Pre- Release Programming Section:

Paper 2 of Computer Science will have a section of 20 marks which must be solved by using any procedural programming language. We will use VB.NET console programming to solve all the programming problems. In the upcoming section of programming students will learn about VB.NET Console programming.

You are advised to go through the basics of VB.NET and all the examples but in exam you will have to solve the questions derived from Tasks of pre-release material so you should manly focus on preparing the tasks and relevant question to these tasks.

All the tasks along with solution, description and possible questions are at the end of this chapter.

Introduction to VB.NET

Note: While attempting the programming questions in CIE exam, you have to mention the programming language you will use to solve the paper i-e VB.NET

What is VB.NET?

VB is also known as **VISUAL BASIC**. It is a programming language developed by Microsoft, first released in 2002.

It is used to create Windows Applications, Windows RT Applications, Windows Phone Applications, Windows Console Applications and much more.

Because it uses the Microsoft.NET framework for lots of functions thus it is called as VB.NET.

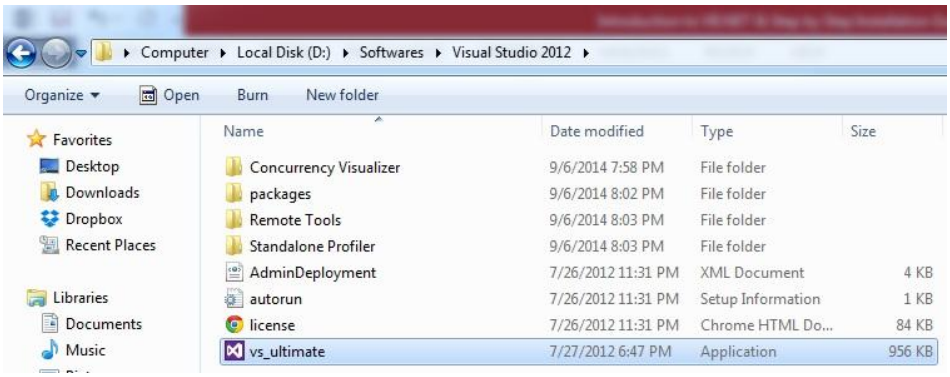
Advantages of VB.NET

- It is quick and easy to develop software and applications.
- It has a very simple and easy to use syntax.
- It is extremely supported by Microsoft & Its community.

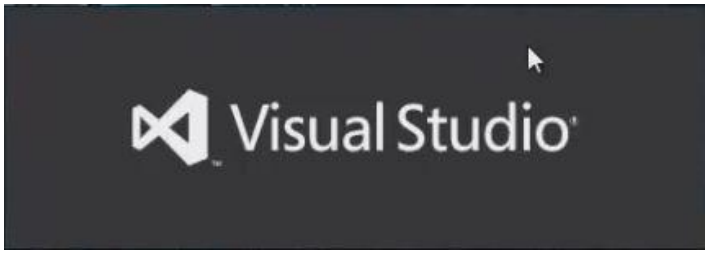
Step by Step Installation Guide for VB.NET

Step 1

Copy and paste the folder of **Visual Studio 2012** in your system. Open the folder and double click on **vs_ultimate** as shown in figure below:



After double clicking on **vs_ultimate**, the installation setup will be started as shown below:



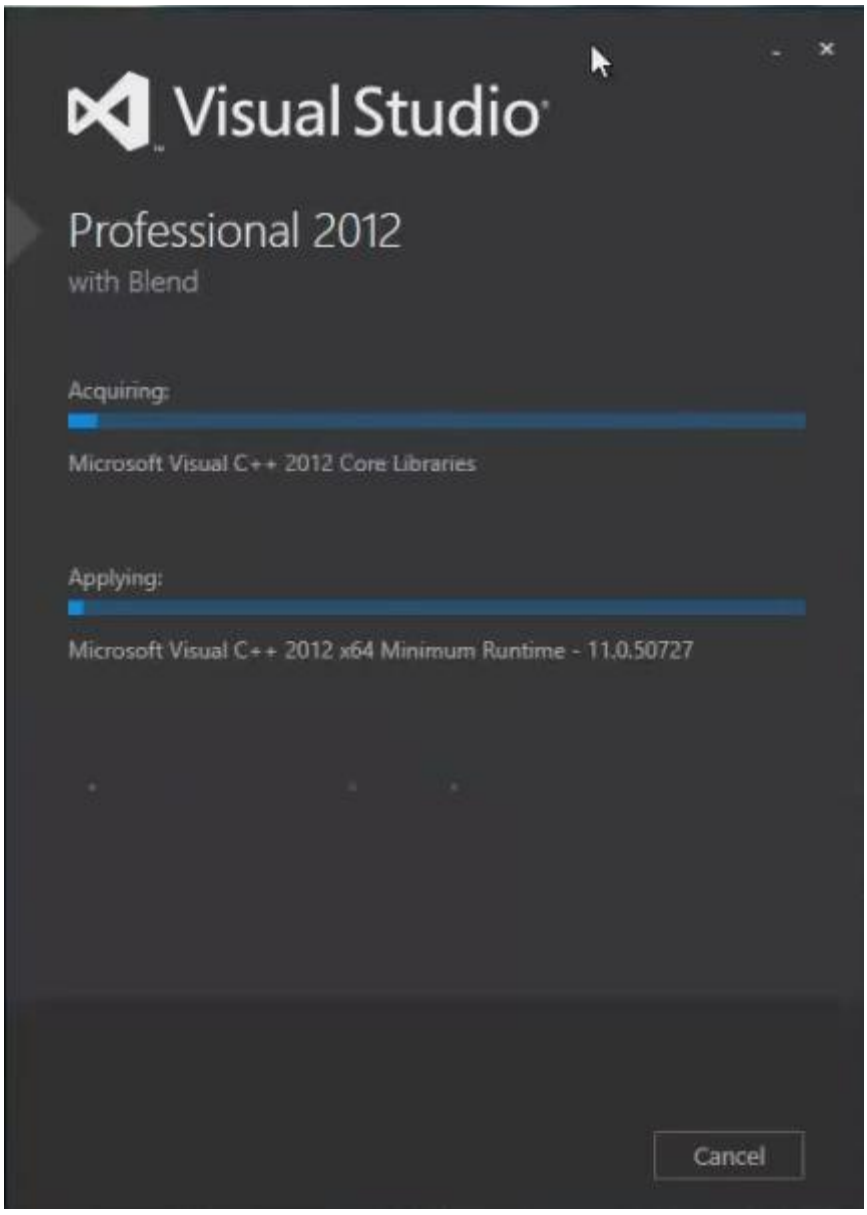
Step 2

Choose the path for installation of **Visual Studio 2012**. It is recommended that select any drive to install **Visual Studio 2012** apart from the main drive where the Operating System (Windows) is installed as shown in figure below:

Step 3

After choosing the **path** click on **Next button**, it will show the **features** you want to install. Select **all features** and click on **INSTALL button** as shown below:

