CSharp Chapter 1 Course Documentation Report

- **Programming Paradigms:**
  Procedural programming is a programming style that organizes code into reusable functions or procedures. It follows a step-by-step approach, where each function performs a specific task and can be reused throughout the program. This paradigm makes the code more structured and easier to debug. However, as programs grow larger, managing functions and global variables can become complex. Examples of procedural programming languages include C and early Python.
  Object-Oriented Programming (OOP) is a paradigm that organizes code into objects, which represent real-world entities. It is based on four key principles: Encapsulation(hiding data), Inheritance (reusing code), Polymorphism (flexibility in code behavior), and Abstraction (hiding implementation details). OOP makes programs more reusable, scalable, and easier to maintain. Popular OOP languages include C#, Java, and Python.
- **Algorithm Programming Components:**
  An algorithm is a step-by-step process to solve a problem. It consists of input (data given to the algorithm), output (the expected result), and instructions (a set of rules or steps to process the input). It follows control structures, such as sequences, loops, and decision-making (`if-else`).
- **Types of Programming Languages (Based on Machine Interaction & Execution) :**
  Programming languages are categorized based on how they interact with the machine and how they execute code. Low-level languages like machine language and assembly interact directly with the hardware, making them fast but difficult to understand.
  High-level languages like C#, C, Java, and Python are easier to use but require a compiler or interpreter to convert them into machine code. Based on execution, languages are either compiled (C, C++)—translated fully before running, or interpreted (Python, JavaScript)—executed line by line. Some, like C# and Java, use both compilation and interpretation.
- **Comparison of Compiler and Interpreter :**
  A compiler translates the entire program into machine code before execution, making it faster but requiring more time for compilation (e.g., C#, C++). An interpreter translates and executes code line by line, which makes it slower but easier to debug (e.g., Python,

JavaScript). Compiled languages run efficiently after compilation, while interpreted languages allow immediate execution without needing a separate compilation step.

- **Data types, Variables, Constants and input/ output statements:**
  C# like the other programming languages have Data types such as string (ex. String name),integer(ex. Int num), float (ex. Float num1), and double (ex double num2). Variables are containers for storing data values, there are different ways to write it like what I wrote above and this is the general syntax `type variableName = value;` . constant is useful when you want a variable to store the same value and there are two ways to write it `const int myNum = 15;` .input is how the user can enter values to the program like this way using console `string userName = Console.ReadLine();` , and the output how to display the result in the console `Console.WriteLine("Enter username:");`
- **Arithmetic and logic operators:**
  Operators are special types of symbols which perform operations on variables or values. There are different types of operation such as arithmetic operators which used to perform basic mathematical operations on numeric values like addition ,subtraction,Multiplication,division and modulus this the way to write it in the program:

```
int x = 8, y = 4;
        // Using different arithmetic operators
        Console.WriteLine("Addition: " + (x + y));
        Console.WriteLine("Subtraction: " + (x - y));
        Console.WriteLine("Multiplication: " + (x * y));
        Console.WriteLine("Division: " + (x / y));
        Console.WriteLine("Modulo: " + (x % y));
```

The second type of operation is logical operators which is used when multiple conditions and there we can combine these to compare complex conditions:
Logical AND (&&) : returns true when both conditions are true.
Logical OR (||) : returns true if at least one condition is true.
Logical NOT (!): returns true when a condition is false and vice-versa.

```
bool a = true, b = false;
        // conditional operators
        if (a && b)
            Console.WriteLine("a and b are true");
        if (a || b)
            Console.WriteLine("Either a or b is true");
        if (!a)
            Console.WriteLine("a is not true");
        if (!b)
```

```
            Console.WriteLine("b is not true");
```

- **Conditional statements (if else / switch case):**

  we use the conditional to make a decision or choice of somethings in programming language we have if else and switch conditional

  If else:

```
string name = "Mohammed";
            // Using if-else statement
            if (name == "Mohammed") {
                  Console.WriteLine("My name is Mohammed");
            }
            else {
                  Console.WriteLine("other names");
            }
```

  Switch:

```
int number = 30;
            // Using Switch case
            switch(number)
            {
                  case 10:
                  Console.WriteLine("case 10");
                        break;
                  case 20:
                  Console.WriteLine("case 20");
                        break;
                  case 30:
                  Console.WriteLine("case 30");
                        break;
                  default:
                  Console.WriteLine("None matches");
                        break;
            }
```

- **Looping (for, while and do while):**

  For loop used when you know exactly how many times want to loop through a code

```
for (int i = 0; i < 5; i++)
{
  Console.WriteLine(i);

}
```

  Do while loop this loop will execute the block code at least once before checking if the condition is true then it will repeat the loop as long as the condition is true

```
int i = 0;

do
{
  Console.WriteLine(i);
  i++;
```

```
}

while (i < 5);
```

While loop we use it for condition

```
int i = 0;
while (i < 5)
{
  Console.WriteLine(i);
  i++;

}
```

- **Nested operations (nested conditions and nested looping)**:

  nested means write code inside another code and we have nested condition such as nested if statement

```
int i = 10;
            if (i == 10) {
                  // nested - if statement
                  // will only be executed if statement
                  // above it is true
                  if (i < 12)
                        Console.WriteLine("i is smaller than 12
too");
                  else
                        Console.WriteLine("i is greater than 15");
            }
```

  And nested looping

```
// Outer loop
for (int i = 1; i <= 2; ++i)
{
  Console.WriteLine("Outer: " + i);  // Executes 2 times

  // Inner loop
  for (int j = 1; j <= 3; j++)
  {
    Console.WriteLine(" Inner: " + j); // Executes 6 times (2 * 3)
  }

}
```

- **Array data structure:**

  array is a type of a data structure that stores the same data types with fixed size.

```
int[] myNum = {10, 20, 30, 40};

// Create an array of four elements
string[] cars = new string[4];
```

- **Functions in CSharp (built-in):**

C# has many built-in function that can help for specific tasks such as mathematical calculations or array operations and more here the example:

Math.Round(x)

Math.Sqrt(x)

Math.Pow(x, y)

- **Errors in programming:**

  different error can occur when write program or when run the program so the c# language can handle this type of errors by using try and catch exception.there are different types of errors such as runtime error,logical error ,syntax error.

```
try
{
  int[] myNumbers = {1, 2, 3};
  Console.WriteLine(myNumbers[10]);
}
catch (Exception e)
{
  Console.WriteLine(e.Message);

}
```