Clase 1:

Temario:

1.1 Palabras reservadas a destacar para esta clase

Dim	Se utiliza para reservar memoria para almacenar los datos de una variable
Const	Se utiliza para declarar una constante
Integer	Tipo de dato que almacena un número entero
Float	Tipo de dato que almacena un número decimal
Double	Tipo de dato que almacena un número decimal con mayor precisión y rango
Char	Tipo de dato que almacena un caracter
String	Tipo de dato que almacena un texto
Boolean	Tipo de dato que almacena un resultado lógico (true o false)
If	Bloque de control de flujo. Me permite evaluar una condición que, en caso de cumplirse, ejecuta un bloque de código
Else	Debe utilizarse dentro de un bloque If. Esta sentencia se ejecuta en caso de que la condición no se cumpla. También puede combinarse con la palabra If para evaluar una segunda condición en caso de que la primera no se cumpla

Select Case	Bloque De control de flujo. Similar al If, permite elegir qué bloque de código ejecutar en base al valor actual del parámetro que recibe. Cada Case dentro del bloque permite ejecutar una condición distinta
Case	Esta sentencia se usa dentro del Select Case para crear una nueva posible ejecución de código
While	Bloque de iteración (o bucle). Ejecuta un bloque de código mientras se cumpla una condición. Se verifica que la condición se cumpla al inicio del bloque
Do While	Tipo de bucle. Similar al While en funcionamiento, con la diferencia de que este bloque evalúa si la condición se sigue cumpliendo al final de la ejecución del bucle, con lo que Do While **se ejecuta al menos una vez**
For	Tipo de bucle. Permite definir una cantidad de iteraciones(repeticiones) a realizar

1.2 Variables

Las variables son pequeños bloques de memoria en donde almacenamos información. Podemos pensar en ellas como si fuera un locker.El número del locker representaría la dirección de memoria en donde almacenamos la variable, mientras que su interior guarda el contenido (en el caso de la variable, el valor que deseamos guardar).

El nombre de una variable se conoce como identificador. Por convención, los identificadores se nombran en camelcase (la primera palabra del nombre empieza en minúscula, y las subsecuentes palabras empiezan en mayúscula.

Dim miVariableCamelCase As Integer

Es importante notar que el compilador distingue entre mayúsculas y minúsculas. De esta forma si tuviera las variables min, Min y MIN, las 3 serían variables distintas

Una variable puede tomar valores distintos a lo largo de la ejecución del programa,

1.3 Constantes

Las constantes, a diferencia de las variables, no lleva la palabra reservada Dim, sino que utiliza la palabra reservada **Const .** Esto hace que el compilador detecte que esto no se trata de una variable, sino de una constante. Como su nombre indica, el valor de una constante no puede cambiar a lo largo de la ejecución del programa. Otro detalle es que una constante debe ser inicializada en la misma línea en que se la declara

Const PI = 3.1415926535

1.4 Operadores

1.4.1 Aritméticos

+	**Suma**
-	**Resta**
*****	**Multiplicación**
/	**División**
Mod	**Resto**

1.4.2 Relacionales y Lógicos

<	**Menor**
<=	**Menor o igual**
>	**Mayor**
>=	**Mayor o igual**
=	**Igual**
<>	**Distinto**
And	**Y. Compara dos condiciones y devuelve true solamente si se cumplen ambas**

Or	O. Compara dos condiciones y devuelve true con que una de ellas sea verdadera

1.5 Bloques de control de flujo

If - Else If - Else

El bloque if evalúa una condición booleana, es decir, una condición que pueda ser verdadera o falsa. En caso de ser cierta, se ejecuta el código de contiene. De lo contrario, este bloque no se ejecuta. Su estructura es la siguiente

```
If [mi Condicion] Then
        Bloque de código a ejecutar
End If
```

Cabe destacar que un bloque if puede evaluar múltiples condiciones. Por ejemplo, si quisiéramos ejecutar un bloque de código solamente cuando la variable miNumero sea mayor a 0 y tambien sea par, tendríamos

```
If miNumero > 0 And miNumero Mod 2 = 0 Then
        Bloque de código a ejecutar
End If
```

También podemos definir un bloque de código que se ejecuta en caso de que la condición sea falsa. Para esto utilizamos la palabra reservada Else

```
If miNumero Mod 2 = 0 Then
        Console.WriteLine("El numero es par")
Else
        Console.WriteLine("El número es impar")
End If
```

