

PROYECTO SABÁTICO:

23/01/2017 - 22/01/2018

5.1 MANUAL DE PRÁCTICAS

PROGRAMA 5. ELABORACIÓN DE MATERIAL DIDACTICO PARA LA ENSEÑANZA, BASADO EN EL MODELO EDUCATIVO VIGENTE.

Asignatura: **FUNDAMENTOS DE PROGRAMACIÓN**
(CLAVE: AEF-1032) INGENIERÍA INFORMÁTICA

Elaborado por: KARINA AIDEE ROMERO ALVARADO
kromero@tectijuana.edu.mx, karinaromeroalvarado@gmail.com
Enero 2018

CONTENIDO

I.	INTRODUCCIÓN	1
II.	OBJETIVO GENERAL DEL MANUAL	1
III.	FUNDAMENTOS DE PROGRAMACIÓN (CLAVE: AEF-1032) DE LA ING. INFORMÁTICA	2
	COMPETENCIA GENERAL DE LA ASIGNATURA	2
	COMPETENCIAS ESPECÍFICAS DE LA ASIGNATURA	2
	COMPETENCIAS GENÉRICAS DE LA ASIGNATURA	2
	FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS	3
	COMPETENCIAS A DESARROLLAR.	3
	PRÁCTICA 1. CREACIÓN DE UNA APLICACIÓN EN VISUAL C#	6
	PRÁCTICA 2. CALCULAR EL AREA DE UN TRIÁNGULO.	12
	PRÁCTICA 3. CALCULAR EL AREA DE UN CÍRCULO.	19
	COMPETENCIAS A DESARROLLAR	25
	PRÁCTICA 4. CONVERTIR DÓLARES A PESOS	28
	PRÁCTICA 5. SUMA DE DOS NÚMEROS	36
	HERRAMIENTAS DE PROGRAMACIÓN	39
	COMPETENCIAS A DESARROLLAR	39
	PRÁCTICA 6. CONVERTIR PESOS A DÓLARES	40
	PRÁCTICA 7. SUMA DE DOS NÚMEROS	46
	PROGRAMACIÓN ORIENTADA A OBJETOS Y MODELADO	52
	COMPETENCIAS A DESARROLLAR	52
	PRÁCTICA 8. DETERMINAR SI UNA PERSONA ES MAYOR DE EDAD UTILIZANDO LA SELECTIVA SIMPLE	53
	PRÁCTICA 9. DETERMINAR SI UNA PERSONA ES MAYOR O MENOR DE EDAD, UTILIZANDO LA SELECTIVA DOBLE.	59
	PRÁCTICA 10. SELECCIÓN DE ALUMNO PARA PARTICIPACIÓN EN CONCURSO, LOS REQUISITOS SON: PROMEDIO DE 90 O MÁS Y SER DE 5TO. SEMESTRE UTILIZANDO LA SELECTIVA COMPUESTA	65
	PRÁCTICA 11. DETERMINAR EL MAYOR DE DOS NUMEROS CONSIDERANDO SER IGUALES UTILIZANDO LA SELECTIVA ANIDADA	71
	PRÁCTICA 12. ASIGNAR EL VALOR NUMÉRICO A LAS CALIFICACIONES DADAS EN LITERAL. A=10, M=9, B=8, R=7, S=6 Y N=5 UTILIZANDO LA SELECTIVA MÚLTIPLE.	77

IMPLEMENTACIÓN ORIENTADA A OBJETOS.	83
COMPETENCIAS A DESARROLLAR.	83
PRÁCTICA 13. IMPRIME LOS NÚMEROS DEL 1 AL 10 UTILIZANDO EL CICLO DO-WHILE	84
PRÁCTICA 14. IMPRIME LOS NÚMEROS DEL 1 AL 10 UTILIZANDO EL CICLO WHILE.....	89
PRÁCTICA 15. IMPRIME LOS NÚMERO DEL 1 AL 10 UTILIZANDO EL CICLO FOR.	94
PRÁCTICA 16. PROGRAMA QUE IMPRIME LOS NÚMEROS PARES ENTRE 1 Y 40 UTILIZANDO EL CICLO FOR.	99
IMPLEMENTACIÓN ORIENTADA A OBJETOS.	104
COMPETENCIAS A DESARROLLAR.	104
PRÁCTICA 17. SUMA DE DOS NÚMEROS UTILIZANDO FUNCIONES QUE NO RETORNAN VALOR (PROCEDIMIENTO) Y PASE DE PARÁMETROS.....	105
PRÁCTICA 18. SUMA DE DOS NÚMEROS POR MEDIO DEL USO DE UNA FUNCIÓN Y PASE DE PARÁMETROS.	111
PRÁCTICA 19. SUMA DE DOS NÚMEROS UTILIZANDO FUNCIONES Y VARIABLES GLOBALES.	117
IV. CONCLUSIONES	123

I. INTRODUCCIÓN

El Tecnológico Nacional de México marca el camino hacia el cambio constante en aras de formar profesionistas que puedan resolver problemas de base tecnológica que la sociedad presenta, por lo que los estudiantes deben mantenerse actualizados y con un espíritu de superación alineado al Plan Nacional de Desarrollo de nuestro país por lo que es necesario darles las herramientas necesarias para esto.

De igual manera la industria de Tecnologías de Información de la región requiere de profesionales que tengan los conocimientos para desarrollar aplicaciones robustas que den soporte a la demanda constante de información confiable. Como institución tenemos ese gran compromiso ante la sociedad por lo que los estudiantes deben dar respuesta a dichas necesidades.

La asignatura de Fundamentos de Programación aporta al perfil del egresado, los conocimientos, habilidades, metodologías, así como capacidades de análisis y síntesis, para plantear la solución de problemas susceptibles de ser computarizados, a través de diagramas de flujo, pseudocódigo, algoritmos y el paradigma de programación orientada a objetos. La elaboración de un manual de prácticas tiene como objetivo desarrollar en los estudiantes, de Ing. Informática, esas habilidades necesarias para la solución de problemas de base tecnológica en el área de Tecnologías de Información, a través de prácticas que impulsen el logro de competencias específicas y genéricas dirigidas a la aplicación de los conceptos básicos de la programación.

II. OBJETIVO GENERAL DEL MANUAL

Desarrollar competencias específicas y genéricas dirigidas a la aplicación de los conceptos básicos de la programación, siendo capaz de escribir expresiones aritméticas y lógicas en un lenguaje de programación; así como el uso y funcionamiento de las estructuras secuenciales, selectivas y repetitivas, como también la organización de los datos, además de la declaración e implementación de funciones para construir y desarrollar aplicaciones de software que requieran dichas estructuras.

III. FUNDAMENTOS DE PROGRAMACIÓN (CLAVE: AEF-1032) DE LA ING. INFORMÁTICA

COMPETENCIA GENERAL DE LA ASIGNATURA

Analiza y soluciona problemas informáticos y representa su solución mediante herramientas de software orientado a objetos.

COMPETENCIAS ESPECÍFICAS DE LA ASIGNATURA

- Analiza la evolución de los paradigmas de programación para comprender las bases de la POO.
- Comprende los conceptos de la programación orientada a objetos, para conocer las peculiaridades de la POO frente a la programación estructurada, y su aplicación en situaciones del mundo real.
- Aplica el modelo de las 6'D para definir las etapas para solución de problemas utilizando computadora.
- Diseña diagramas UML, algoritmos y pseudocódigo empleando el paradigma de POO para la solución de problemas.
- Aplica estructuras de datos y estructuras lógicas basándose en la POO para desarrollar la lógica de programación.
- Aplica estructuras de datos y estructuras lógicas basándose en la POO para desarrollar la lógica de programación.
- Implementa métodos para diseñar módulos independientes y robustos y que correspondan a soluciones en el mundo real.

COMPETENCIAS GENÉRICAS DE LA ASIGNATURA

- Capacidad de abstracción y análisis.
- Capacidad de aplicar los conocimientos en la práctica.
- Habilidades en el uso de las tecnologías de la información y de la comunicación.
- Habilidad para buscar, procesar y analizar información procedente de fuentes diversas.
- Capacidad para identificar, plantear y resolver problemas.
- Capacidad de trabajar en equipo Y Habilidades interpersonales.
- Capacidad creativa.
- Habilidad para trabajar en forma autónoma, Compromiso ético Y Compromiso con la calidad.

FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS.

COMPETENCIAS A DESARROLLAR.

- Analiza la evolución de los paradigmas de programación para comprender las bases de la POO.
- Comprende los conceptos de la programación orientada a objetos, para conocer las peculiaridades de la POO frente a la programación estructurada, y su aplicación en situaciones del mundo real.
- Capacidad de abstracción y análisis.
- Capacidad de aplicar los conocimientos en la práctica.
- Habilidades en el uso de las tecnologías de la información y de la comunicación.
- Habilidad para buscar, procesar y analizar información procedente de fuentes diversas.
- Capacidad para identificar, plantear y resolver problemas.
- Capacidad de trabajar en equipo.
- Habilidades interpersonales.
- Capacidad creativa.
- Habilidad para trabajar en forma autónoma.
- Compromiso ético.
- Compromiso con la calidad.

FUNDAMENTO TEÓRICO DE LOS PARADIGMAS DE LA PROGRAMACIÓN

Este tema introduce al estudiante, de forma teórica, en la evolución de los paradigmas de la programación unificando la conceptualización para que sea común y se relacione con el resto del programa de forma práctica.

De acuerdo con el estilo de programación, podemos clasificar los lenguajes en las siguientes categorías: Programación Imperativa o por Procedimientos, Programación Declarativa, Programación Funcional, Lenguajes Lógicos, Programación estructurada, Programación Orientada a Objetos, Programación Visual.

De los paradigmas existentes esta asignatura hace uso de los siguientes:

Programación estructurada

La programación estructurada es un paradigma de programación orientado a mejorar la claridad, calidad y tiempo de desarrollo de un programa de computadora, utilizando únicamente subrutinas y tres estructuras: secuencia, selección (if y switch) e iteración (bucles for y while), considerando innecesario y contraproducente el uso de la instrucción de transferencia incondicional (GOTO), que podría conducir a "Código espagueti", que es mucho más difícil de seguir y de mantener, y era la causa de muchos errores de programación.

Programación Orientada a Objetos

Este paradigma algunas veces se mezcla con alguno de los otros paradigmas como la programación estructurada y visual, sin embargo mantiene características propias, que lo diferencian claramente. Los programas de este tipo, se concentran en los objetos que van a manipular, y no en la lógica requerida para manipularlos. Ejemplos de objetos pueden ser: estudiantes, coches, casas etc, cada uno de los cuales tendrá ciertas funciones (métodos) y ciertos valores que los identifican, teniendo además, la facultad de comunicarse entre ellos a través del paso de mensajes. Cabe mencionar con más detalle los elementos fundamentales que deben de poseer este tipo de lenguajes: Abstracción, Encapsulamiento, Modularidad, Jerarquía.

Programación Visual

La programación visual brinda los conocimientos necesarios para diseñar y desarrollar aplicaciones con un entorno visual amigable y fácil de utilizar para el usuario. Los lenguajes de programación visual tienden a facilitar la tarea de los programadores, dado que con los primeros lenguajes de programación crear una ventana era tarea de meses de desarrollo y de un equipo de trabajo.

PROGRAMACIÓN ORIENTADA A OBJETOS Y VISUAL

Como ya se mencionó, la POO define los programas en términos de "clases de objetos", objetos que son entidades que combinan estado (es decir, datos), comportamiento (esto es, procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto). La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas.

Las prácticas 1 y 2 buscan fortalecer la conceptualización adquirida sobre los distintos paradigmas de la programación a través de la implementación de programas en C# ya que es un lenguaje visual que incorpora la programación orientada a objetos y permite aprovechar al máximo toda la funcionalidad que ofrece este lenguaje para el desarrollo de aplicaciones de gestión.

PRÁCTICA 1. CREACIÓN DE UNA APLICACIÓN EN VISUAL C#.

OBJETIVO ESPECÍFICO.

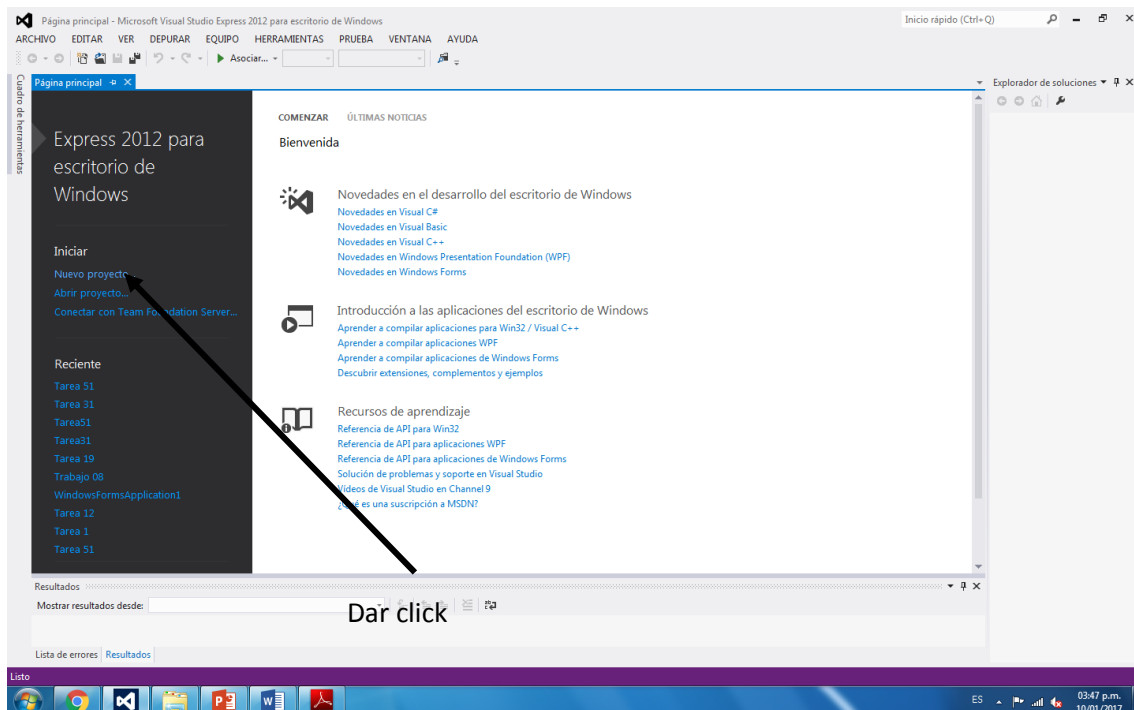
Conocer y utilizar el ambiente de programación orientado a objetos como Visual C# para la creación de una nueva aplicación en la solución de problemas del contexto.

TEMAS CON LOS QUE SE RELACIONA.

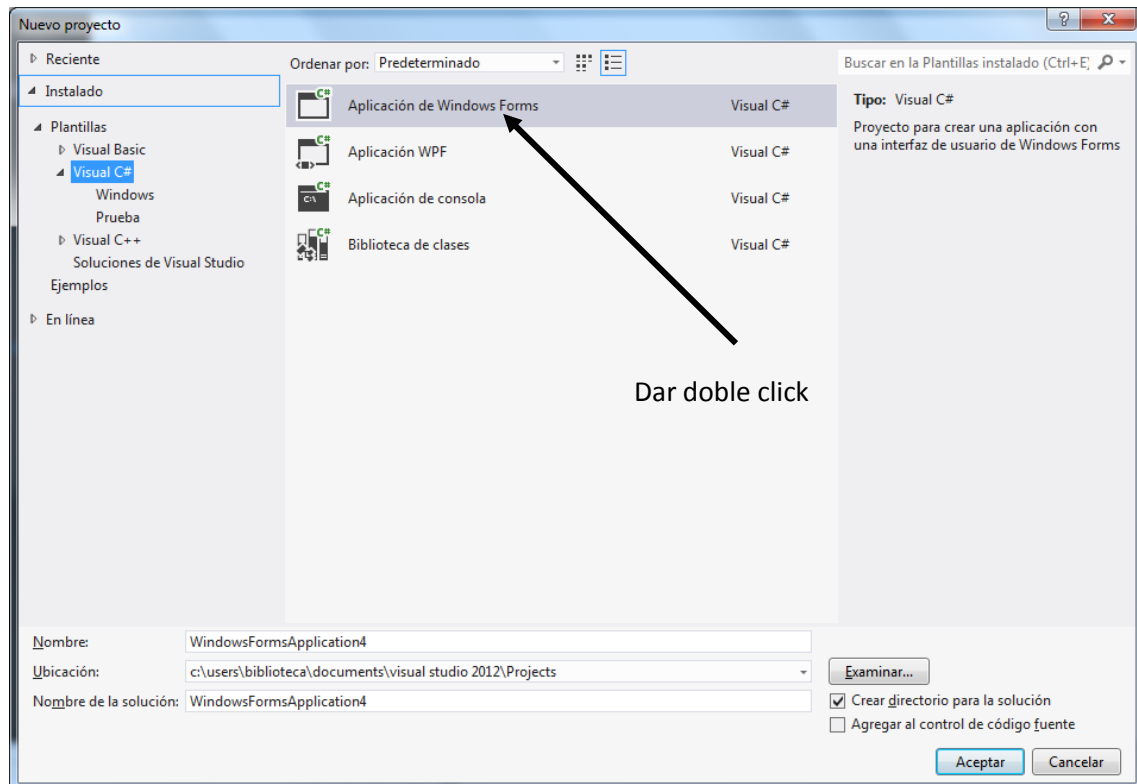
- 1.3 Lenguajes orientados a objetos.
- 1.4 Relaciones entre clases y objetos.
- 1.5 Papel de clases y objetos en el análisis y el diseño.

DESARROLLO DE LA PRÁCTICA.

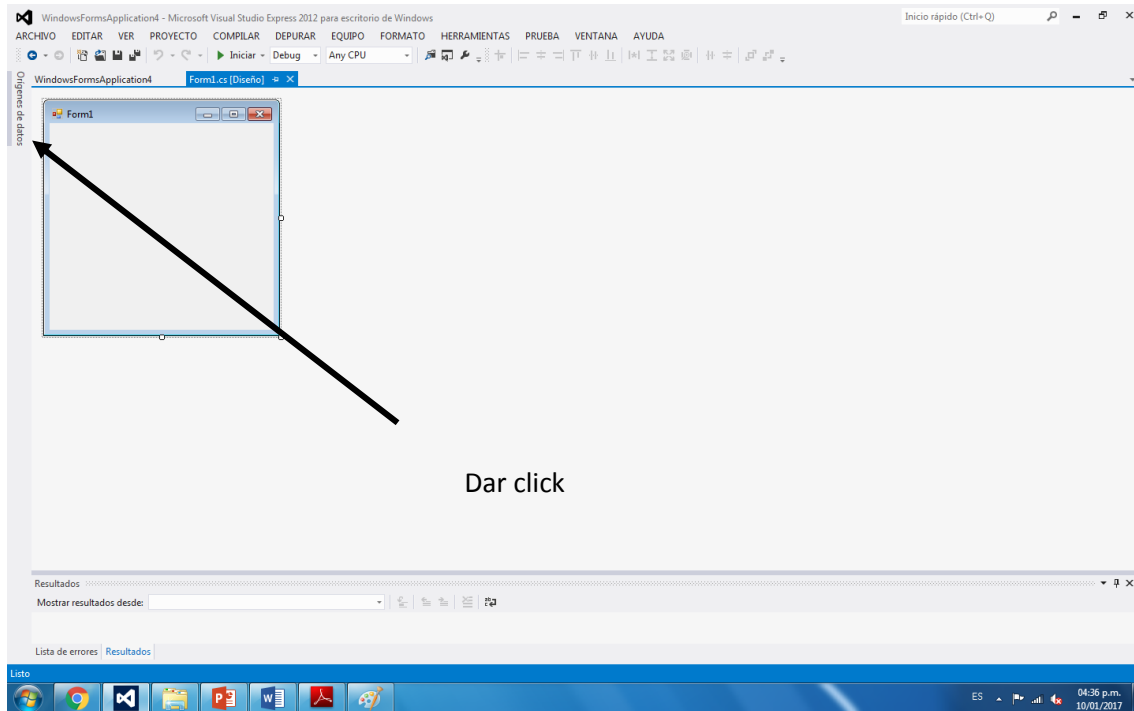
- 1.- Iniciar Visual C#
- 2.- Aparecerá la siguiente ventana
- 3.- Elegir **Nuevo Proyecto**



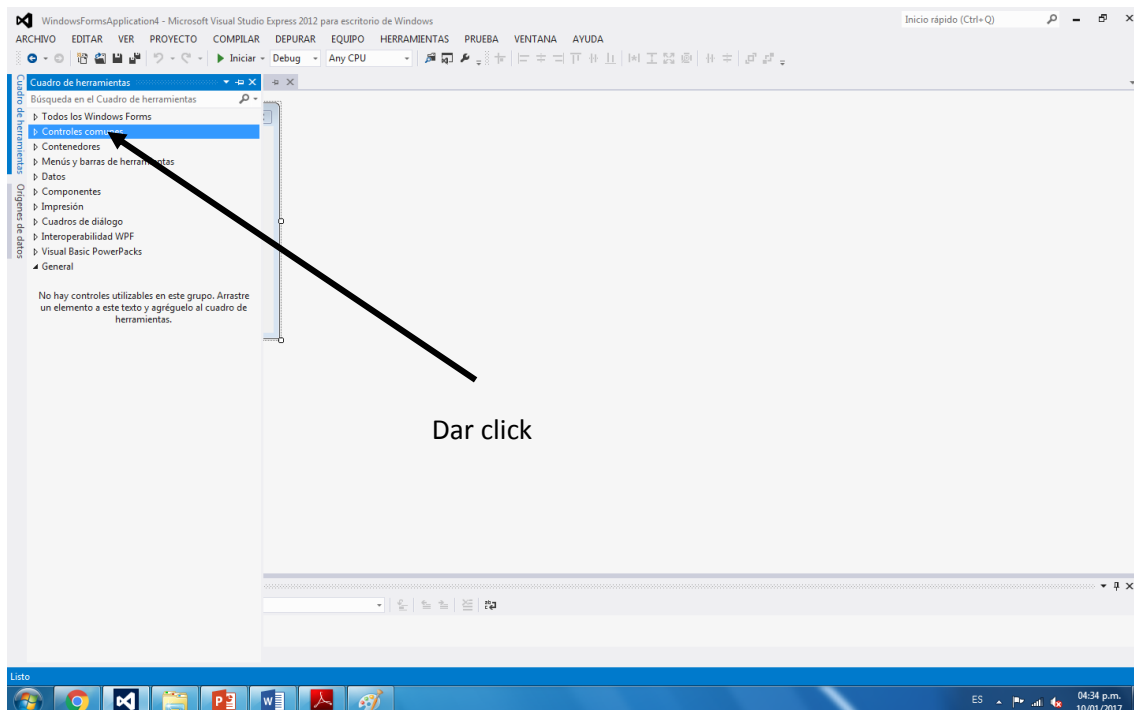
4.- Elegir **Aplicación de Windows Forms** y dar click en el botón **Aceptar**



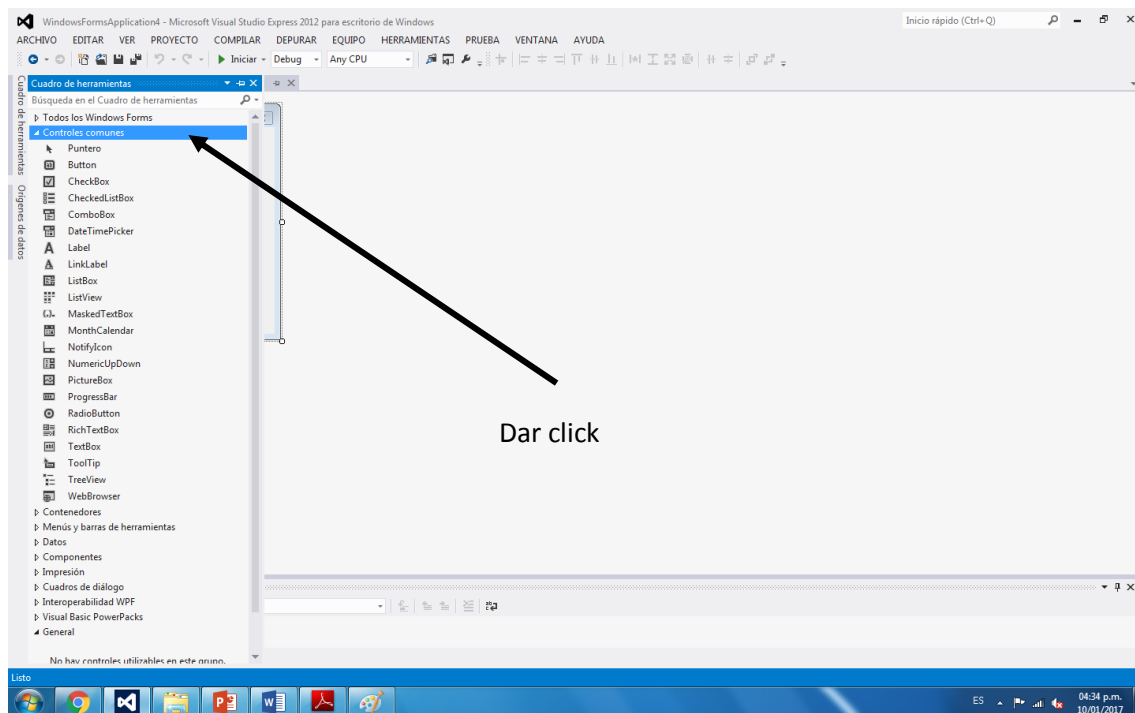
5.- Aparece la siguiente pantalla lista para iniciar la creación de la aplicación visual. Elegir la **Cuadro de herramientas** dando click en **Cuadro de herramientas** en caso de que no apareciera o elegir la opción de menú **VER** y seleccionar **Cuadro de herramientas**. Estas herramientas se utilizan para agregar objetos a la aplicación.



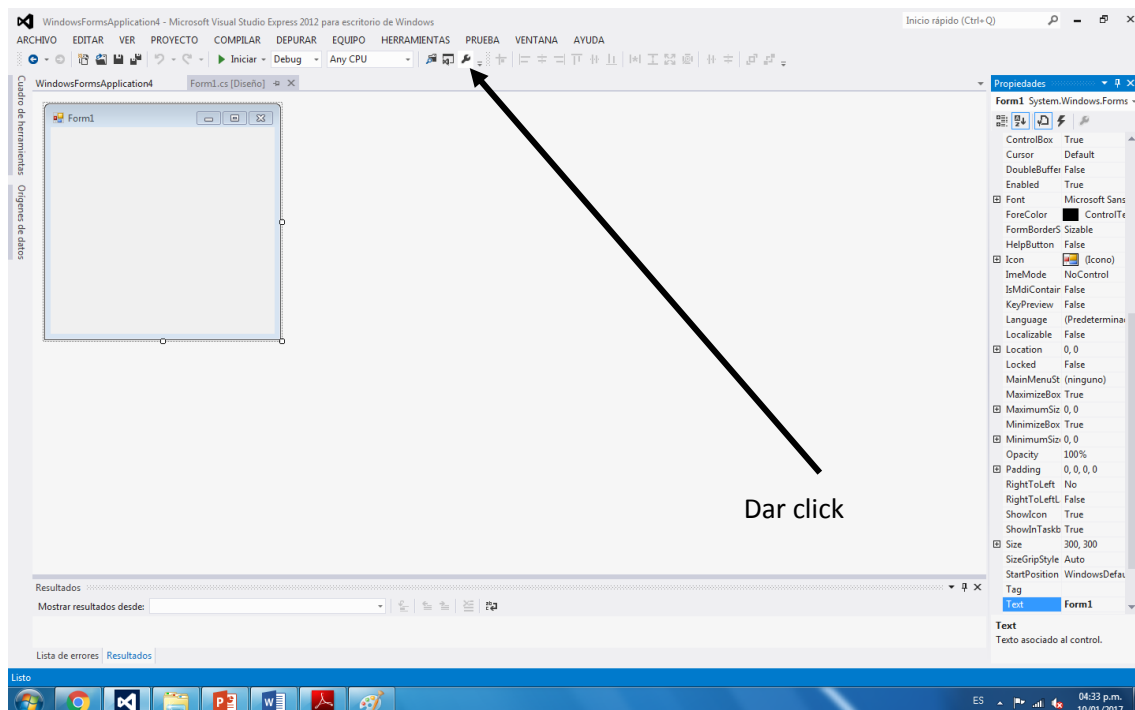
5.- Elegir **Controles comunes**



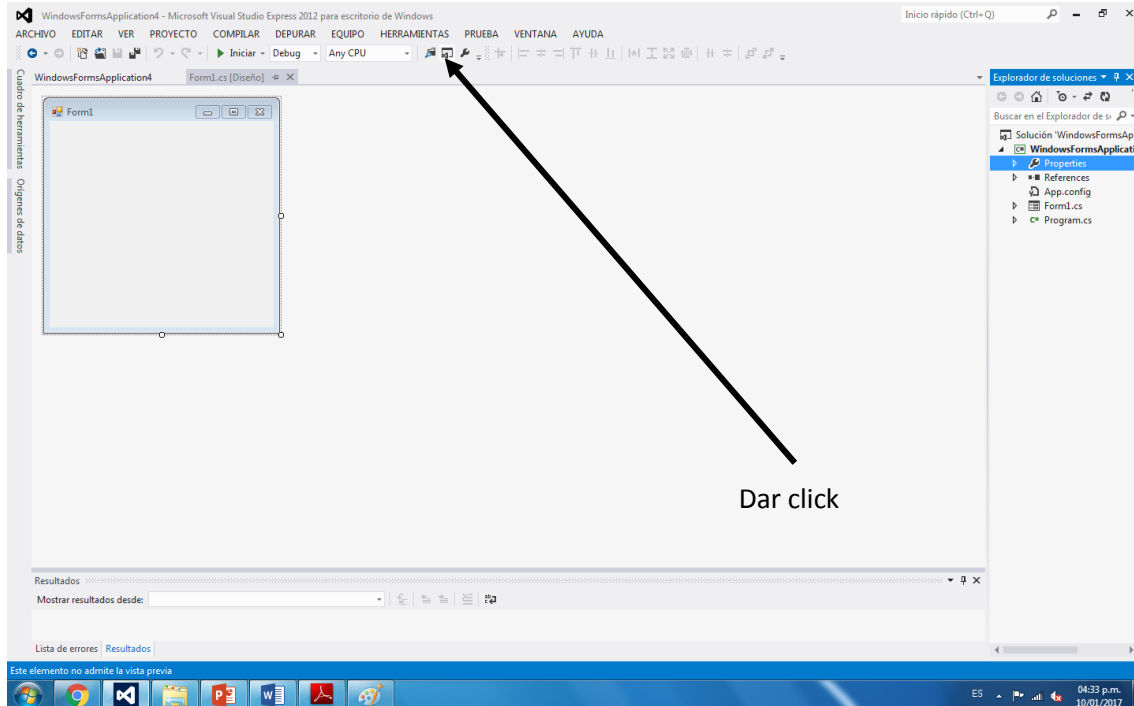
Aparece lo siguiente:



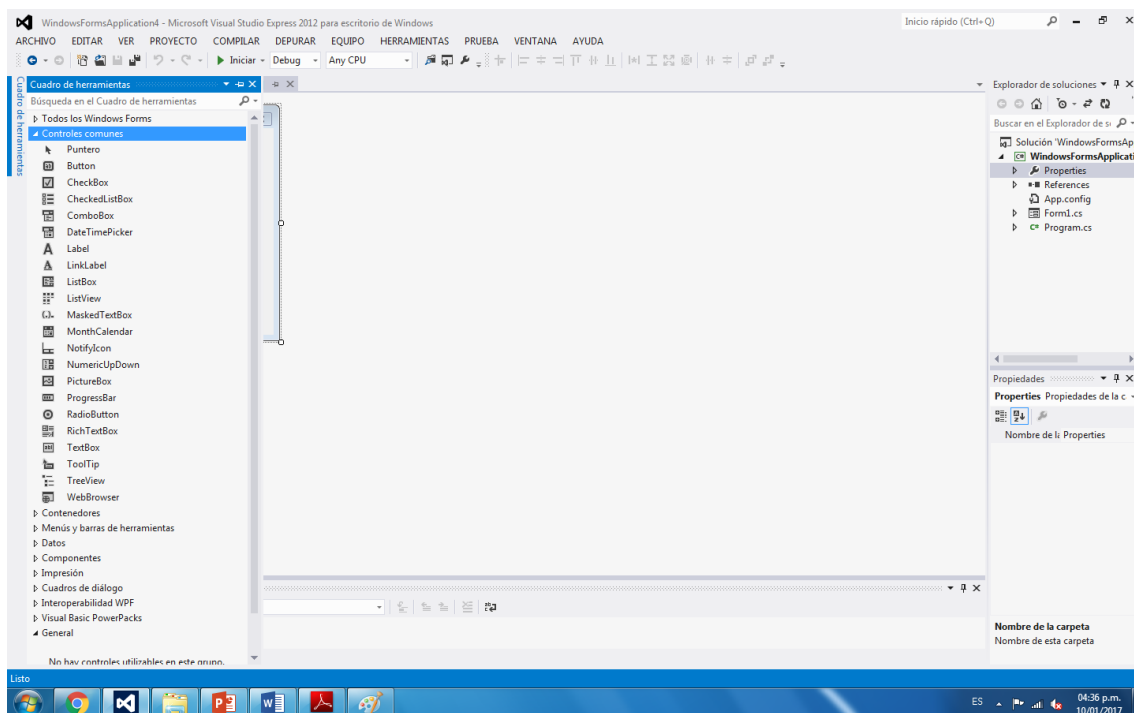
6.- Mostrar la caja de las propiedades de los componentes dando un click en  o ir a la opción del menú **VER** y elegir **Ventana propiedades** esta ventana permite, entre otras propiedades, modificar propiedades que cambian el aspecto de los objetos.



7.- En caso de que no aparezca el Explorador de soluciones dar click en  o ir a la opción del menú **VER** y elegir **Explorador de soluciones**. Este se utiliza para mostrar todos los archivos generados en el programa.



Una vez desplegado tanto el **Cuadro de herramientas**, la **Ventana propiedades** y el **Explorador de soluciones** la ventana de trabajo se verá de la siguiente manera:



Notas práctica 1:

- Dentro de la práctica 1 se vió la clase Form y una instancia de dicha clase a través del objeto Form1, que como ya se dijo, un objeto es una unidad dentro de un programa que consta de un estado (propiedades) y de un comportamiento (métodos), que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución. Un objeto puede ser creado instanciando una clase, como ocurre en la programación orientada a objetos, o mediante escritura directa de código y la replicación otros objetos.
- Una vez concluida la práctica el estudiante conoce el ambiente de programación del paradigma visual y orientado a objetos de C#.
- Se utiliza para la creación de un nuevo programa.
- Se encuentra listo para iniciar la implementación, a través de un programa, del algoritmo en Visual C#.

PRÁCTICA 2. CALCULAR EL AREA DE UN TRIÁNGULO.

OBJETIVO ESPECÍFICO.

Conocer y utilizar los operadores aritméticos para la solución de problemas del contexto por medio de algoritmos y un lenguaje orientado a objetos.

TEMAS CON LOS QUE SE RELACIONA.

- 1.3 Lenguajes orientados a objetos.
- 1.4 Relaciones entre clases y objetos.
- 1.5 Papel de clases y objetos en el análisis y el diseño.
- 3.1 Simbología.
- 3.2 Reglas para la construcción de diagramas.
- 3.3 Pseudocódigo.
- 3.4 Tipos de datos y expresiones.
- 3.5 Estructuras lógicas.

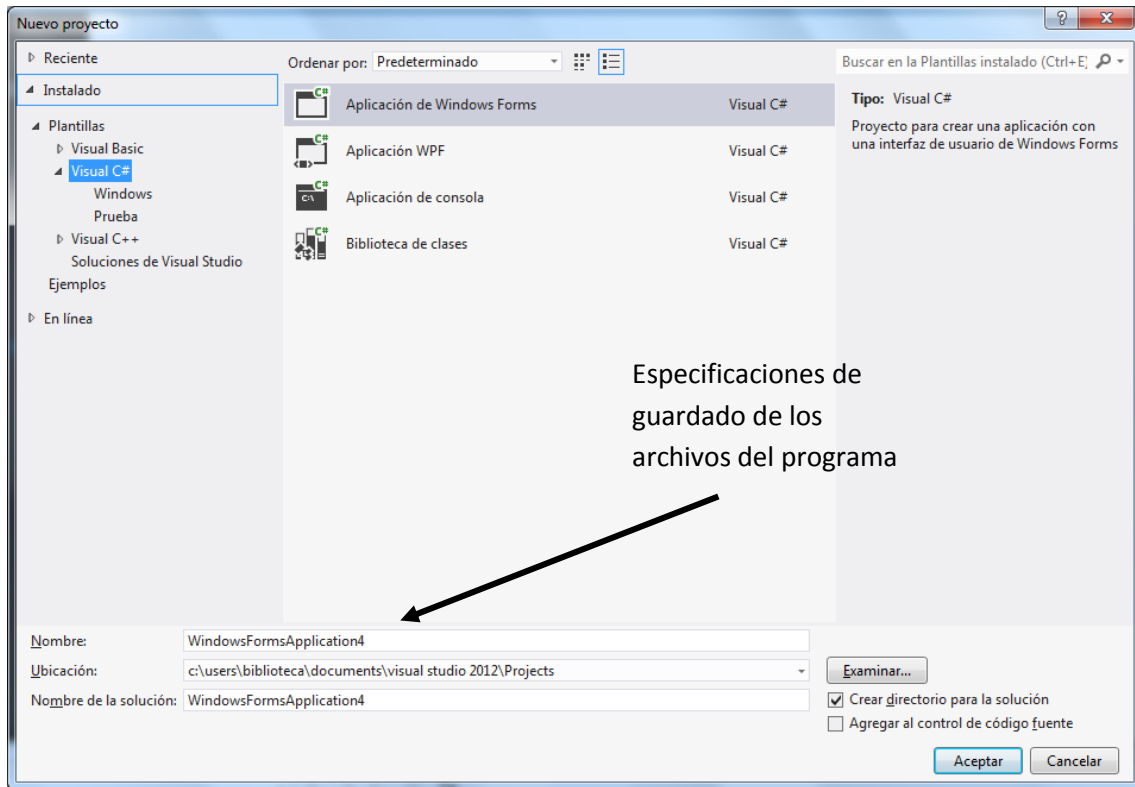
DESARROLLO DE LA PRÁCTICA.

- 1.- Iniciar Visual C#
- 2.- Elegir **Aplicación de Windows Forms**
- 3.- Elegir **Nuevo Proyecto**
- 4.- Elegir **Aplicación de Windows Forms**
- 5.- Dentro de la caja que aparece ir a la sección de Especificaciones de guardado del archivo del programa e introducir lo siguiente:

Nombre: **Práctica 2**

Ubicación: (Ir a **Examinar** y elegir ubicación, usar unidad de almacenamiento móvil)

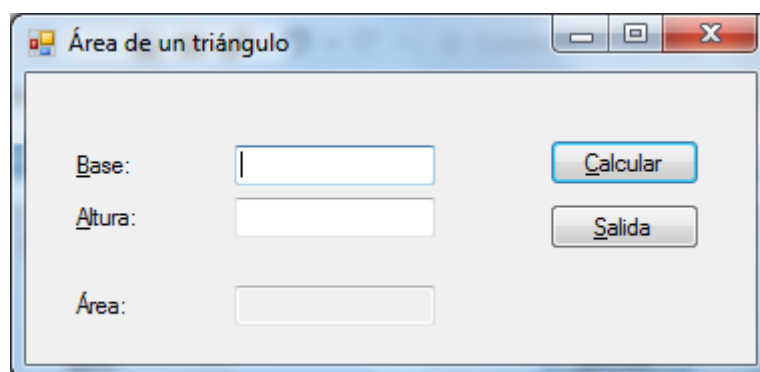
Nombre de la solución: (Automáticamente se copia el **Nombre**, dejarlo con el mismo) como se muestra a continuación:



6.- Dar click en **Cuadro de herramientas** y elegir **Controles comunes**

7.- Mostrar la **Ventana propiedades** de los componentes.

8.- Iniciar con el diseño de la forma, que deberá quedar como se muestra en la siguiente figura:



a) Detalles de la forma.

9.- Dar click en la forma e ir a la **Ventana propiedades** y localizar la propiedad **Name** y dar un click a su derecha donde dice **Form1** y sustituir por **frmPractica2** (no acepta espacios en blanco)

para así asignarle nombre a la forma. En este manual de prácticas el identificador o nombre para los objetos de tipo Forma (Form) se iniciará con el prefijo **frm**.

10.- Estando la forma seleccionada ir a la **Ventana propiedades** y localizar la propiedad **Text** y dar un click a su derecha donde dice **Form1** y sustituir por **Área de un triángulo** para así agregarle el título que aparecerá en la forma.

11.- Ir la propiedad **BackColor** y elegir un color para la forma.

12.- El tamaño de la forma se puede cambiar como cualquier ventana de Windows con arrastrar cualquier parte de su contorno.

b) Detalles de las etiquetas.

13.- Para crear la etiqueta dar click en la **Cuadro de herramientas** en **Label**  y dar un click en la forma donde se desea insertar la etiqueta.

14.- Estando seleccionada la etiqueta ir a la **Ventana propiedades** y modificar **Name** dando un click a su derecha donde dice **Label1** y sustituir por **lblBase** (no acepta espacios en blanco) para así asignarle nombre a la etiqueta. En este manual de prácticas el identificador o nombre para los objetos de tipo Etiqueta (Label) se iniciará con el prefijo **lbl**.

15.- Estando la etiqueta seleccionada ir a la **Ventana propiedades** y modificar **Text** dando un click a su derecha donde dice **Label1** y sustituir por **&Base:** para así agregarle el texto que aparecerá en la etiqueta (El & se utiliza para que la letra que le sigue se subraye y pueda ser activada con la combinación de Alt+letra al momento de la ejecución).

16.- Ir la propiedad **ForeColor** y elegir un color para la etiqueta.

17.- Ir la propiedad **Font** y elegir un Tipo de letra para la etiqueta.

c) Detalles de las cajas de texto.

18.- Para insertar una caja de texto dar click en la **Cuadro de herramientas** en  **TextBox** y dar click en la forma donde se desea insertar la caja de texto.

19.- Estando seleccionada la caja de texto ir a la **Ventana propiedades** y modificar **Name** dando un click a su derecha donde dice **textBox1** y sustituir por **txtBase** (no acepta espacios en blanco) para así asignarle nombre a la caja de texto. En este manual de prácticas el identificador o nombre para los objetos de tipo Caja de Texto (TextBox) se iniciará con el prefijo **txt**.

20.- Ir la propiedad **BackColor** y elegir un color para la caja de texto.

21.- Ir la propiedad **ForeColor** y elegir un color para el texto que posteriormente aparecerá en la caja de texto.

22.- Ir la propiedad **Font** y elegir un Tipo de letra para el texto que posteriormente aparecerá en la caja de texto.

23.- Repetir los pasos del 13 al 22 para las siguientes etiquetas y caja de texto (crear primero la etiqueta y después su correspondiente caja de texto):

Propiedades		
<i>Name</i>	<i>Text</i>	<i>ReadOnly</i>
lblAltura	&Altura:	
lblArea	Área:	
txtAltura	(dejar vacío)	False
txtArea	(dejar vacío)	True

d) Detalles de los botones de comando.

24.- Para insertar un botón de comando dar un click en la **Cuadro de herramientas** en  **Button** y dar click en la forma donde se desea insertar el botón de comando.

25.- Estando seleccionado el botón de comando ir a la **Ventana propiedades** y modificar **Name** dando un click a su derecha donde dice **Button1** y sustituir por **cmdCalcular** (no acepta espacios en blanco) para así asignarle nombre. En este manual de prácticas el identificador o nombre para los objetos de tipo Botón de Comando (Button) se iniciará con el prefijo **cmd**.

26.- Ir a la **Ventana propiedades** y modificar **Text** dando un click a su derecha donde dice **Button1** y sustituir por **&Calcular** para así agregarle el texto que aparecerá en el botón (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución).

27.- Puedes cambiar tipo de letra color, etc.

28.- Repetir los pasos 24 al 27 para el botón de salida con las siguiente propiedades: Name: **cmdSalida** y Text: **&Salida**

29. Dar un click sobre la forma e ir a la **Ventana propiedades** y seleccionar **AcceptButton**, aquí se deberá elegir de la lista el nombre del botón **cmdCalcular** para que este sea el botón que se ejecute por default, o sea, cuando se presione Enter. Posteriormente dar un click sobre la forma nuevamente e ir a la **Ventana propiedades** y seleccionar **CancelButton**, aquí se deberá elegir de la lista el nombre del botón **cmdSalida** para que por medio de la tecla Esc se cierre el programa.

e) Introducir código.

30.- Dar doble click sobre el botón **cmdCalcular** y el cursor aparecerá dentro de las llaves del siguiente subprograma:

```
private void cmdCalcular_Click(object sender, EventArgs e)
{

}
}
```

31.- Sin mover el cursor introducir el siguiente código:

```
// Programa que calcula el área de un triángulo
// Práctica 2

// Declaración de variables
float Base, Altura, Area

// Asignación de valores a variables
Base = System.Single.Parse(txtBase.Text);
Altura = System.Single.Parse(txtAltura.Text);

// Calculo área
Area = Base * Altura / 2;

// Salida resultado
txtArea.Text = Area.ToString();
```

32.- Para regresar al diseño de la forma ir a la caja **Explorador de soluciones** y dar doble click sobre Form1.cs o ir a la opción del menú **VER** y elegir **Diseñador**.

33.- Dar doble click sobre el botón **cmdSalida** y el cursor aparecerá dentro de las llaves del siguiente subprograma:

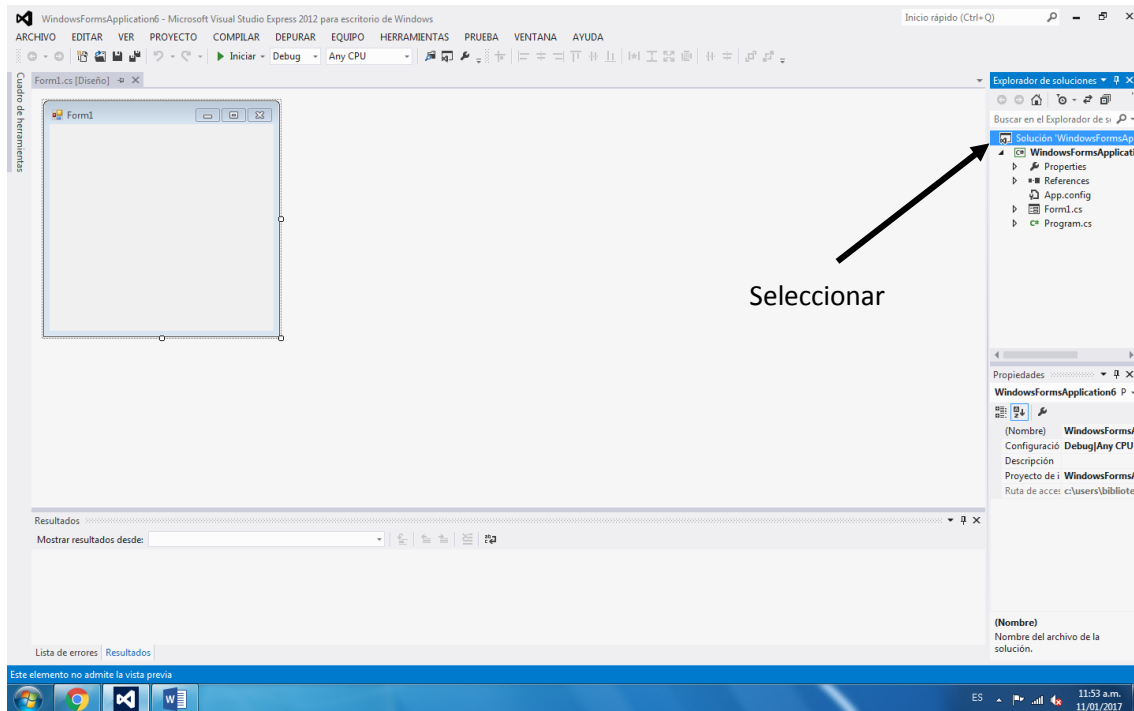
```
private void cmdSalida_Click(object sender, EventArgs e)
{

}
}
```

34.- Sin mover el cursor introducir el siguiente código:

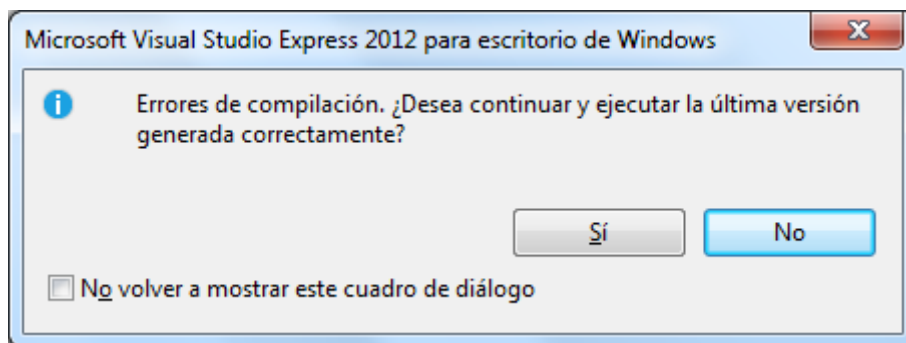
```
//Salida del programa
Close();
```

35.- Guardar el programa **Práctica 2**. En caso de que no hayas especificado desde el inicio el nombre y ubicación del programa ir a la opción del menú **ARCHIVO** y posteriormente ir a **Guardar WindowsFormsApplication1 como**. Asegurarse que en el **Explorador de soluciones** se encuentre elegido **WindowsFormsApplication1** para guardar correctamente.

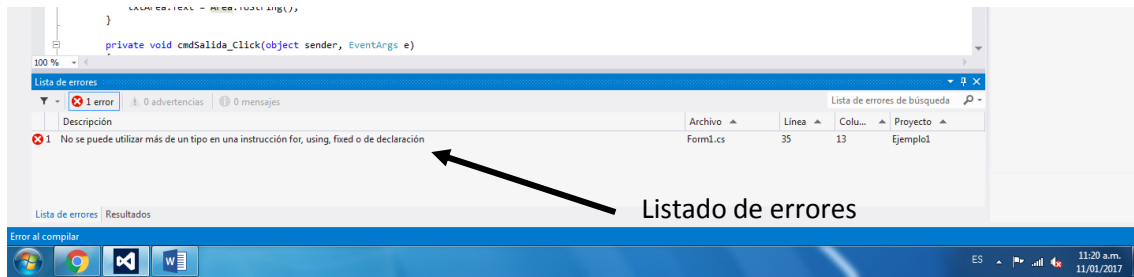


36.- Para compilar y ejecutar el programa dar click en , presionar F5 o ir a la opción del menú **DEPURAR** y elegir **Iniciar depuración**.

37.- En caso que no se ejecute quiere decir que se presentaron errores durante la compilación o traducción y mostrará la siguiente caja:



Dar click en **No** para que permita revisar e identificar los errores de sintaxis y así poder corregirlos. Mostrará una división en la ventana donde se enlistan los errores (los errores varían):



Notas práctica 2:

- Dentro de la práctica 2 se vieron las clases Form, Label, Text, Button y las instancias de dichas clases a través de los objetos frmPractica2, lblBase, lblAltura, lblArea, txtBase, txtAltura, txtAArea, cmdCalcular, cmdSalida, respectivamente, que como ya se dijo, un objeto es una unidad dentro de un programa que consta de un estado (propiedades) y de un comportamiento (métodos), que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.
- Se modificaron las propiedades de Name para los objetos usando prefijos de acuerdo a la clase: Form (prefijo frm), Label (prefijo lbl), Text (prefijo txt), Button (prefijo cmd).
- Se ligaron métodos a los objetos de la clase Button (cmdCalcular, cmdSalida).
- Esta práctica introduce la declaración de variables de tipo real **float**.
- Las **//** se utilizan para dar inicio a un comentario dentro del código del programa. Estos comentarios no afectan el funcionamiento del mismo.

// Programa que calcula el área de un triángulo

- El objeto **Text Box** permite la entrada de datos que posteriormente se le asignarán a las variables correspondientes. Como la cajas de texto almacenan precisamente texto, es necesario convertir los caracteres numéricos a valores numéricos, en este caso como la variable que recibe es tipo **float** su convertidor es **Single**:

```
Base = System.Single.Parse(txtBase.Text);
```

- Por otro lado al momento de mostrar un valor numérico en una caja de texto es necesario convertirlo a cadena:

```
txtArea.Text = Area.ToString();
```

PRÁCTICA 3. CALCULAR EL AREA DE UN CÍRCULO.

OBJETIVO ESPECÍFICO.

Conocer y utilizar los operadores aritméticos para la solución de problemas del contexto por medio de algoritmos y un lenguaje orientado a objetos.

TEMAS CON LOS QUE SE RELACIONA.

- 1.3 Lenguajes orientados a objetos.
- 1.4 Relaciones entre clases y objetos.
- 1.5 Papel de clases y objetos en el análisis y el diseño.
- 3.1 Simbología.
- 3.2 Reglas para la construcción de diagramas.
- 3.3 Pseudocódigo.
- 3.4 Tipos de datos y expresiones.
- 3.5 Estructuras lógicas.

DESARROLLO DE LA PRÁCTICA.

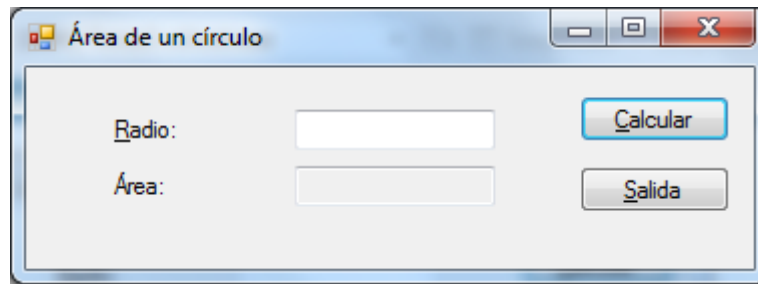
- 1.- Iniciar Visual C#
- 2.- Elegir **Aplicación de Windows Forms**
- 3.- Elegir **Nuevo Proyecto**
- 4.- Elegir **Aplicación de Windows Forms**
- 5.- Dentro de la caja que aparece ir a la sección de Especificaciones de guardado del archivo del programa e introducir lo siguiente:

Nombre: **Práctica 3**

Ubicación: (Ir a **Examinar** y elegir ubicación, usar unidad de almacenamiento móvil)

Nombre de la solución: (Automáticamente se copia el **Nombre**, dejarlo con el mismo)

- 6.- Diseñar la forma, que deberá quedar como se muestra en la siguiente figura:



a) Detalles de la forma.

7.- Dar click en la forma e ir a la **Ventana propiedades**, localizar la propiedad **Name** y dar un click a su derecha donde dice **Form1** y sustituir por **frmPractica3** (no acepta espacios en blanco) para así asignarle nombre a la forma.

8.- Estando la forma seleccionada ir a la **Ventana propiedades**, localizar la propiedad **Text** y dar un click a su derecha donde dice **Form1** y sustituir por **Área de un círculo** para así agregarle el título que aparecerá en la forma.

9.- Ir la propiedad **BackColor** y elegir un color para la forma.

10.- El tamaño de la forma se puede cambiar como cualquier ventana de Windows con arrastrar cualquier parte de su contorno.

b) Detalles de las etiquetas.

11.- Para crear la etiqueta dar click en la **Cuadro de herramientas** en **Label**  y dar un click en la forma donde se desea insertar la etiqueta.

12.- Estando seleccionada la etiqueta ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Label1** y sustituir por **lblRadio** (no acepta espacios en blanco) para así asignarle nombre a la etiqueta.

13.- Estando la etiqueta seleccionada ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Label1** y sustituir por **&Radio:** para así agregarle el texto que aparecerá en la etiqueta (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución).

14.- Ir la propiedad **ForeColor** y elegir un color para la etiqueta.

15.- Ir la propiedad **Font** y elegir un Tipo de letra para la etiqueta.

c) Detalles de las cajas de texto.

16.- Para insertar una caja de texto dar click en la **Cuadro de herramientas** en  **Text Box** y dar click en la forma donde se desea insertar la caja de texto.

17.- Estando seleccionada la caja de texto ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **textBox1** y sustituir por **txtRadio** (no acepta espacios en blanco) para así asignarle nombre a la caja de texto.

18.- Ir la propiedad **BackColor** y elegir un color para la caja de texto.

19.- Ir la propiedad **ForeColor** y elegir un color para el texto que posteriormente aparecerá en la caja de texto.

20.- Ir la propiedad **Font** y elegir un Tipo de letra para el texto que posteriormente aparecerá en la caja de texto.

21.- Repetir los pasos del 11 al 20 para las siguientes etiquetas y caja de texto (crear primero la etiqueta y después su correspondiente caja de texto):

Propiedades		
<i>Name</i>	<i>Text</i>	<i>ReadOnly</i>
lblArea	Área:	
txtArea	(dejar vacío)	True

d) Detalles de los botones de comando.

22.- Para insertar un botón de comando dar un click en la **Cuadro de herramientas** en  **Button** y dar click en la forma donde se desea insertar el botón de comando.

23.- Estando seleccionado el botón de comando ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Button1** y sustituir por **cmdCalcular** (no acepta espacios en blanco) para así asignarle nombre.

24.- Ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Button1** y sustituir por **&Calcular** para así agregarle el texto que aparecerá en el botón (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución). Puedes cambiar tipo de letra color, etc.

25.- Agregar botón de comando repitiendo los pasos 22 al 24 para la salida del programa dejando que también se active con la tecla **Esc**, así como establecer el botón que se ejecutará al presionar la tecla **Enter**, para lo que se modificarán dos propiedades de la forma **frmPractica3**. Utilizar las siguientes propiedades:

Propiedades			
<i>Name</i>	<i>Text</i>	<i>AcceptButton</i>	<i>CancelButton</i>
cmdSalida	&Salida	-	-
frmPractica3	-	cmdCalcular	cmdSalida

e) Introducir código.

26.- Dar doble click sobre el botón **cmdCalcular** y el cursor aparecerá dentro de las llaves del siguiente subprograma:

```
private void cmdCalcular_Click(object sender, EventArgs e)
{

}

}
```

27.- Sin mover el cursor introducir el siguiente código:

```
//Programa que calcula el área de un círculo
//Práctica 3

//Declaración de variables
double Radio, Area;

//Asignación de valores a las variables
Radio = System.Double.Parse(txtRadio.Text);

//Calculo del área
Area = 3.1416 * Math.Pow(Radio, 2);

//Mostrar resultado
txtArea.Text = Area.ToString();
```

28.- Para regresar al diseño de la forma ir a la caja **Explorador de soluciones** y dar doble click sobre Form1.cs o ir a la opción del menú **VER** y elegir **Diseñador**.

29.- Dar doble click sobre el botón **cmdSalida** y el cursor aparecerá dentro de las llaves del siguiente subprograma:

```
private void cmdSalida_Click(object sender, EventArgs e)
{

}

}
```

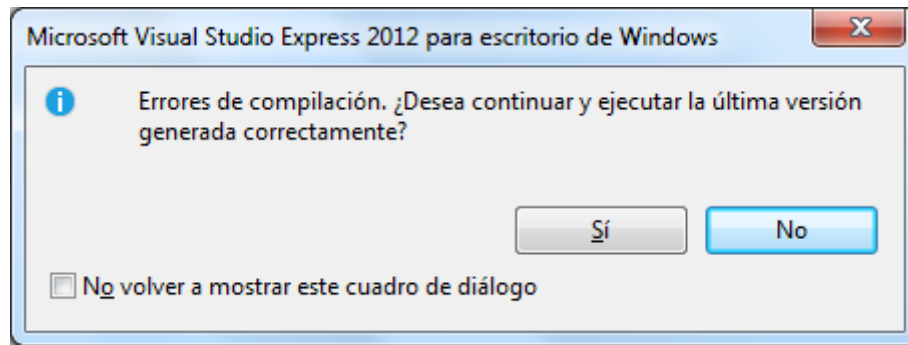
30.- Sin mover el cursor introducir el siguiente código:

```
//Salida del programa
Close();
```

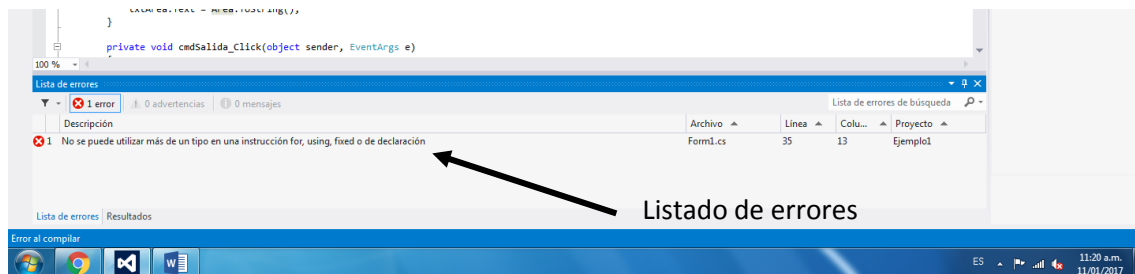
31.- Guardar el programa **Práctica 3**. En caso de que no hayas especificado desde el inicio el nombre y ubicación del programa ir a la opción del menú **ARCHIVO** y posteriormente ir a **Guardar WindowsFormsApplication1 como**. Asegurarse que en el **Explorador de soluciones** se encuentre elegido **WindowsFormsApplication1** para guardar correctamente.

32.- Para compilar y ejecutar el programa dar click en , presionar F5 o ir a la opción del menú **DEPURAR** y elegir **Iniciar depuración**.

33.- En caso que no se ejecute quiere decir que se presentaron errores durante la compilación o traducción y mostrará la siguiente caja:



Dar click en **No** para que permita revisar e identificar los errores de sintaxis y así poder corregirlos. Mostrará una división en la ventana donde se enlistan los errores (los errores varían):



Notas práctica 3:

- Dentro de la práctica 3 se vieron las clases Form, Label, Text, Button y las instancias de dichas clases a través de los objetos frmPractica3, lblRadio, lblArea, txtRadio, txtArea, cmdCalcular, cmdSalida, respectivamente, que como ya se dijo, un objeto es una unidad dentro de un programa que consta de un estado (propiedades) y de un comportamiento (métodos), que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.
- Se modificaron las propiedades de Name para los objetos usando prefijos de acuerdo a la clase: Form (prefijo frm), Label (prefijo lbl), Text (prefijo txt), Button (prefijo cmd).
- Se ligaron métodos a los objetos de la clase Button (cmdCalcular, cmdSalida).
- En este ejercicio se introduce un nuevo tipo de dato numérico: **double**
- Las **//** se utilizan para dar inicio a un comentario dentro del código del programa. Estos comentarios no afectan el funcionamiento del mismo.

```
// Programa que calcula el área de un círculo
```

- Al estar trabajando en forma visual el objeto **Text Box** permite la entrada de datos que posteriormente se le asignarán a las variables correspondientes. Como las cajas de texto almacenan precisamente texto, es necesario convertir los caracteres numéricos a valores numéricos, en este caso como la variable que recibe es tipo **double** su convertidor es **Double**:

```
Radio = System.Double.Parse(txtRadio.Text);
```

- Para elevar a cualquier potencia se utiliza **Math.Pow();** (la clase Math y el método Pow()) el cual requiere que el tipo de dato sea **double**. Aquí la variable **Radio** que almacena el radio del círculo es elevado al cuadrado al indicar el **2**.

```
Area = 3.1416 * Math.Pow(Radio, 2);
```

METODOLOGÍA DE SOLUCIÓN DE PROBLEMAS

COMPETENCIAS A DESARROLLAR

- Aplicar el modelo de las 6'D para definir las etapas para solución de problemas utilizando computadora.
- Capacidad de abstracción y análisis.
- Capacidad de aplicar los conocimientos en la práctica.
- Habilidades en el uso de las tecnologías de la información y de la comunicación.
- Habilidad para buscar, procesar y analizar información procedente de fuentes diversas.
- Capacidad para identificar, plantear y resolver problemas.
- Capacidad de trabajar en equipo.
- Habilidades interpersonales.
- Capacidad creativa.
- Habilidad para trabajar en forma autónoma.
- Compromiso ético.

BASE TEÓRICA DEL MODELO DE LAS 6'D

Este tema introduce al estudiante al uso del modelo de las 6'D. Se describen sus etapas, así como los pasos y resultados que se deben esperar de cada una de ellas relacionándolo con el modelo de objetos. Las 6 etapas del Modelo 6'D son las siguientes:

Etapas 1: Descripción del problema (enunciado).

En esta etapa se identifica y comprende el problema que se desea resolver y se sabe cuál es el resultado al que se quiere llegar.

- Identificación del problema
- Descripción del problema
- Enunciado claro y preciso del problema

El resultado de esta etapa es un ENUNCIADO claro del problema a solucionar.

Etapas 2: Definición de solución (especificaciones).

En esta etapa se analiza a fondo el problema para saber exactamente en qué consiste y así poder descomponerlo en cada una de sus partes facilitando su comprensión y posterior solución. