SDEV2301

C# Fundamentals

AN INSTITUTE OF TECHNOLOGY COMMITTED TO STUDENT SUCCESS

1. The Parts of a C# Program

CONCEPT: A C# program has parts that serves specific purposes

Some C# Rules

- C# is a case-sensitive language.
- Comments are ignored by the compiler.
- A C# programs must be stored in a file with a named that ends with **`.cs`**.
- Every C# application program must have a method named **`Main`**.
- For every left brace, or opening brace, there must be a corresponding right brace, or closing brace.
- Statements are terminated with semicolons.

Dynamic vs Static Typing

Python & JavaScript:

- Variable type can change at runtime
- Errors often occur while the program is running

C#:

- Variable type is fixed at declaration
- Errors are caught at compile time
- The compiler enforces correctness

Python	C#

`x = 5`
`x = "hello" # allowed`
	`int x = 5;`
	`x = "Hello"; // compile-time error`

The C# Compiler Is Not Being Picky -- It's Protecting You

In C#:

- Many errors are caught before the program runs
- Red squiggles means a rule is being violated
- Fixing compile-time errors saves runtime failures
- This is why C# feels stricter than Python or JavaScript at first.

The is a feature, not a limitation.

var Does not Mean "Dynamic"

```
var count = 10;     // count is an int
var price = 9.99;   // price is a double
// This will not compile
count = "ten";
```

- var still uses static typing
- The type is inferred at compile time
- Once inferred, it cannot change

Explicit warning: This is NOT JavaScript var.

Integer Division: A Common Surprise

```
double result = 5 / 2;    // result is 2
double result2 = 5.0 / 2; // result is 2.5
```

- If both operands are integers → integer division
- The decimal portion is discarded
- This differs from Python and JavaScript

Special Characters

Character	Name	Meaning
{ }	Opening and closing braces	Encloses a group of statements, such as contents of a class or a method
( )	Opening and closing parentheses	Used in a method header
;	Semicolon	Marks the end of a computer programming statement
" "	Double quotation marks	Encloses a string of characters, such as a message that is to be printed on the screen
' '	Single quotation marks	Encloses a single character
//	Double Slash	Marks the beginning of a line comment
/* */	Slash Asterisk	Encloses a block of comments

Output & Escape Sequences

- Console.Write writes without a new line
- Console.WriteLine writes and moves to a new line
- Escape sequences behave the same as Python/JS

  `\n` `\t` `\"` `\\`

```
Console.WriteLine("Item\tPrice");
```

3. Variables and Literals

A variable in C#:

- Has a name AND a fixed data type
- Can only store values of that type
- Cannot change type after declaration

```
int count = 5;
count = 10;    // OK
count = "ten"; // Compile-time error
```

Formatting Output

- Three techniques for formatting a string with literal values and variable values:
  - String Concatenation using the + operator
  - Composite Formatting using the {positionNumber} placeholder
  - String Interpolation using an interpolated String $"{expression} literalValue"

```
 string name = "Uncle Bob";
 int age = 69;
```

- String Concatenation example:

```
Console.WriteLine("My name is " + name + " and I am " + age + " years old.");
```

- Composite Formatting example:

```
Console.WriteLine("My name is {0} and I am {1} years old.", name, age);
```

- String Interpolation example:

```
Console.WriteLine($"My name is {name} and I am {age} years old.");
```

Identifier Naming Rules

- The first character must be a letter (a-z or A-Z), or an underscore (_).
- After the first character, you may use letters, digits, or underscore (_)
- Uppercase and lowercase characters are distinct. This means **`itemsOrdered`** is not the same as **`itemsordered`**
- Identifiers cannot include spaces

Choosing Numeric Types in C#

- `int` → whole numbers
- `double` → general decimals
- `decimal` → money and financial values

IntegerVariables.cs

```
int checking;   // Declare an int variable named checking
long days;      // Declare a long variable named days

checking = -20;
days = 189000;

Console.WriteLine($"Our account balance is ${checking}");
Console.WriteLine($"About {days} days ago Columbus stood on this spot.");
```

Program Output

```
Our account balance is $-20
About 189000 days ago Columbus stood on this spot.
```

Sale.cs

```
double    price,
          tax,
          total;

price = 29.75;
tax = 1.76;
total = 31.51;

Console.WriteLine($"The price of the item is {price}");
Console.WriteLine($"The tax is {tax:C}");
Console.WriteLine($"The total is {total:C}");
```

Program Output

```
The price of the item is 29.75
The tax is 1.78
The total is 31.51
```

TrueFalse.cs

```
bool isHot;

isHot = true;
Console.WriteLine(isHot);
isHot = false;
Console.WriteLine(isHot);
```

Program Output

```
true
false
```

5. Arithmetic Operators

CONCEPT: There are many operators for manipulating numeric values and performing arithmetic operations.