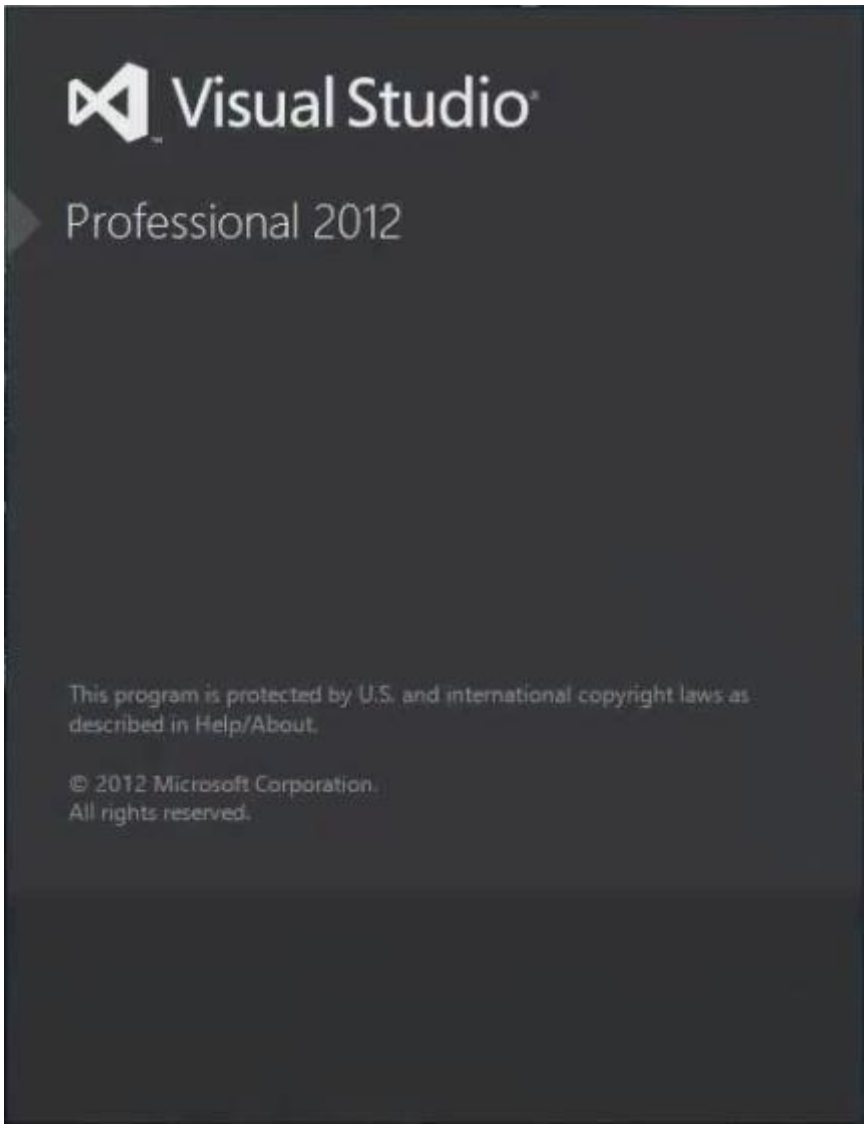
The **installation process** will start as shown in figure below:



Step 4

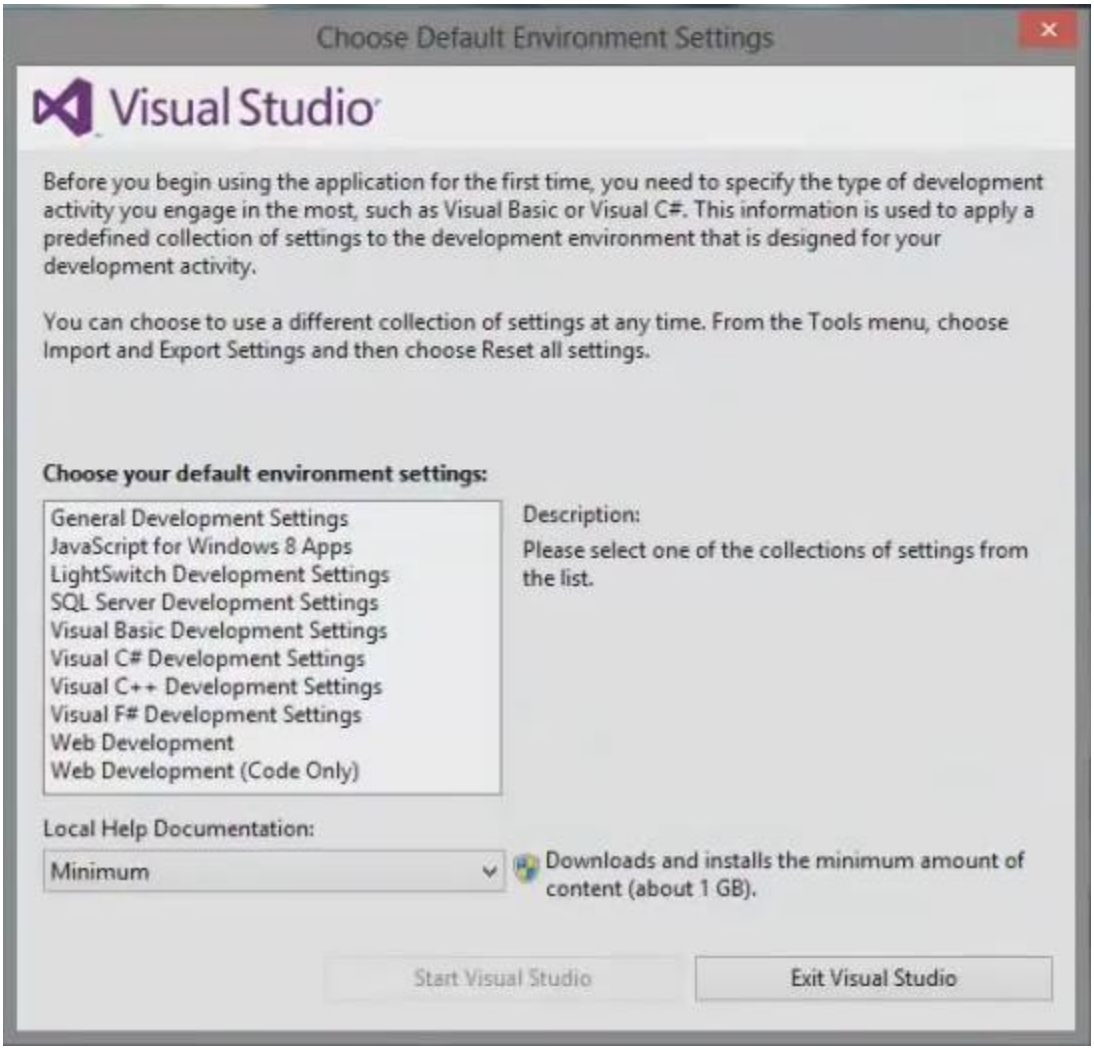
Once the installation is complete a window will appear as shown below. Click on **LAUNCH button** to execute the **Visual Studio 2012** for the very first time.

Once you click on the **LAUNCH button**, a window will appear as shown below:



