INSTITUTO SUPERIOR UNIVERSITARIO COTOPAXI

GUÍA DE ESTUDIO

PROGRAMACIÓN BÁSICA

AUTOR

CÉSAR GRANIZO
PAMELA CHACHA

Instituto Superior Universitario Cotopaxi

PROGRAMACIÓN BÁSICA

Carrera en Electromecánica

Guía de estudio

César A. Granizo L.
Pamela M. Chacha M.

Guía de estudio de Programación

Carrera en Electromecánica CEM-102

César A. Granizo L.

2024

Esta publicación ha sido sometida a revisión interna y validación por parte de los jefes de área y coordinadores de carrera institucionales. Sin embargo, no ha completado su proceso total de editorialización.

Diseño de portadas: Raúl Jiménez Tello.

Versión 1.0

Instituto Superior Universitario Cotopaxi

Latacunga - Ecuador

Instituto Superior Universitario Cotopaxi

Nuestra Historia

El Instituto Superior Universitario Cotopaxi es un ícono de la transformación y revalorización de las políticas públicas en favor de la educación técnica y tecnológica a nivel de todo el Ecuador.

Misión

Somos una institución de educación superior, orientada en la formación integral de profesionales de tercer nivel competentes e innovadores con compromiso ético, social y ambiental que fomentan el desarrollo territorial sostenible.

Visión

Ser un instituto superior universitario con altos estándares de calidad, referente de la transformación técnica y tecnológica que contribuya al desarrollo sustentable y sostenible de la sociedad.

Gestión de Actividades de aprendizaje

Componente Docencia

Son actividades enfocas al alcance de las actitudes que permitan alcanzar los resultados de aprendizaje a lo largo del desarrollo de las unidades que conforman la guía de estudio, el mismo puede ser acompañado por el docente o en forma colaborativa.

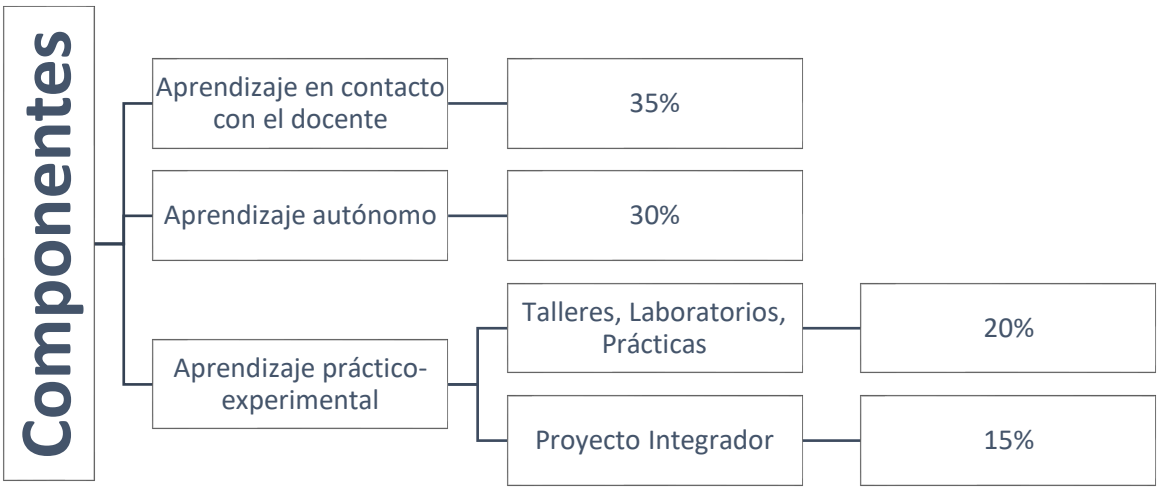
Prácticas Aprendizaje:

Son actividades que permiten que los estudiantes en contribución con su docente la consolidación de los resultados de aprendizaje en forma práctica, con el desarrollo y aplicación de conocimientos teóricos y métodos experimentales.

