TechLift
KICKSTART YOUR CAREER

WEB ENGINEERING

.Net

P@SHA

PAKISTAN PSEB SOFTWARE EXPORT BOARD

Week 2 - Day 12

Recap Project (Time: 1 hr)

The students will work in pairs and complete the following project:

Simple Flying Bird Game in C#

They may take help from this video or this tutorial to complete the project.

Marked Project (Total marks: 20) (Time: 1.5 hrs)

The students will work individually and complete the following project:

Simple Calculator App in C#

Marks will be calculated based on the ratio of correct coding and steps.

Basics

Learning Objectives

By the end of this session, the students will have developed an understanding of:

- Conditional Statements
- Looping

Conditional Statements

- C# provides many decision-making statements that help the flow of the C# program based on certain logical conditions.
- Here, you will learn about if, else if, else, and nested if else statements to control the flow based on the conditions.
- C# includes the following flavors of if statements:
  - if statement
  - else-if statement
  - else statement

C# if Statement

The if statement contains a boolean condition followed by a single or multi-line code block to be executed. At runtime, if a boolean condition evaluates to true, then the code block will be executed, otherwise not.

Syntax:

```
if(condition)
{
    // code block to be executed when if condition evaluates to true
}
```

Example ‘if’ statement

The trainer will demonstrate this code to explain the statement.

```
int i = 10, j = 20;

if (i < j)
{
    Console.WriteLine("i is less than j");
}

if (i > j)
{
    Console.WriteLine("i is greater than j");
}
```

Output:

```
i is less than j
```

Task: Find out the Error

Write this code on your computer and try finding out the error in it.

```
int i = 10, j = 20;

if (i + 1)
{
    Console.WriteLine("i is less than j");
}

if (i + j)
{
    Console.WriteLine("i is greater than j");
}
```

Example: Calling Function as Condition

The trainer will demonstrate this code on their computer and encourage the students to guess its output.

Once completed, click here to check the output.

```
static void Main(string[] args)
{
    int i = 10, j = 20;

    if (isGreater(i, j))
    {
        Console.WriteLine("i is less than j");
    }

    if (isGreater(j, i))
    {
        Console.WriteLine("j is greater than i");
    }
}

static bool isGreater(int i, int j)
{
    return i > j;
}
```

else if Statement

Multiple else if statements can be used after an if statement. It will only be executed when the if condition evaluates to false. So, either if or one of the else if statements can be executed, but not both.

Syntax

```
if(condition1)
{
    // code block to be executed when if condition1 evaluates to true
}
else if(condition2)
{
    // code block to be executed when
    //      condition1 evaluates to flase
    //      condition2 evaluates to true
}
else if(condition3)
{
    // code block to be executed when
    //      condition1 evaluates to flase
    //      condition2 evaluates to false
    //      condition3 evaluates to true
}
```

Example: else if Statements

The trainer will demonstrate this code on their computer and encourage the students to guess its output.

Once completed, click here to check the output.

```
int i = 10, j = 20;

if (i == j)
{
    Console.WriteLine("i is equal to j");
}
else if (i > j)
{
    Console.WriteLine("i is greater than j");
}
else if (i < j)
{
    Console.WriteLine("i is less than j");
}
```

else Statement

- The else statement can come only after if or else if statement and can be used only once in the if-else statements.
- The else statement cannot contain any condition and will be executed when all the previous if and else if conditions evaluate to false.

Example: else Statement

The trainer will demonstrate this code on their computer and encourage the students to guess its output.

Once completed, click here to check the output.

```
int i = 20, j = 20;

if (i > j)
{
    Console.WriteLine("i is greater than j");
}
else if (i < j)
{
    Console.WriteLine("i is less than j");
}
else
{
    Console.WriteLine("i is equal to j");
}
```

Nested if Statements

C# supports if else statements inside another if else statements. This are called nested if else statements. The nested if statements make the code more readable.