- Three types of operators: unary, binary, ternary
- Unary operators require only a single operand.
- Binary operators work with two operands
- Ternary operators work with three operands. The conditional operator (`?:`) for example.

Wages.cs

```
double regularWages;        // The calculated regular wages
double basePay = 25;        // The base pay rate
double regularHours = 40;   // The hours worked less overtime.
double overtimeWages;       // Overtime wages
double overtimePay = 37.5;  // Overtime pay rate
double overtimeHours = 10;  // Overtime hours worked
double totalWages;          // Total wages

regularWages = basePay * regularHours;
overtimeWages = overtimePay * overtimeHours;
totalWages = regularWages + overtimeWages;
Console.WriteLine($"Wages for this week are {totalWages}");
```

Program Output

```
Wages for this week are $1375.0
```

Integer Division (Detailed)

- When both operands of the division operator are integers, the operator will perform integer division.
- The result of integer division is an integer, if there is a remainder, it will be discarded.

```
double number;
number = 5 / 2;         // result is 2 not 2.5
```

- Convert one of the operands to floating point number if you want a floating-point result.

```
double number;
number = 5.0 / 2;       // result is 2.5
```

The Math Class

- The Math class contains methods that are useful for performing complex mathematical operations.
- Use the **`Math.Pow`** method to return a specified number raise to the specified power

```
result = Math.Pow(2,3);// result is 8
```

- Use the **`Math.Sqrt`** method, to return the square root of a specific number

```
result = Math.Sqrt(9); // result is 3
```

- Use the **`Math.Round`** method, to round to nearest integer or to the specified number of fractional digits

```
result = Math.Round(1.3768); // result is 1
result = Math.Round(1.3768, 2); // result is 1.38
```

6. Combined Assignment Operators

CONCEPT: The combined assignment operators combine the assignment operator with the arithmetic operators.

Operator	Example Usage	Equivalent to
+=	x += 5;	x = x + 5;
-=	y -= 2;	y = y - 2;
*=	z *= 10	z = z * 10;
/=	a /= b;	a = a / b;
%=	c %= 3;	c = c % 3;

7. Conversion between Simple Data Types

CONCEPT: Before a value can be stored in a variable, the value's data type must be compatible with the variable's data type.

- Use ***cast operator*** to manually convert a value

```
double doubleMark = 88.88;
int intMark = (int) doubleMark; // = 88
double doubleResult = (double) intMark / 10; // = 8.8
```

8. Creating Named Constants with const

CONCEPT: The **const** key word can be used in a variable declaration to make the variable constant. Named constants are initialized with a value, and that value cannot change during the execution of the program.

```
const double GstRate = 0.05;
const int MinAge = 18;
const int MaxAge = 65;
```

9. The `string` Class

CONCEPT: The **`string`** class allows you to create objects for holding strings. It also has various methods that allow you to work with strings.

StringDemo.cs

```
static void Main(string[] args)
{
    string greeting = "Good morning ";
    string name = "Herman";

    Console.WriteLine($"{greeting} {name}");
}
```

Program Output

```
Good morning, Herman
```

10. Scope

CONCEPT: A variable's scope is the part of the program that has access to the variable.

```
static void Main(string[] args)
{
  Console.WriteLine(value); // Error
  int value = 100;
}
```

11. Comments

CONCEPT: Comments are notes of explanation that document lines or sections of a program. Comments are part of the program, but the compiler ignores them. They are intended for people who may be reading the source code.

- Single-Line Comments (**//**)

```
// This is a line comment
```

- Multi-Line (Block) Comments (**/* */**)

```
/*
This is a comment
This is a comment
*/
```

12. Programming Style

CONCEPT: Programming style refers to they way a programmer uses spaces, indentation, blank lines, and punctuation characters to visually arrange a program's source code.

13. Reading Keyboard Input (1)

CONCEPT: The **`Console.ReadLine`** method reads the next line of characters from the keyboard.

```
string name;
Console.WriteLine("What is your name?");
name = Console.ReadLine();
```

13. Reading Keyboard Input (2)

CONCEPT: The **`Console.ReadLine`** method always returns a **string**, if you are expecting a number you must **parse** it.

```
string inputValue;
int age;
double income;
Console.WriteLine("What is your age?");
inputValue = Console.ReadLine();
age = int.Parse(inputValue);
Console.WriteLine("What is your income?");
inputValue = Console.ReadLine();
income = double.Parse(inputValue);
```

14. Common Errors to Avoid

- Mismatched braces, quotation marks, or parentheses
- Misspelling key words
- Using capital letters in key words
- Using a key word as a variable name
- Using inconsistent spelling of variable names
- Using inconsistent case of letters in variable names
- Inserting a space into a variable name
- Forgetting a semicolon at the end of the statement
- Assigning a `double` literal to a `float` variable
- Using commas or other currency symbols in numeric literals
- Unintentionally performing integer division
- Forgetting to group parts of a mathematical expression
- Inserting a space in a combined assignment operator