3.Decision Making Constructs in C#.Net

Decision constructs show how to control the flow of program execution.

- **Branching Statements**
- **Jumping Statement**
- **Control Flow Change Statement**
- **Conditional**

3.1 The Branching Statements:

a. if

b. if-else

c. if-else if-else

d. nested if

```
if (conditional expression)
{
		[execute if above condition returns true];
}
```

```
if (conditional expression)
{
		[execute if above condition returns true];
}
else{
		[Execute if above condition returns false];
}
```

Demo 3.1: To decide a person is major or minor by taking age from user.

The if-else if-else Statement

This construct is use to work with multiple conditions but it has a draw back in compare with switch statement that its checks all conditions unless and until not satisfying with the condition.

obtain a literal Boolean value.

```
if (conditional expression)
{
            [execute if above condition returns true];
}
else if(conditional expression){
            [Execute if above condition returns true];
}else{
             [Execute if above all condition returns false];
}
```

Demo 3.2: To display name of month by taking number from user.

3.3 The Nested if statement

If within if is use to check conditions upon dependencies and priorities.

obtain a literal Boolean value.

```
if (conditional expression)
{
            [Execute if outer condition returns true];

            if(conditional expression){
            [Execute if nested condition returns true];
            }
            else{
             [Execute if nested condition returns false];
            }
}
else{
             [Execute if outer condition returns false];
}
```

Demo 3.3: To decide eligibility of candidate on basis qualification and professional experience using nested if.

3.4 Jumping Statement

The switch Statement

The other simple selection construct offered by C# is the switch statement. As in other C-based languages, the switch statement allows you to handle program flow based on a predefined set of choices. This construct is more faster than if.

```
switch (variable)
{
case 1: [Expression/Statement];
break;
case 2: [Expression/Statement];
break;
default: [Expression/Statement];
break;
}
```

Demo 3.4: To display name of month by taking number from user.

3.5 Control Flow Statement:

go to-Label statement

go to statement is use to change control flow of program conditionally,using go to we can make a block for recursive call. This block usually nest in if statement To know when it can stop recursion.

```
Label :

if(Conditional Expression)
{
  goto Label;
}else{
            Expression or Statement;
}
```

Demo 3.5: To calculate sum of two numbers recursively by user interest using go to statement.

3.6 Conditional Operators

These operators also returns true/false.

- Short Circuiting operators [&& ||]
- Ternary Operator [Conditional Expression ? true : false]

```
if(Conditional Expression1 && Conditional Expression2)
{
        [Executes if all Expression returns true]
}else{
        [Executes if either Expression returns false]
}
```

```
if(Conditional Expression1 || Conditional Expression2)
{
        [Executes if either Expression returns true]
}else{
        [Executes if all Expression returns false]
}
```

Demo 3.6.1: To decide eligibility of candidate on basis qualification and professional experience using nested if.

Ternary operators [Conditional Expression ? true : false]

Ternary operator is alternative solution for if/else statement but this only useful

When expression or statements are short it use to optimize length of code.

```
Conditional Expression ? Expression/Statement [true] : Expression/Statement [false]
```

Demo 3.6.2: To decide a person is major or minor by taking age from user.

Minutes of Chapter

- ❑The if/else Statement
- ❑The if/else if/else Statement
- ❑The Nested if statement
- ❑The switch Statement [Jumping Statement]
- ❑go to statement [Control flow State]
- ❑Conditional Operators
- ❑Ternary operators [Conditional Expression ? true : false]