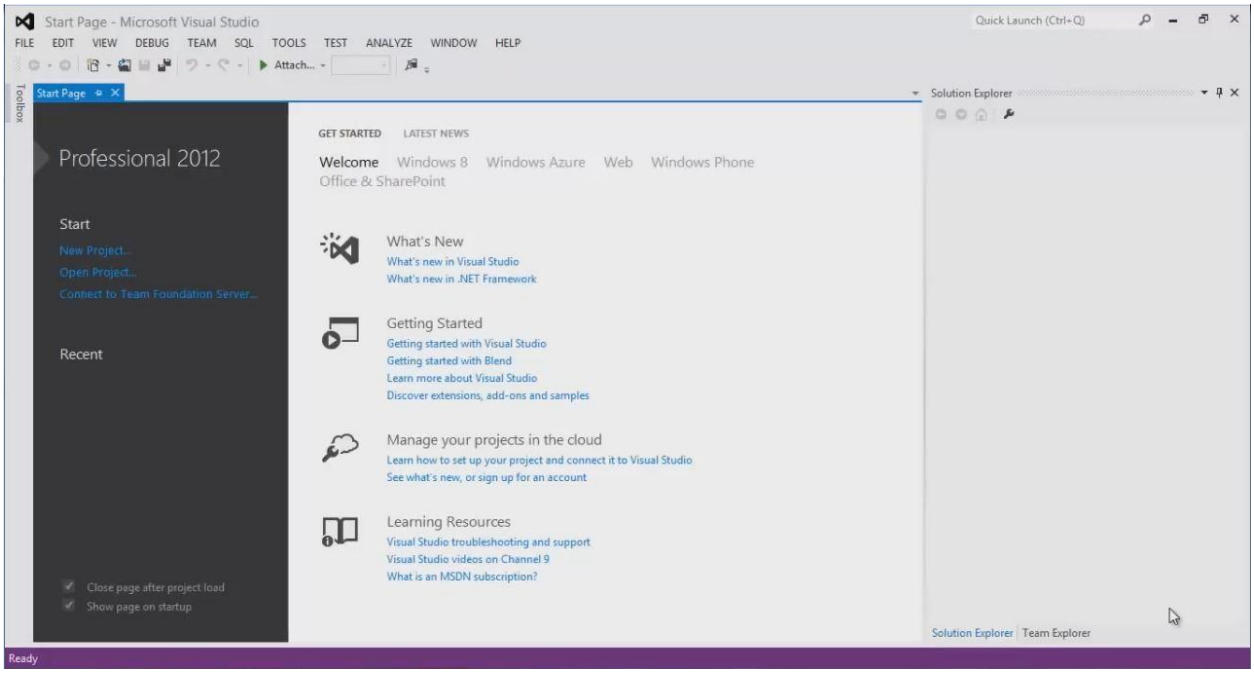
Step 5

Now you have to **Choose Default Environment Settings**, click on the very first option **"General Development Settings"**, select **"Minimum"** in Local Help Documentation and click on **Start Visual Studio** button.



It will load user settings as shown below:

After loading all user settings, finally **the Visual Studio 2012 is launched** as shown below:



If you are able to see the above window after the installation of **Visual Studio 2012** then it means you have installed the software properly and now you can **start programming** by using **Visual Studio 2012**.

To Get a Start with Visual Studio 2012/2013 & Write First Program

Getting Start with Visual Studio 2012/2013

Locate the Visual Studio 2012/2013 icon and double click on it as shown in figure 2.1:

Figure 2.1

It will take few seconds while loading the software. Once it is loaded then click on “File” menu, click on “New” and finally click on “Project” to start a new project to program as shown in figure 2.2:

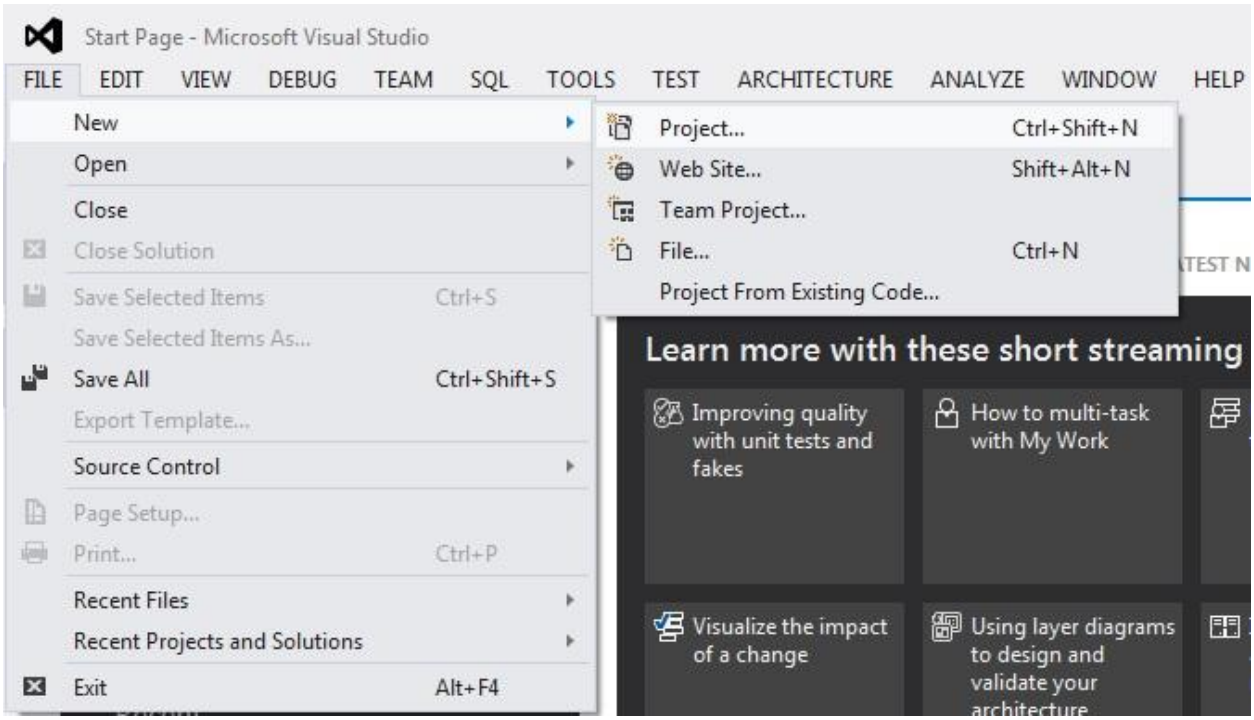


Figure 2.2

Visual Studio allows us to create and design programs by using many languages such as VB.NET, C# etc. We will use VB.NET as a programming language to design Console Applications throughout this course as a requirement of CIE.

Whenever we start a new project, it first asks that what language we will use to program for this project and you have to specify it as shown in figure 2.3:

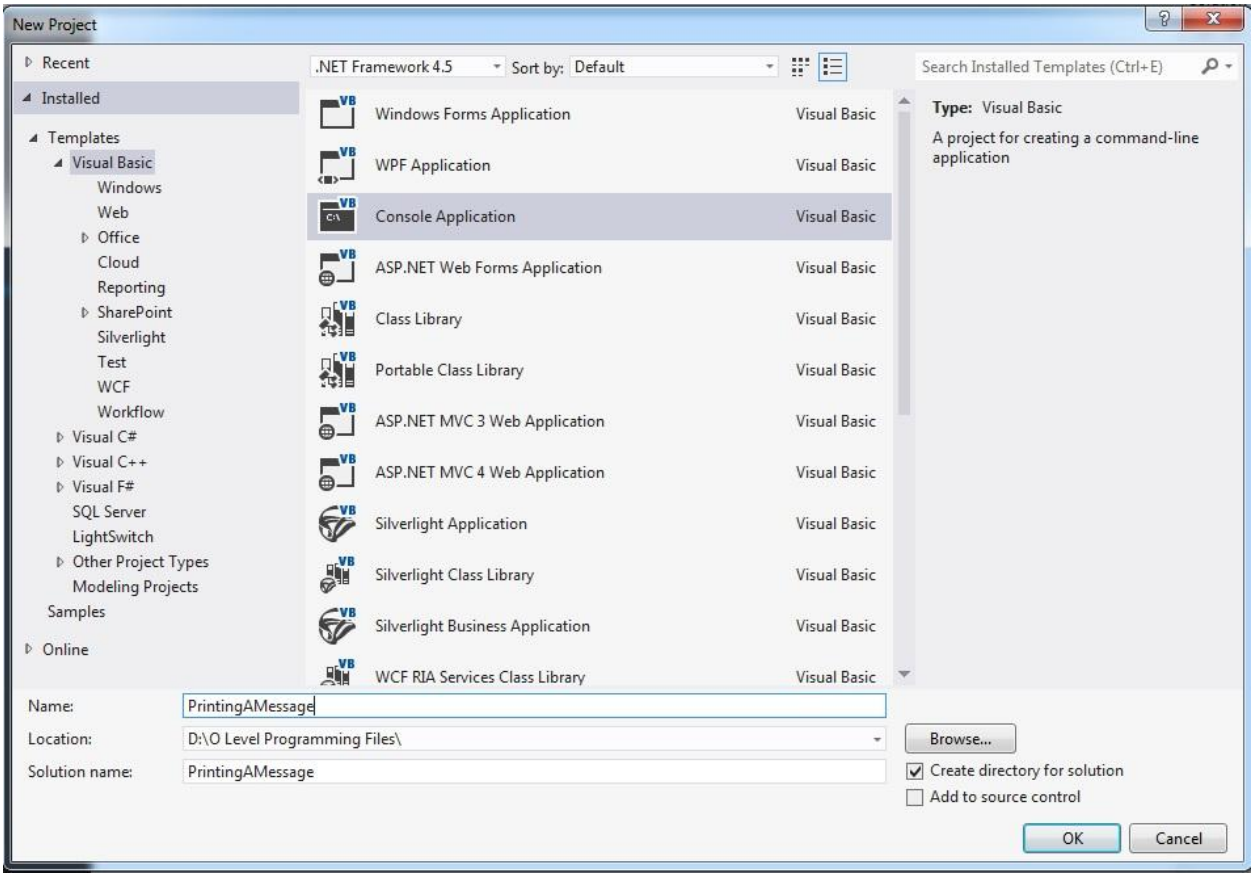


Figure 2.3

Click on **"Installed"**, Click on **"Templates"**, Click on **"Visual Basic"**, Select **"Console Application"**, write down the name of project as **"PrintingAMessage"**, specify the **"Location"** of project to store and finally click on the **Ok button** to create the project.

Note: Every time you will follow the above mentioned procedure to create a new project.

It will take few seconds to process and create a new project. Once a project is created, a program editing window will appear as shown in Figure 2.4:

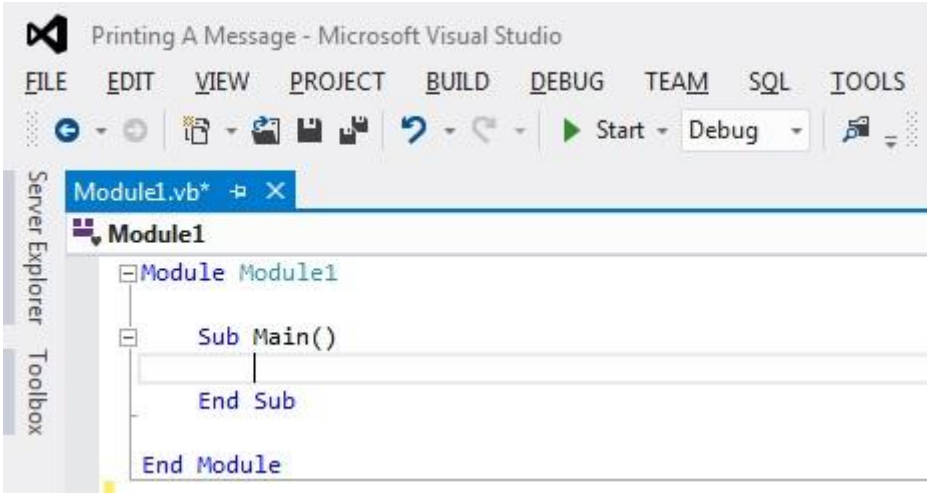


Figure 2.4

Visual Studio is a smart programming tool which generates few lines of code automatically as shown in Figure 2.4.

Whenever you will create a project for designing a Console Application, it will automatically generate these 4 lines of code.

Note: You have to remember and write down these 4 lines of code in every program while attempting the programming questions in exams.

Now we will have a look on these lines that what is the actual meaning of each line of code.

- **Module:** Code in Visual Basic is stored in the form of modules. The keyword **Module** tells that a Module is starting from this point and **Module1** is the name of that Module.

- **Sub Main():** Every Visual Basic application must contain a procedure called Main. This procedure serves as the starting point and overall control for your application. There are four types of **Main()** procedure. We will use its first type **Sub Main().**

- **End Sub:** This keyword is used to tell that it is the point where **Sub Main()** ends.

- **End Modue:** It tells that **Module Module1** is ending at this point.

Note: You should never change or edit the name of Module or Sub Main() otherwise it will cause errors and your program will not work. Visual Basic is a Case Sensitive Programming Language.

Note: In Visual Basic, everything with () after its name is a function/procedure/subroutine.

A function is a block of code which performs a particular task. In this course we will use the built-in functions of Visual Basic to write the programs.

Your First Program

Now it is the time to start with our first program of printing a message on screen. We use to print a message in pseudocode using the following command:

WRITE "This is my first message"
Or
OUTPUT "This is my first message"
Or
PRINT "This is my first message"

While in Visual Basic, we use the following command to print a message on screen.

Console.WriteLine("This is my first message")

Here **WriteLine()** is a function/subroutine/procedure used to print a message on screen. **Console** means that we are using **WriteLine()** with a **Console Application**.

Console

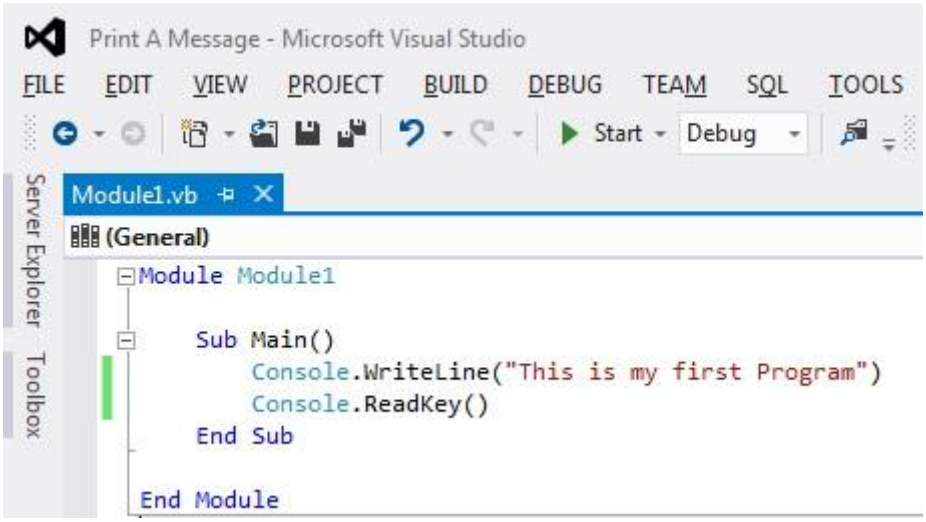


Figure 2.5

Type the code in the code editor of **Visual Studio 2012** as shown in Figure 2.5 and click on the **Start Button** located below the **DEBUG menu** and it will debug your program and will show if there is any error. If the program contains no error then it will execute as shown in Figure 2.6.

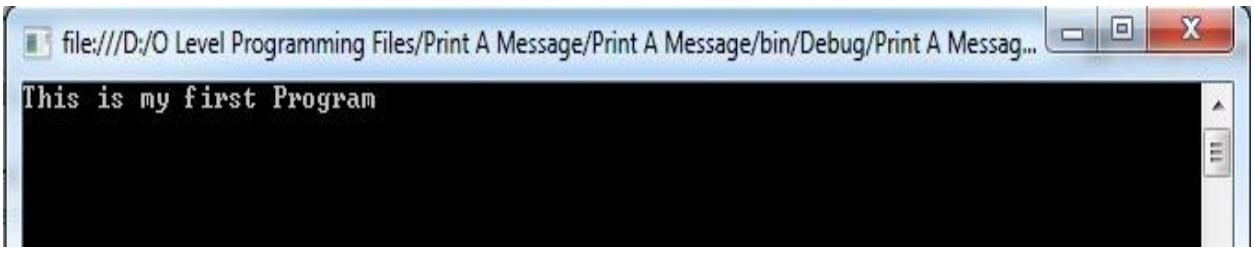


Figure 2.6

Printing Message on Multiple Lines (Method 1)

You can display messages on multiple lines as shown in code in Figure 2.7.