Componente Trabajo Autónomo:

Son actividades que permiten fortalecer las áreas específicas o amplias de conocimiento, a través de investigaciones bibliográficas, tareas o talleres. El estudiante organiza la forma y tiempo de las actividades a desarrollar.

Evaluación del Estudiante por Resultados de Aprendizaje



Instrucciones generales

Estimado estudiante revisar este acápite es de importancia porque le permitirá observar la secuencialidad de los procesos de la asignatura de Programación; para que alcance un aprendizaje significativo durante su formación académica.

La guía de estudio será un uno de los medios de orientación y recurso para el aprendizaje de la asignatura de Programación. Cada unidad didáctica inicia con una breve introducción y el resultado de aprendizaje de la asignatura

Para realizar un diagnóstico de conocimientos el docente abrirá una discusión en grupos sobre una pregunta que parta de interrogantes significativas para los alumnos, esto permitirá conocer si el estudiante cuenta con conocimientos previos de la asignatura, sea por su formación escolar o por su experiencia cotidiana.

En las actividades de desarrollo tienen la finalidad para que el estudiante interaccione con una nueva información o con una serie de conocimientos previos en mayor o menor medida acerca del tema. Estas fuentes de información pueden ser la exposición docente, discusión sobre una lectura, video relacionados al tema y apoyo de medios digitales como Moodle, Google Drive, YouTube, etc.

Las unidades de la guía de estudio deben estar desarrolladas en consecuencia con los resultados de aprendizaje, por tal motivo se recomienda seguir el orden en el que se encuentran.

Antes de iniciar el nuevo tema es importante haber comprendido la unidad anterior, caso contrario repase de nuevo o consulte a su profesor de la asignatura, quien le ayudará a clarificar los temas en los que tenga dificultad.

El tiempo para el desarrollo de la asignatura es de 16 semanas comprendidas en 32 horas de docencia, 48 horas de aprendizaje autónomo. Se recomienda al estudiante, programar un horario de estudio de cuatro horas semanales como mínimo, para la asignatura.

La ponderación de las actividades que evaluarán durante el periodo serás calificadas sobre diez puntos (10/10).

Para la evaluación de actividades de tareas e informes de prácticas de aprendizaje se revisará: originalidad, presentación *claridad, orden y fecha de entrega.* El desarrollo del documento o informe seguirá las directrices de las normas APA de la institución. Además, se realizarán evaluaciones de tareas en aula y autónomas como actividades expositivas, debates, análisis de casos, mediante una rúbrica.

La síntesis del proceso de aprendizaje se realizará mediante la aplicación de un cuestionario al final de la guía de estudio, esta evaluación comprende las evidencias de aprendizaje significativo alcanzadas a lo largo del desarrollo de las actividades realizadas y permitirá hacer los ajustes pertinentes a la secuencia didáctica en función de los resultados obtenidos.

ÍNDICE

Fundamentos de la Programación

1 Variables y constantes

- Una **variable** es una entidad que permite almacenar datos que pueden cambiar durante la ejecución del programa (Joyanes Aguilar, 2011). Las variables tienen un tipo de dato que define qué tipo de información pueden almacenar (por ejemplo, números enteros, texto, etc.).
- Una **constante**, por el contrario, es un valor fijo que no puede ser alterado durante la ejecución del programa (Sebesta, 2016).

  Ejemplo en pseudocódigo:

  VARIABLE edad : ENTERO

  CONSTANTE PI : REAL = 3.1416

  Ejercicios

1. Declara una variable para almacenar el nombre de una persona y una constante con el valor de la gravedad terrestre (9.8).
2. Escribe un programa que almacene la edad de una persona y calcule su edad en el próximo año.
3. Reflexiona: ¿Por qué es más eficiente usar constantes para valores como PI o la gravedad?

