UNIVERSIDAD SIMÓN BOLÍVAR
Departamento de Computación y Tecnología de la Información
CI-3641 - Lenguaje De Programación
Trimestre: Septiembre - Diciembre 2025
Blanyer Vielma, 16-11238

1. Pregunta 1

Parte A

Breve descripción de Visual Basic:

Es un lenguaje de programación orientado a eventos desarrollado por Microsoft. Su versión moderna, VB.NET, se integra con la plataforma .NET y permite crear aplicaciones de escritorio, web y móviles. Su sintaxis es sencilla y legible..

Alcances y Asociaciones: Tiene alcance estático más asociación superficial, Visual Basic usa asociación superficial, porque al pasar funciones o subrutinas como parámetros, el entorno léxico en que fueron definidas es el que se conserva.

Módulos que dispone y formas de importar y exportar nombres

Dispone de creación de aliases, sobrecarga y polimorfismo

Aliases: Si es válido usarlo en Visual Basic, en la documentación (https://learn.microsoft.com/es-es/dotnet/visual-basic/language-reference/statements/alias-clause) se indica que es posible usarlo tanto para asignar un alias a una función externa como para un tipo de datos. Por ejemplo:

```
Declare Function GetUserName Lib "advapi32.dll" Alias "GetUserNameA" (ByVal lpBuffer As String, ByRef nSize As Integer) As Integer

Imports VBInt = System.Int32
```

SobreCarga: Si se permite e incluso de diferentes formas, es posible la sobrecarga por incremento del número de parámetros o por cambio del tipo de datos de los parámetros (https://learn.microsoft.com/es-es/dotnet/visual-basic/programming-guide/language-features/procedures/procedure-overloading)

```
Sub post(custName As String, amount As Single)
   Console.WriteLine("Por nombre: " & custName & " → $" & amount)
End Sub

Sub post(custAcct As Integer, amount As Single)
   Console.WriteLine("Por cuenta: " & custAcct & " → $" & amount)
End Sub
```

Polimorfismo: El lenguaje es compatible también con polimorfismo.

```
Class Animal
   Overridable Sub Mover()
      Console.WriteLine("El animal se mueve.")
   End Sub
End Class

Class Pez
   Inherits Animal

   Overrides Sub Mover()
      Console.WriteLine("El pez nada.")
   End Sub
End Class
```

Herramientas Potenciales a Desarrolladores

Parte b

1. Implementación de función recursiva para rotar una cadena k posiciones

```
Imports System

Module Module1
```

```
    ' Función recursiva para rotar una cadena k posiciones
    Function rotar(w As String, k As Integer) As String
        If k = 0 OrElse w.Length = 0 Then
            Return w
        Else
            Dim a As String = w.Substring(0, 1)
            Dim x As String = w.Substring(1)
            Return rotar(x & a, k - 1)
        End If
    End Function

    Sub Main()
        Console.WriteLine("rotar(""Hola"", 0) = " & rotar("Hola", 0))
        Console.WriteLine("rotar(""Hola"", 1) = " & rotar("Hola", 1))
        Console.WriteLine("rotar(""Hola"", 2) = " & rotar("Hola", 2))
        Console.WriteLine("rotar(""Hola"", 3) = " & rotar("Hola", 3))
        Console.WriteLine("rotar(""Hola"", 4) = " & rotar("Hola", 4))
        Console.WriteLine("rotar(""Hola"", 5) = " & rotar("Hola", 5))

        Console.WriteLine("Presiona ENTER para salir.")
        Console.ReadLine()
    End Sub

End Module
```

2. Implementación de funcion de calculo del producto de matrices cuadradas con calculo de matriz transpuesta

```
Imports System

Module Module1

    ' Función para calcular la transpuesta de una matriz cuadrada
    Function Transpuesta(matriz(,) As Integer) As Integer(,)
        Dim n As Integer = matriz.GetLength(0)
        Dim resultado(n - 1, n - 1) As Integer

        For i As Integer = 0 To n - 1
            For j As Integer = 0 To n - 1
                resultado(j, i) = matriz(i, j)
            Next
        Next

        Return resultado
    End Function
```

```
' Función para calcular el producto de dos matrices cuadradas
Function Producto(m1(,) As Integer, m2(,) As Integer) As Integer(,)
    Dim n As Integer = m1.GetLength(0)
    Dim resultado(n - 1, n - 1) As Integer

    For i As Integer = 0 To n - 1
        For j As Integer = 0 To n - 1
            resultado(i, j) = 0
            For k As Integer = 0 To n - 1
                resultado(i, j) += m1(i, k) * m2(k, j)
            Next
        Next
    Next

    Return resultado
End Function

' Función para imprimir una matriz
Sub Imprimir(matriz(,) As Integer)
    Dim n As Integer = matriz.GetLength(0)
    For i As Integer = 0 To n - 1
        For j As Integer = 0 To n - 1
            Console.Write(matriz(i, j).ToString().PadLeft(5))
        Next
        Console.WriteLine()
    Next
    Console.WriteLine()
End Sub

Sub Main()
    ' Matrices de prueba 3x3
    Dim A(,) As Integer = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}
    Dim AT(,) As Integer = Transpuesta(A)

    Console.WriteLine("Matriz A:")
    Imprimir(A)

    Console.WriteLine("Matriz B = A_{T}:")
    Imprimir(AT)

    Console.WriteLine("Producto A × AT:")
    Imprimir(Producto(A, AT))

    Console.WriteLine("Presiona ENTER para salir.")
    Console.ReadLine()
```

```
    End Sub

End Module
```

Pregunta 2:

Caso 1 Alcance estático y asociación profunda:
Valores Iniciales:

X = 2, Y = 3, Z = 8

Valores Globales

a = Y + Z + 1 = 3 + 8 + 1 = 12,
b = X + Y + 1 = 2 + 3 + 1 = 6
c = Z + Y + 1 = 8 + 3 + 1 = 12

Paso 1 - Estado inicial (antes de la llamada a P)

Global X = 2 Y = 3 Z = 8 a = 12 b = 6 c = 12 R(b) -> a := b + c - 1 Q(a,r) -> { b:= a + 1; R() } P(a,s,t) -> procedimiento principal (main)

-> Hay una llamada a P, se prepara la ejecución P(a, s, t) -> P(12, Q, R)

Paso 2 - Crear el primer marco de pila de P

P(a=12) s = Q -> del entorno global t = R -> del entorno global
global a = 12 b = 6 c = 12

Paso 3 - Se define Q,R y c interna

P(a=12, s = Q, t = R)
c = a + b = 12 + 6 = 18
R(a) interna1 -> {
b := c + a 1
}
Q(b,r) interna1 -> {
c := a + b
r(c + a)
t(c + b)
}

global a = 12 b = 6 c = 12

-> Se evalúa la condición (12 < 24) True
- llamada recursiva: P(12 + 24, s, R) -> P(36, Q, R_local)

Paso 4 - Se llama recursivamente P

P(a=36, s = s, t = R_interna1)
c = a + b = 36 + 6 = 42
R(a) interna2 -> {
b := c + a 1
}
Q(b,r) interna2 -> {
c := a + b
r(c + a)
t(c + b)
}

P(a=12, s = Q, t = R)
c = a + b = 12 + 6 = 18
R(a) interna1 -> {
b := c + a 1
}
Q(b,r) interna1 -> {
c := a + b
r(c + a)
t(c + b)
}

global a = 12 b = 6 c = 12

-> Se evalúa la condición (36 < 24) False

- Se entra a else

Paso 5 - Se entra a else y se evalúa s(c * a, R)

P(a=36, s = s, t = R_interna1)
c = a + b = 36 + 6 = 42
R(a) interna2 -> {
b := c + a 1
}
Q(b,r) interna2 -> {
c := a + b
r(c + a)
t(c + b)
}

P(a=12, s = Q, t = R)
c = a + b = 12 + 6 = 18
R(a) interna1 -> {
b := c + a 1
}
Q(b,r) interna1 -> {
c := a + b
r(c + a)
t(c + b)
}

global a = 12 b = 6 c = 12

- s(c * a = 42 + 43 = 85, R = Rinterna2) -> Qinterna1(b = 85, r = Rinterna2)
Dentro de Q

c := a+ b = 43 + 6 = 49
r(c + a) = Rinterna2(a = 43 + 49 = 92) -> b := c + a 1 = 49 + 92 + 1 = 142
t(c + b) = Rinterna2(a = 43 + 142 = 185) -> b := c + a + = 49 + 185 + = 235