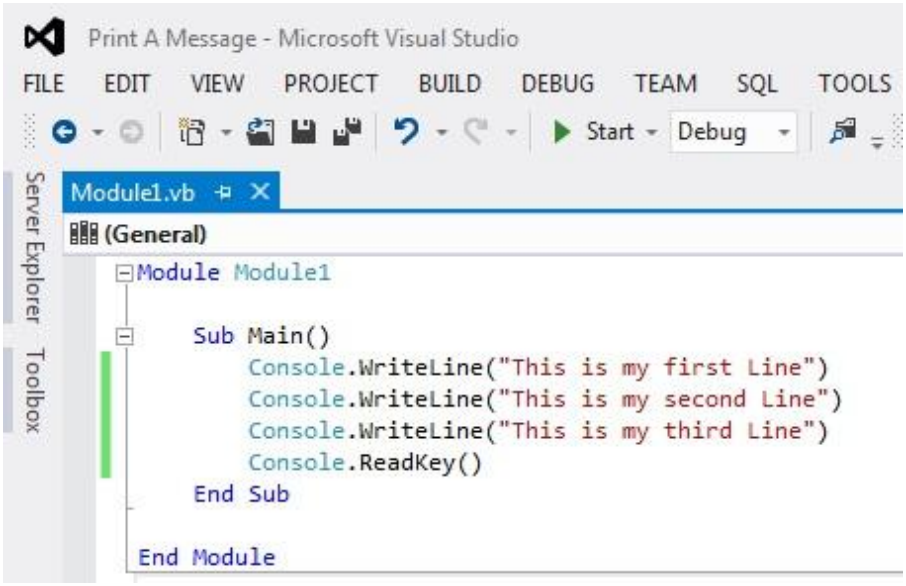


Figure 2.7

The output of above code is shown in Figure 2.8.

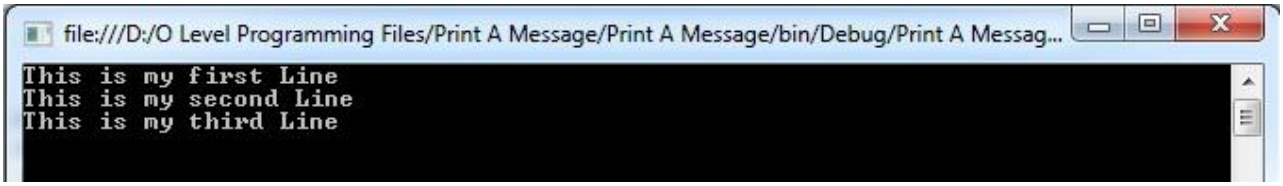


Figure 2.8

Declaring and Initializing a Variable

A variable is declared when it is assigned a name and data type. The name of variable should be relevant while data type may be Integer, Single, String, Char or Boolean.

Integer means variable will store whole numbers only.

Single means variable will store decimal (point form) numbers only.

String means variable will store anything typed from keyboard e.g number, alphabets, symbols etc.

Char means variable will store only one character.

Boolean means variable will store only true or false value.

Program1: Write down a program in VB.NET in which two variables should be declared & defined (values should be assigned) and finally their answer after mathematical operations is shown as output. Type in the code given in each block and observe the change in the code in each block and change in the output for each code block.

```
Module Module1

    Sub Main()

        Console.WriteLine("This program will show the sum of two numbers")

        Dim num1 As Integer
        Dim num2 As Integer
        Dim ans As Integer

        num1 = 2
        num2 = 3

        ans = num1 + num2

        Console.WriteLine(ans)
        Console.ReadKey()

    End Sub

End Module
```

---

1. **What is the output if you type the following code:**

```
Module Module1

    Sub Main()

        Console.WriteLine("This program will show the sum of two numbers")


        num1 = 3.3
        num2 = 2

        ans = num1 * num2

        Console.WriteLine("Answer is: {0}", ans)
        Console.ReadKey()

        Dim num1 As Single
        Dim num2 As Integer
        Dim ans As Single

    End Sub
```

```
End Module
```

2. **What is the output if you type the following code:**

```
Module Module1

    Sub Main()

        Console.WriteLine("This program will show the sum of two numbers")

        Dim num1 As Single
        Dim num2 As Integer
        Dim ans As Single

        num1 = 1
        num2 = 3

        ans = num1 / num2

        Console.WriteLine("Answer is: {0}", ans)
        Console.ReadKey()


    End Sub

End Module
```

3. **What is the output if you change the following lines in above code with the one mentioned below:**

```
        Dim num1 As Single
        Dim num2 As Integer
        Dim ans As Double
```

4. **What is the output if you type the following code:**

```
Module Module1

    Sub Main()

        Console.WriteLine("This program will show the sum of two numbers")

        Dim num1 As Integer
        Dim num2 As Integer
        Dim ans As Integer

        num1 = 2147483647
        num2 = 0

        ans = num1 + num2
```

```
        Console.WriteLine("Answer is: {0}", ans)
        Console.ReadKey()


    End Sub

End Module
```

5. **What is the output if you change the following lines in above code with the one mentioned below:**

```
num1 = 2147483647
num2 = 1

ans = num1 + num2
```

Understanding & Practicing For Loop in VB.NET

Loops are programming structures that are used to repeat a single line or multiple lines of code. There are three types of loops in VB.NET programming:

(1) For Loop or For To Next Loop

(2) While ….. End While Loop

(3) Do ….. Loop Until

In this lab handout, we will discuss about only For Loop.

For loop is used to repeat a single programming instruction or multiple programming instructions for number of times depending upon the initial and final values in the condition.

Format of a For Loop is as shown below:

For <Variable Name> As <Data Type> = <initial value> To <final value> Step <increment or decrement>

……..Coding Here……

Next

Example 1: Print the message “Hello World” 15 times on console screen using For Loop.Solution:

```
Module Module1

    Sub Main()

        For count As Integer = 1 To 15 Step +1

            Console.WriteLine(“Hello World”)

        Next

        Console.ReadKey()
```

```
    End Sub
End Module
```

Example 2: Mr. Jhon wants to calculate the average marks of his class in subject of Urdu. Help him by making a program using VB.NET language. Total students in class are 12. Total marks of test are 20.

```
Module Module1

    Sub Main()

        Dim marks As Integer
        Dim tmarks As Integer
        Dim avgmarks As Integer

        tmarks = 0

        For count As Integer = 1 To 12 Step 1

            Console.WriteLine("Enter Marks of Students:")
            marks = Console.ReadLine()
            tmarks = tmarks + marks

        Next

        avgmarks = tmarks / 12

        Console.WriteLine("Average Marks are: {0}", avgmarks)

        Console.ReadKey()

    End Sub

End Module
```

Understanding & Practicing While EndWhile Loop in VB.NET

While EndWhile loop is used when it is not known that how many times repetition is required. It executes a series of statements as long as a given condition is True.

Sytanx of While…..EndWhile loop

While <Condition>

[Programming Statements]

End While

Here, programming statements may be a single line statement or a block of statements on multiple lines. The condition may be any expression, and true is logical true. The loop iterates while the condition is true.
When the condition becomes false then loop is stopped.