Syntax:

Syntax

Click here to check the syntax.

```
if(condition1)
{
   if(condition2)
    {
        // code block to be executed when
        //      condition1 and condition2 evaluates to true
    }
    else if(condition3)
    {
        if(condition4)
        {
            // code block to be executed when
            //      only condition1, condition3, and condition4 evaluates to true
        }
        else if(condition5)
        {
            // code block to be executed when
            //      only condition1, condition3, and condition5 evaluates to true
        }
        else
        {
            // code block to be executed when
            //      condition1, and condition3 evaluates to true
            //      condition4 and condition5 evaluates to false
        }
    }
}
```

P@SHA PSEB

Example: Nested if else statements

The trainer will demonstrate this code on their computer and encourage the students to guess its output.

Once completed, click here to check the output

```
int i = 10, j = 20;

if (i != j)
{
    if (i < j)
    {
        Console.WriteLine("i is less than j");
    }
    else if (i > j)
    {
        Console.WriteLine("i is greater than j");
    }
}
else
    Console.WriteLine("i is equal to j");
```

Task 1: Conditional Statements

1. Write a C# Sharp program to accept two integers and check whether they are equal or not.

Test Data :

Input 1st number: 5

Input 2nd number: 5

Expected Output :

5 and 5 are equal

Task 2: Conditional Statements

Write C# Program to find the Largest among Three Variables using **Nested if**

C# Switch Statement

Use the switch statement to select one of many code blocks to be executed.

Syntax

```
switch(expression)
{
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
    break;
}
```

This is how it works:

- The switch expression is evaluated once
- The value of the expression is compared with the values of each case
- If there is a match, the associated block of code is executed

Example:

The trainer will demonstrate this code on their computer and encourage the students to guess its output.

Once completed, click here to check the output.

```
int x = 10;

switch (x)
{
    case 5:
        Console.WriteLine("Value of x is 5");
        break;
    case 10:
        Console.WriteLine("Value of x is 10");
        break;
    case 15:
        Console.WriteLine("Value of x is 15");
        break;
    default:
        Console.WriteLine("Unknown value");
        break;
}
```

Task 1: Switch Statement

Write a switch statement for the following output:

The weekday number to calculate the weekday name.

Click here to tally your statement.

Task 2: Switch Statement

Write C# program to print number of days in a month using switch case.

Click here to check the statement and the output.

