**Tribhuvan University**

**Bhaktapur Multiple Campus**

Doodhpati-17, Bhaktapur, Nepal

Shape, arrow

Description automatically generated

**Lab report of**

**Web-Technology**

**(CSC 367)**

|  |  |
| --- | --- |
| **Submitted by:**  Saini Thapa Magar  B.Sc. CSIT 6th semester  Roll no.: 44 | **Submitted to:**  Ramesh Kharbuja |

**C# Program with Fundamental concepts**

**Introduction to .NET and C#**

.NET is an opensource developer platform, created by Microsoft, for building many different  
types of applications. With .NET, we can use multiple languages, editors, and libraries to build  
for web, mobile, desktop, games, IoT, and more. We can write .NET apps in C#, F#, or Visual  
Basics. We are using C# to build our projects with .NET Framework.

C# is a general- purpose, object- oriented programming language designed for Common  
Language Infrastructure (CLI), which consists of the executable code and the runtime  
environment that allows the use of various high- level languages on different computer  
platforms and architectures.

**Data Types:**Data type is used to represent the type of a variable. The Common Language Runtime (CLR)  
provides two categories of data types. They are Value Type and Reference Type. The value  
type stores its value in the stack and reference type stores its value in the managed heap. The  
value type consists of the following data types:

1. Value Type:

* Simple Type
* Enumeration Type
* Structure Type

1. Reference Type:

* Class Type
* String Type
* Delegate Type
* Interface Type
* Array Type

Simple value type includes the following:

1. Integer type:
2. Int – System.Int32
3. Short – System.Int16
4. Byte – System.Byte
5. Long – System.Int64
6. Floating type:
7. Float – System.Single
8. Double - System.Double
9. Decimal type - System.Decimal
10. Boolean type - System.Boolean
11. Character type - System.Char

**Operators in C#**

An operator is a symbol that tells the compiler to perform specific mathematical or logical  
manipulations. C# has rich set of built-in operators and provides the following type of  
operators:  
• Arithmetic Operators  
• Assignment Operators  
• Relational Operators  
• Logical Operators  
• Bitwise Operators

1. **Arithmetic Operators**The arithmetic operators perform arithmetic operations on the given operands. The arithmetic  
   operator includes:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Addition Operator | A + B |
| **-** | Subtraction Operator | A – B |
| **\*** | Multiplication Operator | A \* B |
| **/** | Division Operator | A / B |
| **%** | Modulo Operator (Finds remainder) | A % B |
| **++** | Increment Operator | A++ or B++ |
| **--** | Decrement Operator | A—or B-- |

1. **Assignment operators**

In C#, an assignment operator is used to assign a value to a variable. It is denoted by = and assigns the value of the right hand operand to a variable, a property or an indexer element given by its left hand operand. Assignment operators in C# are:

Simple Assignment (=), Addition Assignment (+=), Subtraction Assignment (-=), Multiplication Assignment (\*=), Division Assignment (/=), Modulus Assignment (%=), etc.

1. **Comparison Operators**Comparison operators are used to compare two values or variables. To make decisions,  
   comparison operators are used in specific scenarios. The return value of comparison operator  
   is either **true**or **false**. This data type is known as Boolean data type.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| == | Equal to | A == B |
| **!=** | Not equal | A != B |
| **>** | Greater than | A > B |
| **<** | Less than | A < B |
| **>=** | Greater than or equal to | A >= B |
| **<=** | Less than or equal to | A <= B |

1. **Logical Operators**

Logical operators are used to determine the logic between variables or values. Like comparison  
operator, it is also of Boolean type.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| && | Logical AND | A < 5 && B <10 |
| **||** | Logical OR | A < 5 || A < 4 |
| **!** | Logical NOT | !(A<5) |

1. **Bitwise operators**

C# provides four bitwise and two bit shift operators. Bitwise and bit shift operators are used to  
perform bit level operations on integer and Boolean data.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| ~ | Bitwise Complement | ~A |
| **&** | Bitwise AND | A & B |
| **|** | Bitwise OR | A | B |
| **^** | Bitwise Exclusive OR (XOR) | A ^ B |
| **<<** | Bitwise Left Shift | A << 1 |
| **>>** | Bitwise Right Shift | A >> 1 |

**Flow control of Programming**

Flow control refers to the order in which statements are executed in a program. It allows a program to make decisions, repeat operations, and respond to various inputs or conditions.

There are three main types of flow control in programming: selection statements, iteration statements, and jump statements.

1. **Selection statements:**

These statements allow the program to make decisions based on a condition. There are two types of selection statements in C#: if-else statements and switch statements.

1. **if-else** statements allow the program to execute different paths of code based on whether a certain condition is true or false.

**Syntax:**

if(condition)

{statements;}

else

{statements;}

1. **switch** statements allow the program to execute different paths of code based on the value of a variable or expression.

**Syntax:**

switch(argument)

{

Case 1:

//Do something

break;

Case 2:

//Do something

break;

Default:

//Do something

break;

}

1. **Iteration Statements:** These statements allow the program to execute a block of code repeatedly based on a certain condition. There are three types of iteration statements in C#: for loops, while loops, and do-while loops.
2. **for loops** allow the program to execute a block of code a fixed number of times.