Example 1: Write down VB.Net code using While Loop which will take numbers as input and add them. Loop will stop when any negative number is entered and total sum of previous entered numbers will be displayed.

```
Module Module1

    Sub Main()
        Dim a As Integer
        Dim total As Integer
        total = 0
        a = Console.ReadLine()
        While a > -1
            total = total + a
            a = Console.ReadLine
        End While
        Console.WriteLine("Total is: {0}", total)
        Console.ReadKey()
    End Sub

End Module
```

Example 2: Write down VB.NET code which will take 15 numbers as input and calculates their average using while…..end while loop.

```
Module Module1

    Sub Main()
        Dim Count As Integer
        Dim a As Integer
        Dim total As Integer
        total = 0
        Count = 1

        While Count <= 15
            a = Console.ReadLine()
            total = total + a
            Count = Count + 1
        End While
        Console.WriteLine("Total is: {0}", total)
        Console.ReadKey()
    End Sub

End Module
```

Remove Count = Count + 1 from above code and then execute it and observe the output of program.

Understanding & Practicing Do Loop Until in VB.NET

It repeats the enclosed block of statements when the loop condition is **False** and stops when the condition becomes **True**. It could be terminated at any time with the Exit Do statement.

In this loop, condition of the loop is checked at the end of the loop. Which means if a condition is true then still loop will execute at least once?

Format/Syntax of Do…. Loop Until

Do

……. Programming Instructions…….

Loop Until <Condition>

Example 1: Write down the VB.NET code which prints Integer numbers from 10 to 19 using Do Loop Until.

```
Module Module1
   Sub Main()
      Dim a As Integer = 10
      Do
          Console.WriteLine("value of a: {0}", a)
          a = a + 1
      Loop Until (a = 20)
      Console.ReadLine()
   End Sub
End Module
```

Example 1: Write down the VB.NET code using Do Loop Until which will take anything as input from the user and displays it on screen. **Program will stop** when A or a or B or b is entered by user.

```
Module Module1

    Sub Main()

        Dim character As Char

        Do

            Console.WriteLine("Enter a character and It will be displayed on Screen")
            Console.WriteLine("Enter A or a or B or b to stop")
            character = Console.ReadLine()
            Console.WriteLine("You entered {0}", character)
            Console.WriteLine()

        Loop Until ((character = "A") Or (character = "a") Or (character = "B") Or
(character = "b"))

        Console.ReadKey()

    End Sub

End Module
```

Understanding & Implementing the Selection Statements in VB.NET

Selection statements provides the means of choosing between two or more execution paths in a program on the basis of some conditions. There are two types of selection statements:

(1) IF Statements

(2) Select Case Statements

IF Statements have three types:

(i) If … Then Statement

(ii) If … Then … Else Statement

(iii) Nested If Statements

If … Then Statement

It is the simplest form of IF statements where the **If** keyword is followed by the condition. If the condition evaluates to true, then the block of code inside the If statement will be executed. If condition evaluates to false, then the first set of code after the end of the If statement (after the closing End If) will be executed.

Format of If … Then Statement

If (Condition) Then

….. Programming Code …..

End If

Example 1: Write down a VB.Net code which will take a number as input from user and tells that it is positive.

```
Module Module1

    Sub Main()
        Dim a As Integer
        Console.WriteLine("Enter a number")
        a = Console.ReadLine()
        If (a > -1) Then
            Console.WriteLine("Number is Positive")
        End If
        Console.WriteLine("value of number is : {0}", a)
        Console.ReadLine()
    End Sub

End Module
```

If … Then… Else Statement

If statement can be followed by an optional **Else** statement. If the condition evaluates to **true**, then the if block of code will be executed, otherwise else block of code will be executed.

Format of If … Then … Else Statement

If (Condition) Then

….. Programming Code …..

Else

….. Programming Code ….

End If

Example 2: Write down a VB.Net code which will take a number as input from user and tells that it is positive or not.

```
Module Module1

    Sub Main()
        Dim a As Integer
        Console.WriteLine("Enter a number")
        a = Console.ReadLine()
        If (a > -1) Then
            Console.WriteLine("Number is Positive")
        Else
            Console.WriteLine("Number is Negative")
        End If
        Console.WriteLine("value of number is : {0}", a)
        Console.ReadLine()
    End Sub

End Module
```

Nested If Statements

It is possible in VB.NET to nest If-Then-Else statements, which means you can use one If or ElseIf statement inside another If ElseIf statement.

Format of Nested If Statements

If (Condition) Then

….. Programming Code …..

ElseIf (Condition) Then

….. Programming Code ….

Else

….. Programming Code ….

End If

Example 3: Write down a VB.Net code which will take a number as input from user and tells that it is positive, negative or zero.

```
Module Module1
```

```
    Sub Main()
        Dim a As Integer
        Console.WriteLine("Enter a number")
        a = Console.ReadLine()
        If (a > 0) Then
            Console.WriteLine("Number is Positive")
        ElseIf (a = 0) Then
            Console.WriteLine("Number is Zero")
        Else
            Console.WriteLine("Number is Negative")
        End If
        Console.WriteLine("value of number is : {0}", a)
        Console.ReadLine()
    End Sub

End Module
```

Example 4: Write down VB.NET code that will take number in the range of 1 to 100 only and generates error message if the number is out of this range.

```
Module Module1

    Sub Main()
        Dim a As Integer
        Console.WriteLine("Enter a number")
        a = Console.ReadLine()
        If (a > 0 And a < 101) Then
            Console.WriteLine("Number is in between range of 1 and 100")
        Else
            Console.WriteLine("Number is Not in the range of 1 and 100")
        End If
        Console.WriteLine("value of number is : {0}", a)
        Console.ReadLine()
    End Sub

End Module
```

Example 5: Write down VB.NET code that will take PIN number of East Bank as input and will show the message welcome to East bank. PIN number can be any number but should be 4 digit.

```
Module Module1

    Sub Main()
        Dim a As Integer
        Console.WriteLine("Enter your pin number")
        a = Console.ReadLine()
        If (a >= 1000 And a < 10000) Then
            Console.WriteLine("Welcom Dear Customer")
        Else
            Console.WriteLine("The PIN Number is Invalid")
        End If
        Console.WriteLine("The Entered PIN number is : {0}", a)
        Console.ReadLine()
    End Sub

End Module
```

Understanding the Arrays in VB.NET

An array is a collection of variables of same data types. Basically an array gives us a way to declare many variables in one step and then be able to store and access their values.

An **array *element*** is one value in an array. An **array *index*** is an integer indicating the position of a value in an array.

For example, below an array with 10 elements is shown. One thing remember always that index of an array always starts from 0. So if you want to create an array named as **Num** in VB.NET then the code would be:

Dim Num(9) As Integer

An Array named as **Num** with 10 values

index	0	1	2	3	4	5	6	7	8	9
value	12	49	-2	26	5	17	-6	84	72	3

Example 1: Write down the VB.NET code that will take 10 Integer values as input and calculates their average using Arrays.

```
Module Module1

    Sub Main()

        Dim Num(4) As Integer
        Dim total As Integer
        Dim avg As Single



        Console.WriteLine("Enter Numbers")
        Num(0) = Console.ReadLine()
        total = total + Num(0)

        Console.WriteLine("Enter Numbers")
        Num(1) = Console.ReadLine()
        total = total + Num(1)

        Console.WriteLine("Enter Numbers")
        Num(2) = Console.ReadLine()
        total = total + Num(2)

        Console.WriteLine("Enter Numbers")
        Num(3) = Console.ReadLine()
        total = total + Num(3)

        Console.WriteLine("Enter Numbers")
        Num(4) = Console.ReadLine()
        total = total + Num(4)

        avg = total / 5

        Console.WriteLine("Average is {0}", avg)
```

```
        Console.ReadKey()

    End Sub

End Module
```

Example 2: Write down the VB.NET code that will take 10 Integer values as input and calculates their average using Arrays & for Loop.

```
Module Module1

    Sub Main()

        Dim Num(9) As Integer
        Dim total As Integer
        Dim avg As Single
        Dim count As Integer

        For count = 0 To 9
            Console.WriteLine("Enter Numbers")
            Num(count) = Console.ReadLine()
            total = total + Num(count)
        Next

        avg = total / 10

        Console.WriteLine("Average is {0}", avg)
        Console.ReadKey()

    End Sub

End Module
```

Cambridge International Examination 2015

Cambridge Ordinary Level

Computer Science (2210)

Paper 2 Problem Solving and Programming

Pre-Release Material

Disclaimer

Task 1

A school keeps records of the weights of each pupil. The weight, in kilograms, of each pupil is recorded on the first day of term. Input and store the weights and names recorded for a class of 30 pupils. You must store the weights in a one-dimensional array and the names in another one-dimensional array. All the weights must be validated on entry and any invalid weights are rejected. You must decide your own validation rules. You may assume that the pupils' names are unique. Output the names and weights of the pupils in the class.