C# Loops

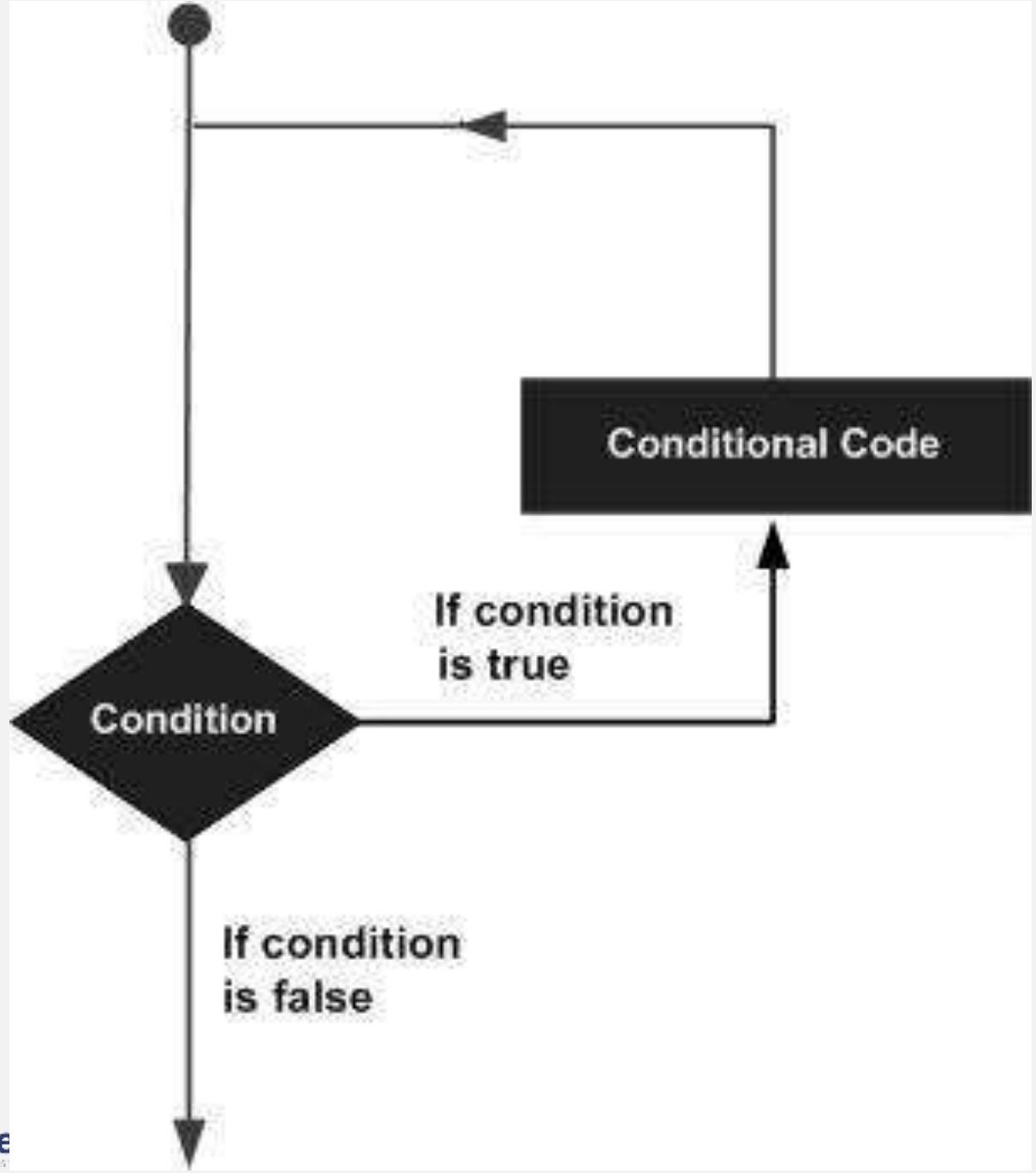
Loop

- There may be a situation, when you need to execute a block of code several number of times. In general, the statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.
- Programming languages provide various control structures that allow for more complicated execution paths.
- A loop statement allows us to execute a statement or a group of statements multiple times and following is the general from of a loop statement in most of the programming languages −

Types of Loops

C# provides following types of loop to handle looping requirements.

Click the **red links** given in the table to check their relevant detail.



Sr.No.	Loop Type & Description
1	**while loop** It repeats a statement or a group of statements while a given condition is true. It tests the condition before executing the loop body.
2	**for loop** It executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
3	**do...while loop** It is similar to a while statement, except that it tests the condition at the end of the loop body
4	**nested loops** You can use one or more loop inside any another while, for or do..while loop.