**Syntax:**for(initializer; condition; iterator)

**{** statements;**}**

1. **while loops** allow the program to execute a block of code while a certain condition is true.

**Syntax:**while(condition)

**{**statements;}

1. **do-while loops** are similar to while loops, but the block of code is executed at least once, even if the condition is false.

**Syntax:**

do{

statements; //body

}while(condition)

1. **foreach loop** construct is a type of iteration statement in C# that allows you to iterate over a collection of items such as an array, a list, or any other object that implements the IEnumerable interface. It simplifies the process of iterating through the elements of a collection by automatically initializing a loop variable, iterating through each element of the collection, and then exiting the loop when all elements have been processed.

**Syntax:**

foreach (var item in collection)

{

// Code to be executed for each item in the collection

}

1. **Jumping Statements:** These statements allow the program to transfer control to another part of the program. There are three types of jump statements in C#: break, continue, and goto.
2. **break** statements allow the program to exit a loop early.
3. **continue** statements allow the program to skip over the current iteration of a loop and go to the next iteration.
4. **goto** statements allow the program to transfer control to a labeled statement in the code.

**Structs and Enum**

**Struct :** In C#, a struct is a value type that is similar to a class, but with some key differences. It is used to encapsulate small groups of related variables, which can then be passed around as a single unit. Structs are typically used for lightweight objects that do not require inheritance or other advanced features.

**Syntax:**

struct StructName

{

// Fields and methods

}

**Enums:** An enum is a special type in C# that allows you to define a set of named constants. Each constant has an underlying integer value that can be accessed using the enum type. Enums are often used to represent sets of related values, such as days of the week or colors.

**Syntax:**

enum EnumName

{

Constant1,

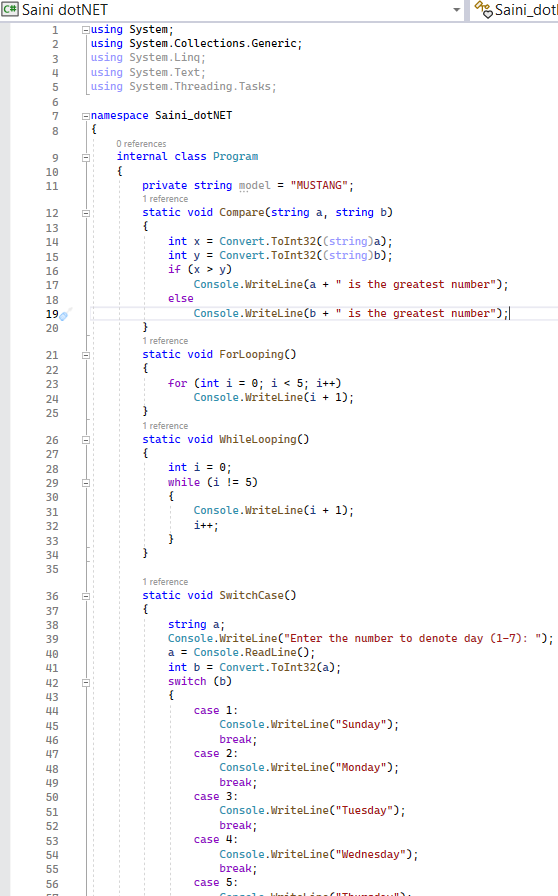
Constant2,

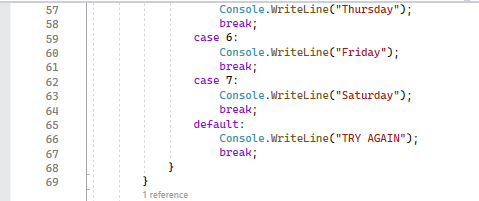
Constant3,

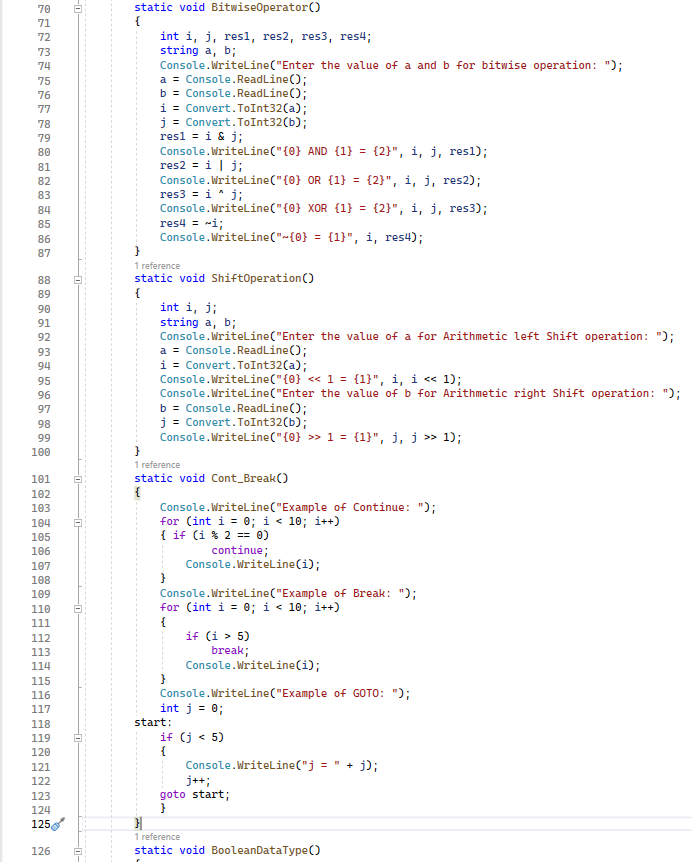
// ...