1.1.1 Tipos de datos

Teoría

- Los **tipos de datos** son fundamentales para definir qué tipo de información almacenarán las variables en un programa (Santos et al., 2020). Entre los tipos más comunes están:
  - **Enteros (INT)**: Para números sin decimales.
  - **Reales (FLOAT o REAL)**: Para números con decimales.
  - **Cadenas (STRING)**: Para texto.
  - **Booleanos (BOOL)**: Verdadero o falso.

  VARIABLE nombre : CADENA

VARIABLE temperatura : REAL

Ejercicios

1. Declara variables para almacenar:
    - El nombre de un producto (cadena).
    - Su precio (real).
    - La cantidad en stock (entero).
2. Convierte un valor real a entero en un programa.
3. Escribe un programa que determine si un número es positivo, negativo o cero.

1.1.2 Condicionales, bucles y arrays

Teoría

- **Condicionales**: Permiten que un programa tome decisiones basadas en condiciones específicas (Forbellone & Cypser, 2005). Por ejemplo:

  SI (edad >= 18) ENTONCES

    IMPRIMIR "Mayor de edad"

  FIN_SI

- **Bucles**: Repiten un bloque de código mientras se cumpla una condición (Sebesta, 2016). Ejemplo:

  PARA i <- 1 HASTA 5 HACER

    IMPRIMIR i

  FIN_PARA

- **Arrays**: Son estructuras que permiten almacenar varios valores en una sola variable de manera ordenada (Joyanes Aguilar, 2011). Ejemplo:

  VARIABLE numeros : ARREGLO[5] DE ENTERO

Ejercicios

1. Escribe un programa que determine si un número es par o impar.
2. Crea un bucle que imprima los números del 1 al 10.
3. Declara un array con los nombres de 5 ciudades y muéstralos en pantalla.

1.1.3 Pseudocódigo

El **pseudocódigo** es una herramienta para representar algoritmos utilizando lenguaje natural estructurado, siendo muy útil para planificar programas antes de escribir código real (Santos et al., 2020).

Ejemplo:

```
INICIO
   LEER numero
   SI (numero > 0) ENTONCES
      IMPRIMIR "Número positivo"
   FIN_SI
FIN
```

Ejercicios

1. Escribe un pseudocódigo que calcule el área de un triángulo.
2. Desarrolla un pseudocódigo que determine el mayor de dos números.
3. Diseña un pseudocódigo para calcular el promedio de tres calificaciones.

1.1.4 Diagramas de flujo (DFD)

Teoría

Los **diagramas de flujo** representan gráficamente los pasos de un algoritmo, facilitando su comprensión y comunicación (Pressman, 2020). Los símbolos principales son:

- **Óvalo**: Inicio/Fin.
- **Rectángulo**: Proceso.
- **Rombo**: Decisión.

Ejemplo: Un algoritmo para determinar si un número es par o impar.

Ejercicios

1. Diseña un diagrama de flujo para calcular el área de un círculo.
2. Crea un diagrama que valide si un usuario puede acceder a un sistema con su contraseña.

3. Dibuja un diagrama que sume los números del 1 al 10 utilizando un bucle.

2 Estructuras de Programación en C++

2.1 Estructuras Secuenciales

Definición

Las **estructuras secuenciales** son el tipo de estructura de control más básica. En estas, las instrucciones se ejecutan en el mismo orden en el que están escritas, una tras otra, sin interrupciones. Estas estructuras son esenciales para implementar operaciones simples como cálculos o la captura de datos del usuario (Joyanes Aguilar, 2011).

Ejemplo en C++

```
#include <iostream>

using namespace std;

int main() {

   int num1, num2, suma;

   cout << "Ingrese el primer número: ";

   cin >> num1;

   cout << "Ingrese el segundo número: ";

   cin >> num2;

   suma = num1 + num2;

   cout << "La suma es: " << suma << endl;

   return 0;

}
```

Ejercicios

1. Escribe un programa que calcule el área de un rectángulo (base × altura).
2. Crea un programa que convierta grados Celsius a Fahrenheit usando la fórmula: F=(C×9/5)+32F = (C \times 9/5) + 32F=(C×9/5)+32.
3. Desarrolla un programa que pida dos números y muestre su promedio.