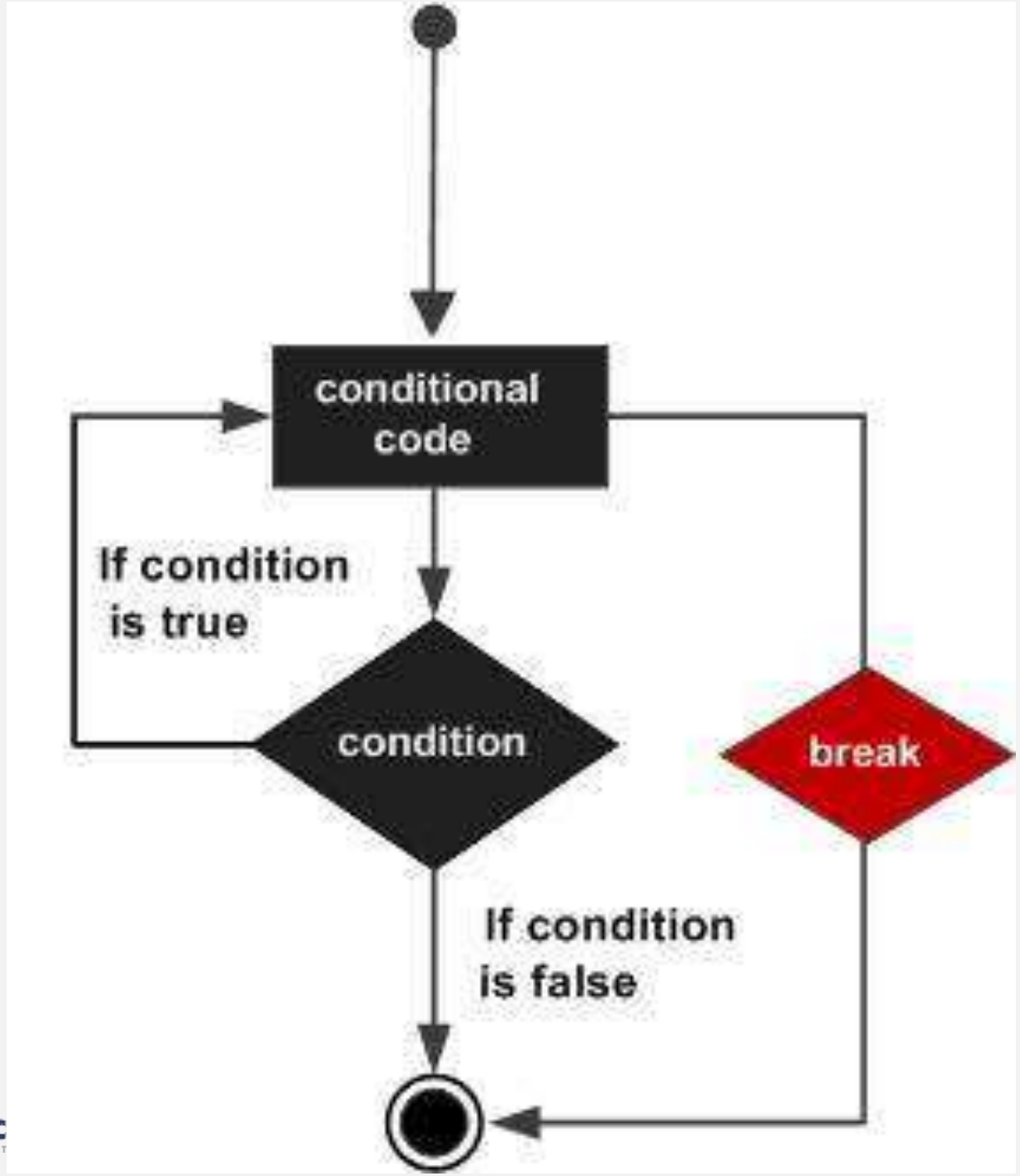
Loop Control Statements

- Loop control statements change execution from its normal sequence.
- When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

Loop Control Statements

C# provides the following control statements.

Click the **red links** given in the table to check their relevant detail.



Sr.No.	Control Statement & Description
1	**break statement** Terminates the loop or switch statement and transfers execution to the statement immediately following the loop or switch.
2	**continue statement** Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

Task 1: Loop

Write C# program to print multiplication table of a given number.

Click here to check the statement and the output.

Task 2: Loop

Write C# program to find sum of even numbers between 1 to n

Click here to check the statement and the output.

Build C# Game Programming

Develop a really simple game named 'Running T Rex'.

After developing this game, click here to check the code and the output.

You may also take help from this link.

https://www.simplilearn.com/tutorials/c-sharp-tutorial/c-sharp-game-programming

Homework

1- Make a money making machine with C# data types and variables.

2- Write a C# Sharp program to check whether a given number is even or odd.

Test Data : 15

Expected Output :

15 is an odd integer

Learning Objectives

By the end of this session, the students have practised

- Conditional Statements
- Looping

Conclusion & Q/A

See you tomorrow!