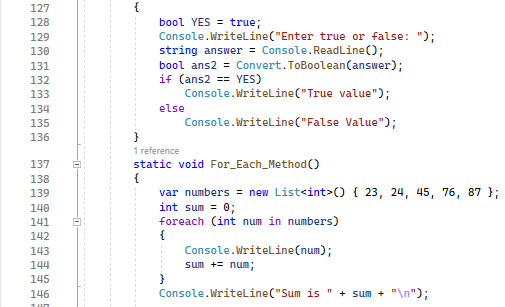
}

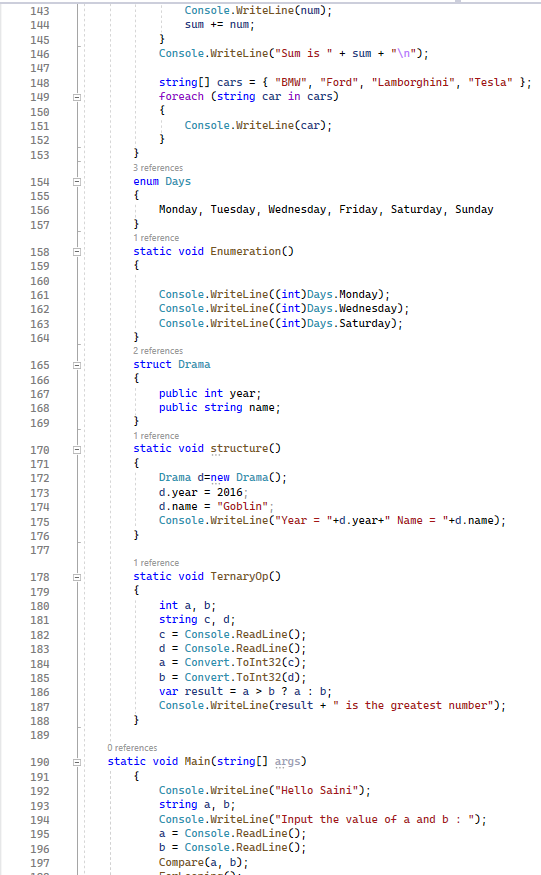
**Program in C#**

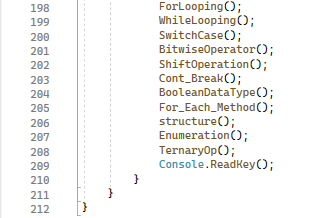






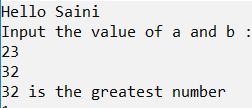




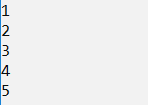


**Output:**

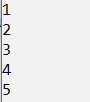
compare(a, b) : use of if/else and > operator



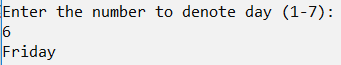
ForLooping(): use of For loop to print numbers from 1 to 5



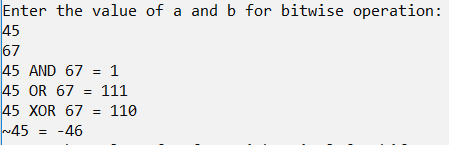
WhileLooping(): use of While loop to print numbers from 1 to 5



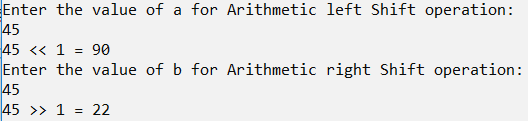
SwitchCase() : use of Switch, case and break



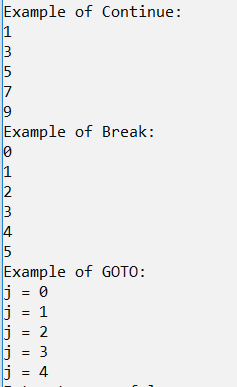
BitwiseOperation(): use of bitwise operator (bitwise OR, AND, XOR and NOT)



ShiftOperation() : use of left and right shift operators



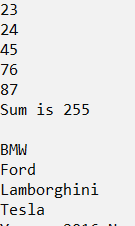
Cont\_Break(): use of jumping statements - continue, break and goto.



BooleanDataType() : use of Boolean data type



For\_each\_Method() : use of For Each Loop and +, = operators



structure(): use of structure instance



Enumeration(): use of enum types



TernaryOp(): use of Conditional operator( Ternary operator)