2.2 Estructuras Selectivas

2.2.1 Definición

Las **estructuras selectivas** permiten que un programa tome decisiones basadas en condiciones específicas. Estas estructuras son fundamentales para la implementación de lógica condicional y la creación de programas dinámicos (Sebesta, 2016). Existen dos variantes principales:

1. **Condición simple (if):** Ejecuta una instrucción o bloque de instrucciones si la condición evaluada es verdadera.
2. **Condición compuesta (if-else):** Permite ejecutar un bloque de código si la condición es verdadera y otro bloque si es falsa.

   Ejemplo en C++

   Estructura simple:

```
#include <iostream>
using namespace std;

int main() {
    int edad;
    cout << "Ingrese su edad: ";
    cin >> edad;
    if (edad >= 18) {
        cout << "Eres mayor de edad." << endl;
    }
    return 0;
}
```

   Estructura compuesta:

```
#include <iostream>
using namespace std;
```

```
int main() {
    int numero;
    cout << "Ingrese un número: ";
    cin >> numero;
    if (numero % 2 == 0) {
        cout << "El número es par." << endl;
    } else {
        cout << "El número es impar." << endl;
    }
    return 0;
}
```

Ejercicios

1. Escribe un programa que determine si un número ingresado por el usuario es positivo, negativo o cero.
2. Desarrolla un programa que solicite la calificación de un estudiante y muestre si aprobó (≥ 60) o no.
3. Escribe un programa que calcule si un año ingresado es bisiesto. (Un año es bisiesto si es divisible entre 4 pero no entre 100, excepto si es divisible entre 400).

2.2.2 Estructuras Repetitivas

Definición

Las **estructuras repetitivas**, también conocidas como bucles, permiten la ejecución de un bloque de código varias veces mientras se cumpla una condición específica. Este enfoque es fundamental para optimizar tareas repetitivas y manejar grandes cantidades de datos (Forbellone & Cypser, 2005). Los tres tipos más comunes en C++ son:

1. **for:** Usado cuando el número de iteraciones es conocido.
2. **while:** Ejecuta un bloque de código mientras una condición sea verdadera.
3. **do-while:** Similar al while, pero garantiza que el bloque se ejecute al menos una vez.

Ejemplo en C++

Bucle for:

```
#include <iostream>
using namespace std;
int main() {
    for (int i = 1; i <= 5; i++) {
        cout << "Iteración " << i << endl;
    }
    return 0;
}
```

Bucle while:

```
#include <iostream>
using namespace std;

int main() {
    int contador = 1;
    while (contador <= 5) {
        cout << "Contador: " << contador << endl;
        contador++;
    }
    return 0;
}
```

Bucle do-while:

```
#include <iostream>
using namespace std;
int main() {
    int numero;
```

```
    do {
        cout << "Ingrese un número mayor que 0: ";
        cin >> numero;
    } while (numero <= 0);
    cout << "Número válido: " << numero << endl;
    return 0;
}
```

Ejercicios

1. Escribe un programa que imprima los números del 1 al 10 utilizando un bucle for.
2. Crea un programa que calcule la suma de los primeros n números naturales ingresados por el usuario utilizando un bucle while.
3. Diseña un programa que pida una contraseña al usuario y continúe solicitándola hasta que sea correcta (usando do-while).

4. Escribe un programa que calcule el perímetro y el área de un círculo. Usa la fórmula A=π×r2A = \pi \times r^2A=π×r2 y P=2×π×rP = 2 \times \pi \times rP=2×π×r, donde rrr es el radio ingresado por el usuario. Usa la constante M_PI de la librería <cmath>.
5. Realiza un programa que reciba tres números y calcule su suma, resta, multiplicación y división.
6. Crea un programa que convierta kilómetros a millas. Usa la fórmula millas=km×0.621371millas = km \times 0.621371millas=km×0.621371.
7. Desarrolla un programa que reciba dos números e indique cuál es el mayor o si son iguales.