Por último, podemos evaluar condiciones alternativas en caso de que la primera sea falsa, anidando un bloque Else If dentro de otro

```
If miNumero = 0 Then
        Console.WriteLine("El número es cero")
Else If miNumero > 0 Then
        Console.WriteLine("El numero es mayor a 0")
Else
        Console.WriteLine("El número es menor a 0")
```

End If

Select Case

La estructura Select Case cumple una función similar al If, con la diferencia de que utiliza casos predefinidos para evaluar qué código ejecutar. en base a la variable que recibe. La estructura básica es

Select Case miVariable

Case {valor 1}
Código a ejecutar
break
Case {valor 2}
Código a ejecutar
break
Case Else
Código a ejecutar por defecto (el valor recibido no pertenece a ningún Case. Este es un case opcional, no necesariamente tiene que estar)

End Select

Esta estructura es útil cuando se deben evaluar múltiples condiciones. Es importante notar el uso de la palabra **break** al final de cada caso. De no incluirla, el programa ejecutaría caso a caso desde el punto al cual entre (si tengo 10 casos y entro al caso 5 sin utilizar ningún break, luego se ejecutan el 6, 7, 8, 9 y 10)

Por ejemplo, si quisiera almacenar la cantidad de días que tiene un mes utilizando esta estructura podría hacer el siguiente código

Select Case mes

Case “Enero”
días = 31
break
Case “Febrero”
dias = 28
break
Case “Marzo”
dias = 31
break
Case “Abril”
dias = 30
break

```
Case “Mayo”
        dias =  31
        break
Case “Junio”
        dias =  30
        break
Case “Julio”
        dias =  31
        break
Case “Agosto”
        dias =  31
        break
Case “Septiembre”
        dias =  30
        break
Case “Octubre”
        dias =  31
        break
Case “Noviembre”
        dias =  30
        break
Case “Diciembre”
        dias =  31
        break

End Select
```

El código de arriba, aunque podría realizarse con una serie de If - Else If anidados, quedaría mucho más largo y difícil de leer

1.6 Bloques de iteración (bucles) (While - Do While - For)

While

Este bloque evalúa una condición de la misma forma en que lo hace el if, con la diferencia de que, en caso de cumplirse, ese bloque de código repite su ejecución hasta que la condición deje de cumplirse. Su estructura es la siguiente

```
While [condición a analizar]
        Bloque de código que quiero ejecutar
End While
```

Cuando el programa se encuentra con un bloque While, **la condición se evalúa antes de entrar al bloque**. Esto quiere decir que existe la posibilidad de que este

código no se ejecute nunca. Un punto importante a destacar es que en esta estructura tenemos que, por un lado, evaluar una condición (lo cual se hace al comienzo), pero luego durante la ejecución del bloque de código interno tenemos que modificar esta condición para que eventualmente deje de repetirse. Caso contrario, se produce un bucle infinito, lo cual causaría un cierre de la aplicación

```
While miNumero < 0
        número -= 1
End While
```

En el caso de arriba también tenemos un ejemplo de bucle infinito. Si entramos a este bucle porque el número es menor a 0, le continuaremos restando 1 hasta que el número sea mayor a 0, lo cual es imposible

Do While

Con un funcionamiento similar al While,. Este bloque de código posee una única diferencia, y es que la condición que permite romper el bucle se evalúa **al final de la ejecución del bucle**. Esto significa que está garantizado que el Do While se ejecute al menos una vez. Su estructura, junto con un ejemplo de uso, podría ser la siguiente

```
Do

        Console.WriteLine("Ingrese un número mayor a 0")

 Loop While Integer.TryParse(Console.ReadLine(), numero) And numero > 0
```

For

El bucle for, a diferencia de los anteriores, define una condición de inicio y una de final para el bucle. Esto nos permite definir cuántas veces queremos que se ejecute el bucle, mientras que los bloques anteriores pueden ejecutarse (o no) una cantidad incierta de veces. ¿Cuántas veces va a ingresar el usuario un número menor a 0 en el ejemplo de arriba, o cuantas veces ingresará? No puede saberse.

La estructura de un bucle for es

```
For i= [Valor inicial] To [Valor final (incluido)]
         Bloque de código a ejecutar
      Next
```

Algo importante a destacar de todos los bucles, independientemente del tipo, es que todos pueden romperse de forma anticipada utilizando la palabra reservada **break**