cast a `string` to an `int` directly, and the string cannot be parsed as an integer.

9. **Which keyword is used to define an immutable variable in C#?**

A) `static`

B) `readonly`

C) `const`

D) `volatile`

Answer: C) `const`

12. **Which of the following C# data types can hold the largest range of values?**

A) `float`

B) `double`

C) `decimal`

D) `long`

Answer: B) `double`

4. What happens if you try to cast an incompatible type using explicit casting?

A) The value is converted to `null`.

B) A compile-time error occurs.

C) A runtime exception is thrown.

D) The program continues with undefined behavior.

Answer: C) A runtime exception is thrown.

6. Which method from the `Convert` class can be used to convert a `string` to an `int`?

A) `Convert.ToInt()`

B) `Convert.ToInt32()`

C) `int.Parse()`

D) Both B and C

Answer: D) Both B and C

7. **What is the difference between `Convert.ToInt32()` and `int.Parse()`?**

A) `int.Parse()` works only with `string`, while `Convert.ToInt32()` can handle other types.

B) `Convert.ToInt32()` throws an exception on invalid input, while `int.Parse()` returns 0.

C) `int.Parse()` can handle `null`, but `Convert.ToInt32()` cannot.

D) They are functionally identical.

Answer: A) `int.Parse()` works only with `string`, while `Convert.ToInt32()` can handle other types.

9. **What will happen if you try to parse a non-numeric string to an integer using `int.Parse()`?**

A) It returns 0.

B) It throws a `FormatException`.

C) It converts the string to ASCII codes.

D) It results in undefined behavior.

Answer: B) It throws a `FormatException`.

11. What is the output of the following code?

csharp Copy code

```csharp
double d = 12.6;
int i = Convert.ToInt32(d);
Console.WriteLine(i);
```

A) 12

B) 13

C) 12.6

D) Runtime Error

Answer: B) 13

10. What is the result of the following code?

csharp Copy code

```csharp
int x = 10;
int y = 3;
Console.WriteLine(x / y);
```

A) 3.3333

B) 3

C) 3.0

D) Compilation Error

Answer: B) 3

6. What is the output of the following code?

csharp

Copy code

```csharp
double d = 5.2;
int i = (int)d;
Console.WriteLine(i);
```

A) 5.2

B) 5

C) 6

D) Compilation Error

Answer: B) 5

13. What is the output of the following code?

csharp

Copy code

```csharp
string input = "ABC";
int result;
bool isParsed = int.TryParse(input, out result);
Console.WriteLine(isParsed);
```

A) `True`

B) `False`

C) Compilation Error

D) Runtime Error

Answer: B) `False`

1. What will be the output of the following code?

csharp Copy code

```csharp
string s = "123.45";
int x = Convert.ToInt32(s);
Console.WriteLine(x);
```

A) 123

B) 124

C) Compilation Error

D) Runtime Error

Answer: D) Runtime Error

Explanation: `Convert.ToInt32` cannot convert a string with a decimal point directly to an integer; it throws a `FormatException`.

2. Identify the issue in this code snippet:

csharp Copy code

```csharp
double d = 10.5;
int i = d;
Console.WriteLine(i);
```

A) Compilation Error

B) Runtime Error

C) Prints `10`

D) Prints `11`

Answer: A) Compilation Error

Explanation: Direct assignment of `double` to `int` is not allowed in C#. Explicit casting `(int)d` is required.

3. **What is the output of the following code?**

csharp

```csharp
object obj = 42;
string str = (string)obj;
Console.WriteLine(str);
```

A) "42"

B) Compilation Error

C) Runtime Error

D) NullReferenceException

Answer: C) Runtime Error

Explanation: You cannot cast an `object` containing an `int` directly to a `string`. A `InvalidCastException` will be thrown.

4. **What will happen if this code is executed?**

csharp

```csharp
string input = null;
int result = int.Parse(input);
Console.WriteLine(result);
```

A) Prints 0

B) Compilation Error

C) Throws a `FormatException`

D) Throws an `ArgumentNullException`

use Convert

Answer: D) Throws an `ArgumentNullException`

Explanation: `int.Parse` cannot handle a `null` value and throws `ArgumentNullException`.

11. Identify the problem with this code:

csharp Copy code

```csharp
string str = "123";
long l = long.Parse(str);
int i = (int)l;
Console.WriteLine(i);
```

A) Compilation Error

B) Runtime Error

C) Prints 123

D) Works fine unless the value exceeds the `int` range

Answer: D) Works fine unless the value exceeds the `int` range

Explanation: Casting from `long` to `int` works, but if the value is out of the `int` range, it will throw an `OverflowException`.

16. What will happen in this scenario?

csharp Copy code

```csharp
object obj = "123";
int num = (int)obj;
Console.WriteLine(num);
```

A) Prints `123`

B) Compilation Error

C) Runtime Error

D) Outputs garbage value

Answer: C) Runtime Error

Explanation: The object contains a `string`, which cannot be directly unboxed as an `int`. Use `int.Parse((string)obj)` or `Convert.ToInt32(obj)`.

19. **What happens in this case?**

```csharp
string str = "true";
bool flag = bool.Parse(str);
Console.WriteLine(flag);
```

A) Prints `true`

B) Prints `False`

C) Compilation Error

D) Throws a `FormatException`

Answer: A) Prints `true`

Explanation: `bool.Parse` converts valid string representations of `true` or `false` (case insensitive) to their respective boolean values.

21. **Debug this type conversion issue:**

```csharp
int i = 300;
byte b = (byte)i;
Console.WriteLine(b);
```

A) Prints `300`

B) Compilation Error

C) Runtime Error

D) Prints `44`

Answer: D) Prints `44`

Explanation: Casting `int` to `byte` can result in data loss if the value exceeds the `byte` range (`0-255`). The value wraps around, and `300 % 256 = 44`.

Would you like to dive deeper into any of these? 😊

2. **Which operator is used for conditional (ternary) operations in C#?**

A) `??`

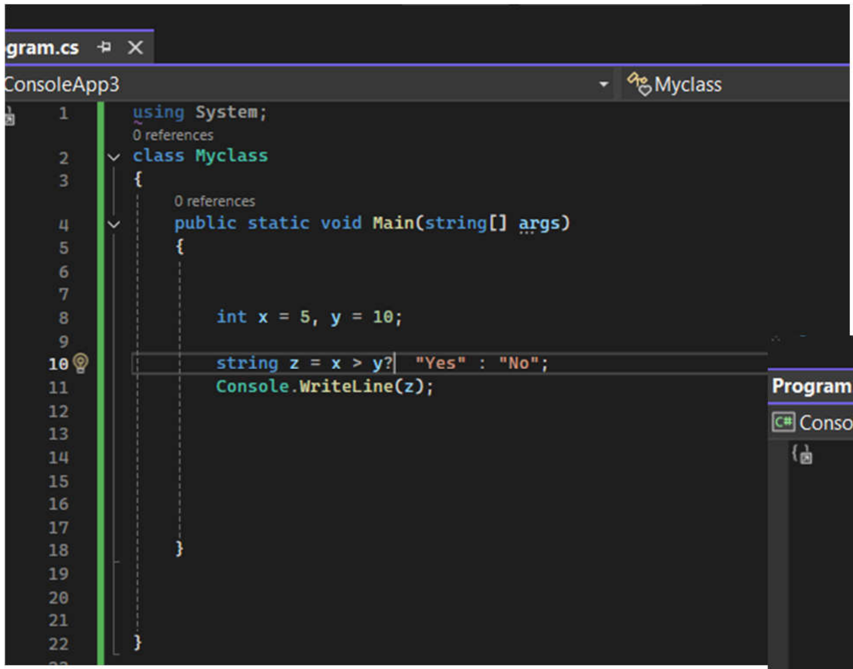
B) `?:`

C) `&&`

D) `||`

Answer: B) `?:`

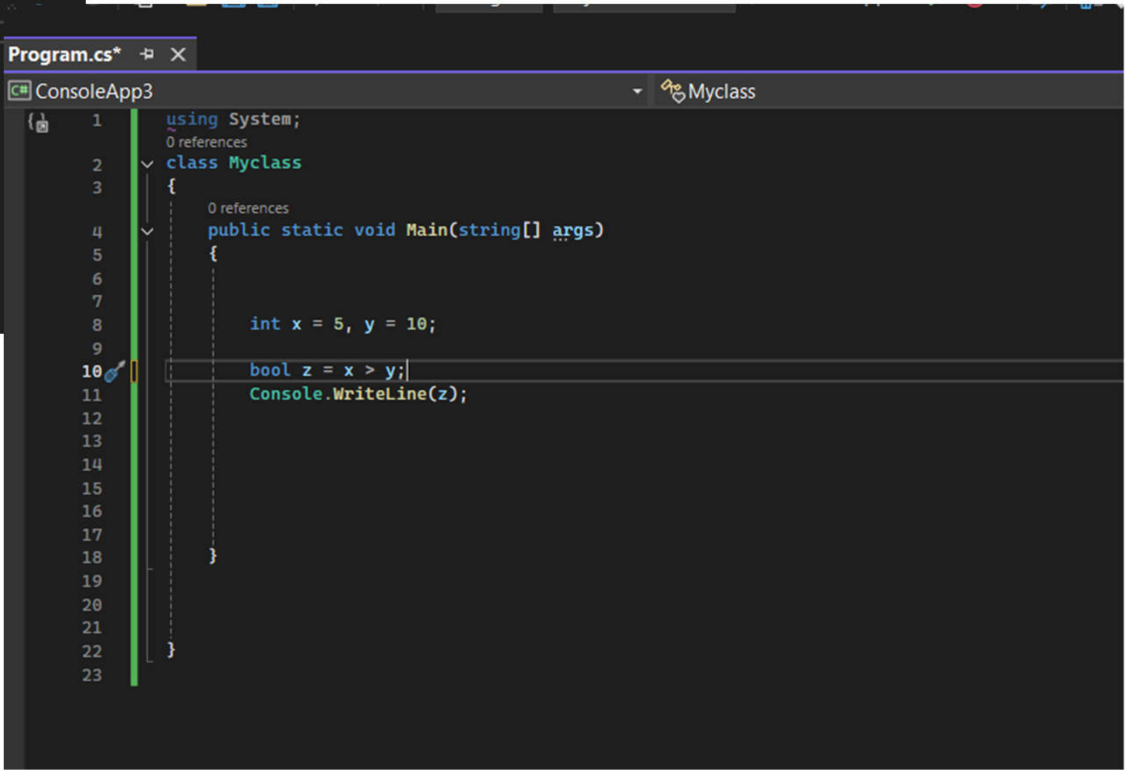
Explanation: The conditional operator is written as `?:` and is used for expressions like `condition ? value_if_true : value_if_false`.



gram.cs

ConsoleApp3 · Myclass

```
using System;
0 references
class Myclass
{
    0 references
    public static void Main(string[] args)
    {


        int x = 5, y = 10;

        string z = x > y?  "Yes" : "No";
        Console.WriteLine(z);






    }



}
```



Program.cs*

ConsoleApp3 · Myclass

```
using System;
0 references
class Myclass
{
    0 references
    public static void Main(string[] args)
    {


        int x = 5, y = 10;

        bool z = x > y;
        Console.WriteLine(z);






    }



}
```

3. **What is the output of this code?**

```csharp
int x = 5, y = 10;
Console.WriteLine(x > y ? "Yes" : "No");
```

A) Yes

B) No

C) True

D) False

Answer: B) No

Explanation: Since `x` (5) is not greater than `y` (10), the condition evaluates to `false`, and "No" is printed.

12. **What does the `??` operator do in C#?**

A) Checks if a value is not null

B) Returns the left-hand operand if it's not null; otherwise, returns the right-hand operand

C) Performs a bitwise OR

D) Throws a `NullReferenceException` if a value is null

Answer: B) Returns the left-hand operand if it's not null; otherwise, returns the right-hand operand

14. Which operator can be overloaded in C#?

A) `==`

B) `+=`

C) `++`

D) Both A and C

Answer: D) Both A and C

Explanation: Operators like `==`, `+`, `++`, and others can be overloaded in C#. However, `+=` cannot be directly overloaded.