   Ejemplo:

```
#include <iostream>
#include <cmath> // Para usar M_PI
using namespace std;
```

```
int main() {
    double radio, area, perimetro;
    cout << "Ingrese el radio del círculo: ";
    cin >> radio;
    area = M_PI * pow(radio, 2);
    perimetro = 2 * M_PI * radio;
    cout << "El área del círculo es: " << area << endl;
    cout << "El perímetro del círculo es: " << perimetro << endl;
    return 0;
}
```

2.2.3 Estructuras Selectivas

Ejercicios Adicionales

4. Escribe un programa que determine si un número ingresado por el usuario es múltiplo de 5.
5. Desarrolla un programa que reciba tres números y determine cuál es el mayor.
6. Diseña un programa que lea una calificación (entre 0 y 100) e imprima la letra correspondiente (A, B, C, D o F) siguiendo este esquema:
    - A: 90–100
    - B: 80–89
    - C: 70–79
    - D: 60–69
    - F: Menor a 60
7. Crea un programa que determine si un carácter ingresado por el usuario es una vocal o una consonante.

Ejemplo:

```
#include <iostream>
using namespace std;
```

```
int main() {
    char letra;
    cout << "Ingrese una letra: ";
    cin >> letra;
    if (letra == 'a' || letra == 'e' || letra == 'i' || letra == 'o' || letra == 'u' ||
        letra == 'A' || letra == 'E' || letra == 'I' || letra == 'O' || letra == 'U') {
        cout << "La letra es una vocal." << endl;
    } else {
        cout << "La letra es una consonante." << endl;
    }
    return 0;
}
```

2.2.4 Estructuras Repetitivas

Ejercicios Adicionales

4. Escribe un programa que imprima todos los números pares del 1 al 100 utilizando un bucle for.
5. Crea un programa que pida un número al usuario y calcule el factorial de ese número utilizando un bucle while.
6. Diseña un programa que lea números ingresados por el usuario y los sume hasta que se introduzca el número 0 (utiliza un bucle do-while).
7. Escribe un programa que genere la tabla de multiplicar de un número ingresado por el usuario, desde 1 hasta 10.
8. Diseña un programa que determine cuántos números positivos, negativos y ceros hay en una lista de n números ingresados por el usuario.

Ejemplo 1: Generar números pares del 1 al 100

```
#include <iostream>
using namespace std;
```

```
int main() {
    for (int i = 1; i <= 100; i++) {
        if (i % 2 == 0) {
            cout << i << " ";
        }
    }
    cout << endl;
    return 0;
}
```

Ejemplo 2: Calcular el factorial de un número

```
#include <iostream>
using namespace std;
int main() {
    int numero, factorial = 1;
    cout << "Ingrese un número: ";
    cin >> numero;
    if (numero < 0) {
        cout << "El factorial no está definido para números negativos." << endl;
    } else {
        int i = 1;
        while (i <= numero) {
            factorial *= i;
            i++;
        }
        cout << "El factorial de " << numero << " es: " << factorial << endl;
    }
    return 0;
}
```

Ejemplo 3: Tabla de multiplicar

```
#include <iostream>
using namespace std;

int main() {
    int numero;
    cout << "Ingrese un número para generar su tabla de multiplicar: ";
```

```
    cin >> numero;
    for (int i = 1; i <= 10; i++) {
        cout << numero << " x " << i << " = " << numero * i << endl;
    }
    return 0;
}
```

3 Introducción al Lenguaje Gráfico Visual Basic

Visual Basic (VB) es un lenguaje de programación desarrollado por Microsoft que utiliza una interfaz gráfica para diseñar aplicaciones, lo que lo hace altamente accesible para principiantes. Su enfoque orientado a eventos permite que los desarrolladores creen programas interactivos con facilidad (Perry, 2007).

