Trabajo Práctico 2
Programación y Estructura de Datos

Carrera: T1-23 INGENIERÍA EN SISTEMAS INFORMÁTICOS

Comisión: 1-B

Profesores:

- Cardacci Dario Guillermo
- Bazaes Balmaceda Emiliano Lautaro

Alumno: Gustavo Daniel Britez Alvarenga

Sede: Centro (San Juan)

Turno: Mañana

1. ¿Qué característica posee la programación visual

Características de la Programación Visual: La programación visual es un enfoque que permite a los desarrolladores crear aplicaciones utilizando elementos visuales y gráficos en lugar de escribir código de programación tradicional. Algunas de las características clave de la programación visual son:

Interfaz Gráfica: Los programadores pueden diseñar la lógica de la aplicación arrastrando y soltando elementos visuales en una interfaz gráfica.

Menos Código Manual: Se reduce la necesidad de escribir código manualmente, lo que puede agilizar el proceso de desarrollo.

Accesibilidad: Facilita la creación de interfaces de usuario interactivas y amigables.

Retroalimentación Visual: Los cambios en la lógica o el diseño se reflejan directamente en la interfaz visual, lo que permite a los desarrolladores ver los resultados inmediatamente.

2. ¿Qué es un control de usuario?

Un control de usuario es un elemento interactivo en una interfaz de usuario que permite a los usuarios interactuar con una aplicación. Los controles de usuario incluyen elementos como botones, casillas de verificación, cuadros de texto, menús desplegables, barras de desplazamiento, etc. Estos controles permiten a los usuarios proporcionar entrada y recibir retroalimentación de la aplicación.

3. ¿Cuáles son los tres elementos constitutivos que un control de usuario debe poseer?

Elementos Constitutivos de un Control de Usuario: Los tres elementos fundamentales que un control de usuario debe poseer son:

Propiedades: Cada control de usuario tiene propiedades que definen su apariencia, comportamiento y otros atributos. Estas propiedades se pueden configurar para personalizar cómo se ve y se comporta el control.

Eventos: Los controles de usuario generan eventos en respuesta a las acciones del usuario, como hacer clic en un botón o ingresar datos en un campo. Estos eventos permiten que la aplicación responda a las interacciones del usuario ejecutando acciones específicas.

Métodos: Los métodos son acciones que se pueden realizar en un control de usuario. Los métodos permiten programar comportamientos específicos para

el control, como establecer un valor en un cuadro de texto o activar una acción cuando se selecciona un elemento en un menú desplegable.

4. ¿Para qué se utiliza la ventana denominada Solution Explorer?

Sirve para administrar los proyectos y archivos de una solución.

5. ¿Para qué se utiliza la ventana denominada Properties?

La ventana Properties se utiliza para:

- Modificar Atributos Visuales
- Configurar Comportamiento
- Asignar Datos
- Definir Nombres y Identificadores
- Modificar Opciones Específicas

6. ¿Para qué se utiliza la ventana denominada Toolbox?

Sirve para mostrar los controles que puede agregar a proyectos de Visual Studio

7. ¿Qué es y que puede contener una Solución en el entorno de trabajo visto en clase?

Una solución es una estructura de nivel superior que contiene uno o varios proyectos relacionados. Una solución es un contenedor que agrupa proyectos para una aplicación o un conjunto de aplicaciones relacionadas. Puede contener información de configuración global y establecer relaciones entre los diferentes proyectos que lo componen.

8. ¿Qué es y que puede contener un Proyecto en el entorno de trabajo visto en clase?

Un proyecto se encuentra dentro de una solución. Representa una colección de archivos, recursos, código fuente y configuraciones necesarios para construir una parte específica de una aplicación.

Podemos entender mejor qué es un proyecto imaginándonos al paquete office, donde cada proyecto es un programa del paquete

- Excel
- PowerPoint
- Word

y el paquete office es la solución que contiene todos estos proyectos .

9. ¿Qué es un formulario?

Un formulario es una ventana gráfica que se utiliza para interactuar con el usuario en una aplicación. Los formularios son una parte esencial de las interfaces de usuario y permiten que los usuarios ingresen datos, realicen selecciones y realicen acciones en una aplicación.

10. Describa al menos 10 propiedades, 10 métodos y 10 eventos del control Button

Button

Propiedades	Métodos	Eventos
FlatStyle	DrawToBitmap	Click
Font	Hide	ClientSizeChanged
Text	BeginInvoke	ControlAdded
AllowDrop	Focus	DragDrop
Size	BringToFont	DragEnter
Enabled	Update	Enter
Anchor	Scale	KeyUp
Dock	Select	KeyDown
Location	Contains	Leave
Locked	Dispose	MouseEnter

11. Describa al menos 10 propiedades, 10 métodos y 10 eventos del control CheckBox.

CheckBox

Propiedades	Métodos	Eventos
BackColor	Select	KeyDown
Checked	BringToFont	KeyUp

Cursor	BeginInvoke	KeyPress
CheckState	Contains	PreviewKeyDown
Appearence	CreateControls	MouseLeave
CheckAlign	CreateGraphics	MouseMove
ForeColor	CreateObjRef	MouseUp
Image	DoDragDrop	MouseDown
RightToLeft	Dispose	MouseEnter
ImageList	GetHashCode	VisibleChanged

12. Describa al menos 10 propiedades, 10 métodos y 10 eventos del control Combobox.

ComboBox

Propiedades	**Métodos**	**Eventos**
AllowDrop	Select	ChangeUICues
ContextMenuStrip	SelectAll	ControlAdded
DrawMode	SelectNextControl	ControlRemoved
DropDownHeight	BeginInvoke	DrawItem
DropDownWidth	BeginUpDate	DropDown
Enabled	GetContainerControl	DropDownClosed
ImeMode	Contains	DropDownStyleChanged
Integralheight	LogicalToDeviceUnits	HelpRequest
ItemHeight	ScaleBitmapLogicalToDevic e	ImeModeChanged
MaxDropDownItems	GetPreferredSize	StyleChanged

13. Describa al menos 10 propiedades, 10 métodos y 10 eventos del control DateTimePicker.

DateTimePicker

Propiedades	Métodos	Eventos
Margin	GetType	Resize
Location	GetHashCode	RightToLeftChanged
Locked	GetChildAtPoint	RightToLeftLayoutChanged
GenerateMember	GetContainerControl	SizeChanged
MaximumSize	GetLifetimeService	StyleChanged
Size	GetNextControl	SystemColorsChanged
MinimumSize	GetPreferredSize	TabIndexChanged
AllowDrop	Scale	Validated
Enabled	ScaleBitmapLogicalToDevice	Validating
MaxDate	PointToScreen	VisibleChanged

14. Describa al menos 10 propiedades, 10 métodos y 10 eventos del control Timer.

Timer

Propiedades	Métodos	Eventos
Comportamiento	Stop	Tick
Datos	Star	Disposed ?
Diseño	ToString	
	Equals	
	Dispose	
	CreateObjRef	

	GetHashCode	
	GetType	
	InitializeLifetimeService	
	¿	

15. Seleccione 10 controles adicionales y de cada uno de ellos describa 5 métodos, 5 propiedades y 5 eventos.

BackGroundWorker1

Eventos [3] = DoWork - ProgressChanged - RunWorkerCompleted

Propiedades [5] = WorkerReportsProgress - WorkerSupportsCancellation - PropetyBinding - Name - GenerateMember

Metodos [5] = RunWorkerAsync - Dispose - CancelAsync - CreateObjRef - ReportProgress

Parte 2

1. ¿Cómo se declara un procedimiento? Ejemplo.

Un procedimiento se declara de la siguiente manera

Void nombre (params)

Tiene la particularidad de no poseer **Return** por lo tanto no retorna valores

2. ¿Cómo se declara una función? Ejemplo

Una función se declara

TipoDato nombre (params)

Tiene la particularidad de utilizar return y devolver valores del tipo de dato definido previamente

3. ¿Qué diferencia existe entre una función y un procedimiento?

La diferencia que existe entre una función y un procedimiento radica principalmente en el uso del RETURN.

Otra diferencia es que las funciones se declaran según el tipo de dato que devuelven y los procedimientos se declaran con VOID

4. ¿Qué significa que el parámetro de un procedimiento se pasa por valor?

Cuando pasamos por parámetro por valor estamos estableciendo el valor de entrada al params de dicho procedimiento como una copia del valor original

En el siguiente ejemplo podemos ver como el número que entra al procedimiento no cambia su valor luego de pasar por el procedimiento ya que el void no retorna el numero cambiado en la memoria del " int numeroOriginal "

```
class Program
{
    static void ModificarValor(int numero)
    {
        numero = 10; // Cambio local al parámetro
    }

    static void Main(string[] args)
    {
        int numeroOriginal = 5;
        Console.WriteLine("Número original: " + numeroOriginal);

        ModificarValor(numeroOriginal); // Llamada a la función

        Console.WriteLine("Número después de la función: " + numeroOriginal);
    }
```

```
    }
```

Resultado -"Número original: 5 & Número después de la función: 5”

5. **¿Qué significa que el parámetro de una función se pasa por referencia?**

```
class Program
{
    static void ModificarValor(ref int numero)
    {
        numero = 10; // Cambio que afecta al valor original del argumento
    }

    static void Main(string[] args)
    {
        int numeroOriginal = 5;
        Console.WriteLine("Número original: " + numeroOriginal);

        ModificarValor(ref numeroOriginal); // Llamada a la función con paso por referencia

        Console.WriteLine("Número después de la función: " + numeroOriginal);
    }
}
```

El valor pasado por parámetro ene la función hace referencia al valor ORIGINAL y puede ser cambiado al retornar dicho valor.

Parte 3

1. **Enumere las estructuras de decisión. Ejemplifique cada una de ellas y explique en que se diferencian.**

Las estructuras de desicion son 3

IF

```
int edad = 18;

if (edad >= 18)

{

   Console.WriteLine("Eres mayor de edad.");

}
```

IF - ELSE

```
int nota = 85

if (nota >= 60)

   Console.WriteLine("Aprobado");

else:

   Console.WriteLine("Reprobado");
```

SWITCH

```
char operador = '+';

switch (operador)
{
   case '+':
      Console.WriteLine("Suma");
      break;
   case '-':
      Console.WriteLine("Resta");
      break;
```

```
    default:
        Console.WriteLine("Operador desconocido");
        break;
}
```

Las principales diferencias entre estas estructuras radican en cómo manejan las condiciones y en la cantidad de opciones.

- If y if-else son adecuados para manejar una o dos condiciones
- Switch es más apropiado cuando se deben considerar múltiples casos.

Cada estructura tiene su lugar en diferentes situaciones, y la elección dependerá del problema que resolvamos.

2. Enumere las estructuras de repetición. Ejemplifique cada una de ellas y explique en que se diferencian.

FOR: Este bucle se utiliza cuando se conoce la cantidad exacta de iteraciones.

```
for (int i = 0; i < 5; i++)
{
    Console.WriteLine("Iteración #" + i);
}
```

WHILE: Este bucle se repite mientras una condición específica sea verdadera

```
int count = 0;
while (count < 5)
{
    Console.WriteLine("Iteración #" + count);
    count++;
}
```

DO WHILE Similar al bucle while, pero se verifica la condición después de cada iteración. Esto garantiza que el bucle se ejecute al menos una vez.

```
int num = 5;
do
{
    Console.WriteLine("Número: " + num);
    num--;
} while (num > 0);
```

3. ¿Cómo se crea una estructura en el entorno visto en clase?

```
namespace EjemploEstructura
{
    // Declaración de la estructura
    struct Persona
    {
        public string Nombre;
        public int Edad;
    }


    class Program
    {
        static void Main(string[] args)
        {
            // Crear una instancia de la estructura
            Persona persona1;
            persona1.Nombre = "Juan";
            persona1.Edad = 30;

            // Acceder a los campos de la estructura
            Console.WriteLine("Nombre: " + persona1.Nombre);
            Console.WriteLine("Edad: " + persona1.Edad);
        }
    }
}
```

4. ¿Qué es una propiedad?

La propiedad proporciona un mecanismo para poder acceder a la modificación de datos encapsulados

5. ¿Qué es un método?

Metodo / Funcion / Procedimiento, a menudo son utilizados como sinónimos pero no es correcto del todo, a grandes rasgos podemos decir que un método es una porción de codigo reutilizable que realiza una tarea especifica.

6. ¿Qué es un evento?

Los eventos son comunes en la programación visual, ya que permiten que los controles interactúen con el usuario y respondan a las acciones del usuario

7. ¿Cuál es el uso ideal de una variable de tipo Boolean?

El uso ideal de un tipo Booleana es cuando lo utilizamos para controlar el flujo de una estructura de repetición, crear interrupciones o validar condiciones

8. ¿Para qué se usa el método Parse cuando se lo utiliza por ejemplo como Decimal? Parse?

El método Parse en este caso **"Decimal.Parse"** se utiliza para las conversiones de un tipo de dato al dato seleccionado previamente del **" .Parse "**, es importante saber que también existe la función **" Convert.TIpo "**, pero tienen utilizaciones distintas y cada una interactúa de manera distinta con el dato NULL

9. ¿Cómo se declara un vector de 10 posiciones de tipo Decimal?

decimal[ ] num = new decimal[10];

10. ¿Cuándo hablamos de boxing y unboxing a que nos referimos?

Boxing, es convertir un tipo de valor a referencia para ser almacenado en un object.

UnBoxing, es convertir un tipo de referencia en un tipo de valor

Lo más relevante de estos términos es el rendimiento y como afecta en términos de sobrecarga a una aplicación o a nuestro trabajo.