Solution Task 1

```
Module Module1

    Sub Main()

        Dim Name(29) As String
        'Declaring the array Name which will store the names of 30 students.
        Dim fd_weight(29) As Single
        'Declaring the array fd_weight which will store the weights of 30 students at
         First Day (fd).

        Dim count As Integer
        'Declaring the count variable that will be used in for loop to count the number
         of repetitions or iterations.

        Dim weight As Integer
        'Declaring a variable weight to store the value of weight for each students until
         it is validated.Once the weight is validated then it will be stored in fd_weight
         Array.

        For count = 0 To 29
        'For Loop is starting at this point. It will keep on repeating the code when
         the value of count is less than or equal to final value in condition that is 29.
         Once the value of count will exceed 29 then loop condition becomes false and
         loop will stop
       'We are starting count from 0 to 29 because this count would be used to refer the
        index number of arrays and one thing you must remmeber that index number of an
        array always starts from 0.
```

```
            Console.WriteLine("Enter Student Name")
            'Console.WriteLine is a function in VB.NET console programming which is used
             to display a message on Console Screen.

            Name(count) = Console.ReadLine()
            'Console.ReadLine() is a function in VB.NET console programming which is used
             to take input from console screen.
            'Once the input is taken from user then it is stored in Name Array.

            Console.WriteLine("Enter the weight of student in between 15KG and 120KG")

            weight = Console.ReadLine()
            'Taking input value from console screen and storing it in weight variable.

            If (weight >= 120 Or weight <= 15) Then
            'Checking the validity of weight by using the IF statement. IF weight is
             greater than or equals to 120 OR weight is less than or equals to 15 then
             value of weight is invalid and the value of weight is discarded while count
             is decremented by 1 because 1 iteration of loop is wasted taking invalid
             value as input

                Console.WriteLine("The value of weight is invalid.")
                Console.WriteLine("Please Enter a Valid Value again from 15 to 120")

                count = count - 1
                'Decrementing the value of count by 1 for invalid entries.

            Else
           'Once the Condition after IF statement becomes false then code written after
            Else keyword will be executed

                fd_weight(count) = weight

            End If
            'End If tells that an IF statement is ended at this point


        Next
        'Next keyword indicates the end of For loop. It takes the control to the start of
         For loop every.

        For count = 0 To 29

'A blank Console.WriteLine function is also used to give one line space on console screen
            Console.WriteLine()
            Console.WriteLine("Student Name:")
            Console.WriteLine(Name(count))
            'count is starting from 0 to 29 so it will display the values in Name Array
             stored at index 0 to 29

            Console.WriteLine("Student Weight in Kilograms:")
            Console.WriteLine(fd_weight(count))
            Console.WriteLine()
            'Displaying the values of weight for each student by using the count variable
             to refer the index of fd_weight Array.
        Next
        'For loop is ending here.
```

```
        Console.ReadKey()
        'Console.ReadKey will hold the output on console screen until any key is pressed.
    End Sub

End Module
```

Task 2

The weight, in kilograms, of each pupil is recorded again on the last day of term. Calculate and store the difference in weight for each pupil.

Solution Task 2

```
Dim Ld_weight(29) As Single
        'Declare an array Ld_weight (last day weight) to store the weight of students at
         last day of term.
        Dim diff_weight(29) As Single
        'Declare an array diff_weight (difference in weight) to store the difference in
         weight of students.

        For count = 0 To 29
       'For loop to take weight values on last day. It will repeate for 30 times
        starting from 0 and ending at 29.

            Console.WriteLine("Enter The Weight of Student in Between 15Kg and 120Kg for
Last Day of Term")
            Ld_weight(count) = Console.ReadLine()
            'Taking input from user and storing it in Ld-weight array by using the count
             variable to refer the index of array.

            diff_weight(count) = fd_weight(count) - Ld_weight(count)
            'Calculating the difference between weight at first day and weight at last
             day and storing the answer in diff_weight Array.

        Next
        'For Loop is ending here at Next keyword.
```

Task 3

For those pupils who have a difference in weight of more than 2.5 kilograms, output with a suitable message, the pupil's name, the difference in weight and whether this is a rise or a fall.

Solution Task 3

```
           'IF condition to check if the weight is increased. The weight is increased
            when weight value on last day is greater than Weight value on first day.
            If (diff_weight(count) <= -2.5) Then

                'Console.WriteLine() function is used to give one line space.
                Console.WriteLine()

                'Displaying the student name
                Console.WriteLine("Student Name:{0}", Name(count))

                'Math.Abs() function is used to remove the negative sign from weight
                 value.
```

```
            diff_weight(count) = Math.Abs(diff_weight(count))

            'Displaying the weight difference
            Console.WriteLine("Weight Difference:{0}", diff_weight(count))

            'Displaying the message for increase in weight
            Console.WriteLine("The weight is Increased")

        ElseIf (diff_weight(count) >= 2.5) Then

            'Console.WriteLine() function is used to give one line space.
            Console.WriteLine()

            'Displaying the student name
            Console.WriteLine("Student Name:{0}", Name(count))

            'Displaying the weight difference
            Console.WriteLine("Weight Difference:{0}", diff_weight(count))

            'Displaying the message for decrease in weight
            Console.WriteLine("The weight is decreased")

        End If
```

Question 1: Declare the arrays to store the student names, weight on first day and last day with appropriate data type.

Answer: Dim Name(29) As String
Dim fd_weight(29) As Single
Dim Ld_weight(29) As Single

Question 2: Write down the programming code/pseudocode or program flowchart to store the names of students and weight in Kgs at first day and finally output the name and weight in kgs and average weight. Do not include validation checks for weight.

```
Module Module1

    Sub Main()

        Dim Name(29) As String
        Dim fd_weight(29) As Single
        Dim count As Integer
        Dim total_weight As Single
        Dim average As Integer


        For count = 0 To 29

            Console.WriteLine("Enter Student Name")
            Name(count) = Console.ReadLine()
            Console.WriteLine("Enter the weight of student ranging from 15KG to 120KG")
            fd_weight(count) = Console.ReadLine()
```

```
            total_weight = total_weight + fd_weight(count)

        Next

        For count = 0 To 29


        Console.WriteLine("Student Name:{0}",Name(count) )

        Console.WriteLine("Student Weight in Kilograms:{0}",fd_weight(count))


        Next

        average = total_weight/30

        Console.WriteLine("Average of Weight of Whole Class is:{0}",average )

        Console.ReadKey()
    End Sub

End Module
```