3.1 Conceptos Básicos de Visual Basic

3.1.1 Definición

Visual Basic es un lenguaje de programación de alto nivel diseñado para facilitar la creación de aplicaciones gráficas. Se basa en el paradigma de la programación orientada a eventos, donde las acciones del usuario (clics, movimientos del ratón, etc.) desencadenan eventos en el programa (Bradley & Millspaugh, 2015).

3.1.2 Características Principales

- **Entorno de Desarrollo Integrado (IDE):** Incluye herramientas para diseñar formularios y escribir código en una sola interfaz.
- **Programación orientada a eventos:** Responde a interacciones del usuario como clics o pulsaciones de teclado.
- **Simplicidad:** Su sintaxis es clara y fácil de aprender.
- **Compatibilidad:** Se integra con otras aplicaciones de Microsoft Office, como Excel y Access.

Ejemplo Básico

```
Private Sub btnSaludar_Click()
    MsgBox "¡Hola, bienvenido a Visual Basic!"
End Sub
```

3.2 Control de Imágenes

3.2.1 Definición

Visual Basic permite crear interfaces gráficas (GUI) utilizando controles como botones, etiquetas y cuadros de texto, que se arrastran y colocan en un formulario dentro del IDE (Deitel & Deitel, 2013).

3.2.2 Controles Comunes

1. **Formulario:** El contenedor principal para la aplicación.
2. **Etiqueta (Label):** Muestra texto o información estática.
3. **Botón (Button):** Ejecuta acciones cuando se hace clic.
4. **Cuadro de texto (TextBox):** Permite la entrada de datos por parte del usuario.
5. **Caja de lista (ListBox):** Muestra una lista de elementos.

Ejemplo de Diseño

- **Formulario:** Diseñado con un botón btnSaludar y una etiqueta lblMensaje.
- Código para el evento del botón:

```
Private Sub btnSaludar_Click()
    lblMensaje.Text = "¡Hola, mundo!"
End Sub
```

Ejercicios

1. Diseña un formulario con:
    - Un cuadro de texto para ingresar el nombre del usuario.
    - Un botón que, al hacer clic, muestre un mensaje de saludo personalizado.
2. Crea un programa que utilice una lista desplegable para mostrar una lista de países

3.3 Archivos

3.3.1 Definición

La programación orientada a eventos se basa en responder a acciones específicas del usuario, como clics, movimientos del ratón o cambios en un cuadro de texto. Cada control tiene propiedades, métodos y eventos que pueden personalizarse (Perry, 2007).

3.3.2 Eventos Comunes

1. **Click:** Se ejecuta al hacer clic en un botón.
2. **Change:** Se activa cuando el valor de un control cambia.

3. **Load:** Se ejecuta al cargar un formulario.

Ejemplo

Un programa que cambia el color de fondo de un formulario al hacer clic en un botón:

```
Private Sub btnCambiarColor_Click()
    Me.BackColor = Color.Blue
End Sub
```

Ejercicios

1. Diseña un programa con tres botones:
    - Botón 1: Cambia el color del formulario a rojo.
    - Botón 2: Cambia el color del formulario a verde.
    - Botón 3: Restaura el color original.
2. Crea un formulario que muestre un cuadro de mensaje cuando el usuario cierre el formulario.

3.4 Base de Datos

3.4.1 Definición

Las variables en Visual Basic son espacios de memoria utilizados para almacenar datos. Cada variable debe declararse con un tipo de dato específico, como Integer, String o Boolean (Bradley & Millspaugh, 2015).

Tipos de Datos Comunes

- **Integer:** Almacena números enteros.
- **Double:** Almacena números decimales.
- **String:** Almacena texto.
- **Boolean:** Almacena valores lógicos (True o False).

Ejemplo de Uso

```
Private Sub btnCalcular_Click()
    Dim num1 As Integer = 10
    Dim num2 As Integer = 20
```

```
    Dim suma As Integer
    suma = num1 + num2
    MsgBox "La suma es: " & suma
End Sub
```

Ejercicios

1. Crea un programa que calcule el área de un triángulo. Usa las variables base y altura ingresadas por el usuario.
2. Diseña un formulario que pida el nombre y la edad del usuario y muestre un mensaje personalizado indicando si es mayor de edad.

3.5 Estructuras de Control

3.5.1 Definición

Visual Basic incluye estructuras de control como:

- **Condicionales (If...Then):** Permiten tomar decisiones.
- **Bucles (For, While):** Ejecutan código repetidamente mientras se cumpla una condición.

Ejemplo de Condicional

```
Private Sub btnVerificar_Click()
    Dim edad As Integer = 20
    If edad >= 18 Then
        MsgBox "Eres mayor de edad."
    Else
        MsgBox "Eres menor de edad."
    End If
End Sub
```

Ejemplo de Bucle

```
Private Sub btnContar_Click()
    For i As Integer = 1 To 10
        MsgBox "Número: " & i
    Next
End Sub
```

Ejercicios

1. Escribe un programa que determine si un número ingresado por el usuario es par o impar.
2. Crea un formulario que calcule la suma de los primeros n números naturales usando un bucle.

Ejercicios Adicionales

1. Crea un programa que solicite el nombre del usuario y, al hacer clic en un botón, muestre un mensaje de bienvenida en un cuadro de mensaje (MsgBox).
2. Diseña un programa que tenga dos botones: uno para mostrar un mensaje de "Hola, Mundo" y otro para cerrar la aplicación.
3. Implementa un programa que tenga un cuadro de texto para ingresar el nombre del usuario y, al pulsar un botón, cambie el texto de una etiqueta (Label) para mostrar: "Bienvenido, [Nombre]".

Ejemplo:

Código para mostrar un mensaje de bienvenida:

```
Private Sub btnSaludar_Click()
    Dim nombre As String
    nombre = InputBox("Ingrese su nombre:", "Bienvenida")
    MsgBox "¡Hola, " & nombre & "! Bienvenido a Visual Basic.", vbInformation, "Saludo"
End Sub
```

3.6 Elementos Básicos de la Interfaz de Usuario

Ejercicios Adicionales

1. Diseña un formulario con:
    - Dos cuadros de texto para ingresar dos números.
    - Un botón que calcule y muestre su suma, resta, multiplicación y división.
2. Crea un formulario con una lista desplegable (ComboBox) que permita seleccionar un color, y al hacer clic en un botón, cambie el color de fondo del formulario.
3. Diseña un programa que tenga una lista (ListBox) donde el usuario pueda agregar nombres ingresados en un cuadro de texto y mostrarlos en una etiqueta al pulsar un botón.
4. Implementa un formulario con:
    - Un control de selección de fecha (DateTimePicker).
    - Un botón que, al pulsarse, muestre en un cuadro de mensaje si la fecha seleccionada es anterior, igual o posterior a la fecha actual.

Ejemplo:

Código para cambiar el color de fondo:

```
Private Sub btnCambiarColor_Click()
    Dim colorSeleccionado As String
    colorSeleccionado = ComboBox1.SelectedItem
    Select Case colorSeleccionado
        Case "Rojo"
            Me.BackColor = Color.Red
        Case "Verde"
            Me.BackColor = Color.Green
        Case "Azul"
            Me.BackColor = Color.Blue
        Case Else
```

```
        MsgBox "Seleccione un color válido.", vbExclamation, "Advertencia"
    End Select
End Sub
```

3.7Programación Orientada a Eventos

Ejercicios Adicionales

1. Diseña un formulario con:
    - Una casilla de verificación (CheckBox) que, al activarse, habilite un cuadro de texto.
    - Un botón que muestre el contenido del cuadro de texto en un cuadro de mensaje.
2. Crea un programa con dos botones:
    - El primero deshabilita todos los controles del formulario.
    - El segundo los habilita nuevamente.
3. Diseña un programa que use un control de selección (RadioButton) para elegir entre dos opciones (por ejemplo, "Hombre" y "Mujer") y muestre la opción seleccionada en una etiqueta al pulsar un botón.
4. Implementa un programa que tenga un control ProgressBar que se incremente automáticamente cada segundo hasta completarse al 100%.

Ejemplo:

Código para habilitar un cuadro de texto con un CheckBox:

```
Private Sub chkHabilitar_CheckedChanged(sender As Object, e As EventArgs) Handles chkHabilitar.CheckedChanged
    If chkHabilitar.Checked Then
        txtEntrada.Enabled = True
    Else
        txtEntrada.Enabled = False
    End If
End Sub
```

3.8 Variables y Tipos de Datos en Visual Basic

Ejercicios Adicionales

1. Crea un programa que lea tres números ingresados por el usuario, los almacene en variables y calcule su promedio.
2. Diseña un formulario que permita ingresar el precio de un producto y un porcentaje de descuento, y calcule el precio final con el descuento aplicado.
3. Implementa un formulario que convierta una temperatura de Celsius a Fahrenheit y viceversa. Usa las fórmulas:
    - $F=(C \times 9/5)+32$F = (C \times 9/5) + 32F=(C×9/5)+32
    - $C=(F-32) \times 5/9$C = (F - 32) \times 5/9C=(F−32)×5/9.
4. Crea un programa que pida el nombre, edad y género de una persona, y muestre un mensaje personalizado dependiendo de si es mayor de edad o no.

Ejemplo:

Código para calcular el promedio:

```
Private Sub btnCalcularPromedio_Click()
    Dim num1, num2, num3, promedio As Double
    num1 = CDbl(txtNum1.Text)
    num2 = CDbl(txtNum2.Text)
    num3 = CDbl(txtNum3.Text)
    promedio = (num1 + num2 + num3) / 3
    MsgBox "El promedio es: " & promedio, vbInformation, "Resultado"
End Sub
```

3.9 Estructuras de Control

Ejercicios Adicionales

1. Escribe un programa que solicite una nota del 1 al 10 y evalúe si el estudiante aprobó (nota ≥ 7) o reprobó.
2. Crea un programa que use un bucle para calcular la suma de los primeros n números pares ingresados por el usuario.

3. Diseña un formulario con un cuadro de texto para ingresar un número, y muestra su tabla de multiplicar utilizando un bucle.
4. Implementa un programa que lea un número y determine si es primo. Usa un bucle For para verificar los divisores del número.
5. Diseña un programa que tenga un botón para generar 10 números aleatorios entre 1 y 100, los almacene en un arreglo y los muestre en una lista (ListBox).

Ejemplo:

Código para verificar si un número es primo:

```
Private Sub btnEsPrimo_Click()
    Dim num As Integer = CInt(txtNumero.Text)
    Dim esPrimo As Boolean = True
    If num <= 1 Then
        esPrimo = False
    Else
        For i As Integer = 2 To num - 1
            If num Mod i = 0 Then
                esPrimo = False
                Exit For
            End If
        Next
    End If
    If esPrimo Then
        MsgBox "El número " & num & " es primo.", vbInformation, "Resultado"
    Else
        MsgBox "El número " & num & " no es primo.", vbExclamation, "Resultado"
    End If
End Sub
```

Bibliografía

- Bradley, J., & Millspaugh, A. (2015). Programming in Visual Basic 2015. Pearson.
- Deitel, P., & Deitel, H. (2013). Visual Basic 2012: How to Program. Pearson.
- Forbellone, A., & Cypser, F. (2005). Algoritmos: Programación, lógica y estructuras de datos. Pearson.
- Joyanes Aguilar, L. (2011). Programación en C++: Algoritmos, estructura de datos y objetos. McGraw Hill.
- Perry, G. (2007). Visual Basic 2008 in Easy Steps. In Easy Steps.
- Santos, E., Martínez, P., & López, J. (2020). Introducción a la programación con Python. Ediciones Paraninfo.
- Sebesta, R. W. (2016). Conceptos de lenguajes de programación. Pearson.

Instituto Superior Universitario Cotopaxi
Parroquia Tanicuchí Panamericana
E35 km. 12 vía Latacunga - Quito