Question 3: Write down the programming code/pseudocode or program flowchart to store the names of students and weight in Kgs at first day and finally output the name and weight in kgs. Include appropriate validation checks/rules.

```
Module Module1

    Sub Main()

        Dim Name(29) As String
        Dim fd_weight(29) As Single
        Dim count As Integer
        Dim weight As Single

        For count = 0 To 29

            Console.WriteLine("Enter Student Name")
            Name(count) = Console.ReadLine()
            Console.WriteLine("Enter the weight of student ranging from 15KG to 120KG")
            weight = Console.ReadLine()

            If (weight >= 120 Or weight <= 15) Then

                Console.WriteLine("The value of weight is invalid.")
                Console.WriteLine("Please Enter a Valid Value again from 15 to 120")

                count = count - 1

            Else

                fd_weight(count) = weight

            End If
```

```
        Next

        For count = 0 To 29

            Console.WriteLine()
            Console.WriteLine("Student Name:")
            Console.WriteLine(Name(count))

            Console.WriteLine("Student Weight in Kilograms:")
            Console.WriteLine(fd_weight(count))
            Console.WriteLine()

        Next

        Console.ReadKey()
    End Sub

End Module
```

Question 4: Write down programming code/pseudocode or program flowchart output the names and weight of student in grams recorded at first day of term.

```
Dim count As Integer
For count = 0 To 29

            Console.WriteLine()
            Console.WriteLine("Student Name:")
            Console.WriteLine(Name(count))

            fd_weight(count) = fd_weight(count) * 1000
            Console.WriteLine("Student Weight in grams:")
            Console.WriteLine(fd_weight(count))
            Console.WriteLine()

Next
```

Question 5: Write down the programming code/pseudocode or program flowchart to store the weight of students in Kgs at last day of term. Do not include validation checks for weight.

```
        Dim Ld_weight(29) As Single
        Dim count As Integer

        For count = 0 To 29

            Console.WriteLine()
            Console.WriteLine("Student Name:{0}", Name(count))
            Console.WriteLine("Enter The Weight of Student ranging from 15Kg to 120Kg for
                               Last Day of Term")
            Ld_weight(count) = Console.ReadLine()


        Next
```

Question 6: Write down the programming code/pseudocode or program flowchart to store the weight of students in Kgs at last day of term. Include appropriate validation checks for weight.

```
Dim Ld_weight(29) As Single
Dim weight As Single
Dim count As Integer

For count = 0 To 29

    Console.WriteLine()
    Console.WriteLine("Student Name:{0}", Name(count))
    Console.WriteLine("Enter The Weight of Student ranging from 15Kg to 120Kg for
                       Last Day of Term")
    weight = Console.ReadLine()

    If (weight => 120 Or weight <= 15) Then

        Console.WriteLine("The value of weight is invalid.")
        Console.WriteLine("Please Enter a Valid Value again from 15 to 120")

        count = count - 1

    Else

    Ld_weight(count) = weight

    End If

Next
```

Question 7: Explain how you will select the student with maximum weight recorded at first day. You may include pseudocode or programming statements to help illustrate your explanation.

Answer:

Description

- Declare a variable called max_weight to store maximum value of weight and assign 15 to it. Also declare mw_name to store the name of student with maximum weight.
- Use loop with 30 iterations to check the weight of each student against maximum weight.
- If student weight at first day is greater than maximum recorded weight of students then replace the value of maximum weight by the weight of student and store the student name in mw_name.
- Output the student name having maximum weight.

Programming Code:

```
Dim max_weight As Single
Dim mw_name As String
Dim count As Integer

max_weight = 15
```

```
For count = 0 to 29
If fd_weight(count) > max_weight
  Then
   max_weight = fd_weight(count)
   mw_name = Name(count)
End If
Next
Console.WriteLine("The Student Name:{0}", mw_name)
Console.WriteLine("The Student Maximum Weight: {0}", max_weight)
```

Question 8: Explain how you will select the student with maximum weight recorded at last day. You may include pseudocode or programming statements to help illustrate your explanation.

Answer:

Description

- Declare a variable called max_weight to store maximum value of weight and assign 15 to it. Also declare mw_name to store the name of student with maximum weight.
- Use loop with 30 iterations to check the weight of each student against maximum weight.
- If student weight at last day is greater than maximum recorded weight of students then replace the value of maximum weight by the weight of student and store the student name in mw_name.
- Output the student name having maximum weight.

Programming Code:

```
Dim max_weight As Single
Dim mw_name As String
Dim count As Integer
max_weight = 15

For count = 0 to 29
If Ld_weight(count) > max_weight
  Then
   max_weight = Ld_weight(count)
   mw_name = Name(count)
End If
Next
Console.WriteLine("The Student Name:{0}", mw_name)
Console.WriteLine("The Student Maximum Weight: {0}", max_weight)
```

Question 9: Explain how you will select the student with minimum weight recorded at first day. You may include pseudocode or programming statements to help illustrate your explanation.

Answer:

Description

- Declare a variable called min_weight to store minimum value of weight and assign 120 to it. Also declare minweight_name to store the name of student with minimum weight.
- Use loop with 30 iterations to check the weight of each student against minimum weight.
- If student weight at first day is less than minimum recorded weight of students then replace the value of minimum weight by the weight of student and store the student name in minweight_name.
- Output the minimum weight value with respective student name.

Programming Code:

```
Dim min_weight As Single
Dim minweight_name As String
Dim count As Integer
min_weight = 120

For count = 0 to 29
If fd_weight(count) < min_weight
  Then
   min_weight = fd_weight(count)
   minweight_name = Name(count)
 End If
Next
Console.WriteLine("The Student Name:{0}", minweight_name)
Console.WriteLine("The Minimum Weight: {0}", min_weight)
```

Question 10: Explain how you will select the student with minimum weight recorded at last day. You may include pseudocode or programming statements to help illustrate your explanation.

Answer:

Description

- Declare a variable called min_weight to store minimum value of weight and assign 120 to it. Also declare minweight_name to store the name of student with minimum weight.
- Use loop with 30 iterations to check the weight of each student against minimum weight.
- If student weight at last day is less than minimum recorded weight of students then replace the value of minimum weight by the weight of student and store the student name in minweight_name.
- Output the minimum weight value with respective student name.

Programming Code:

```
Dim min_weight As Single
Dim minweight_name As String
```

```
min_weight = 120

For count = 0 to 29
If Ld_weight(count) < min_weight
  Then
   min_weight = Ld_weight(count)
   minweight_name = Name(count)
 End If
Next
Console.WriteLine("The Student Name:{0}", minweight_name)
Console.WriteLine("The Minimum Weight: {0}", min_weight)
```

Question 11: Write programming code/pseudocode or program flowchart to calculate the difference in between the weight of students recorded at first day and last day. Also show the number of students having a rise in their weight, decline in weight and no change in weight.

```
Dim count As Integer
Dim rise_counter As Integer
Dim fall_counter As Integer
Dim nochange_counter As Integer
Dim diff_weight As Single

rise_counter = 0
fall_counter = 0
nochange_counter = 0

For count = 0 To 29
diff_weight(count) = fd_weight(count) - Ld_weight(count)

If (diff_weight(count) <= -2.5) Then

     rise_counter = rise_counter + 1

ElseIf (diff_weight(count) >= 2.5) Then

     fall_counter = fall_counter + 1

ElseIf (diff_weight(count) = 0) Then

     nochange_counter = _counter + 1

End If

Next

Console.WriteLine("The number of students having a rise in weight: {0}", rise_counter)

Console.WriteLine("The number of students having a fall in weight: {0}", fall_counter)

Console.WriteLine("The number of students having a no change in weight: {0}", nochange_counter)
```

Question 12: List down suitable data sets along with examples that you may use and test on your code.

Answer:

There are three main types of data sets:

1. Extreme Data (Extreme values of weight)
Examples: 15.1 Kg, 199.9 Kg

2. Normal Data (Normal values of weight)
Examples: 45Kg, 60.5Kg, 100 Kg

3. Abnormal Data (Weight values that are not acceptable by program at all)
Example: 0 Kg, -10 Kg, 500 Kg, 1000 Kg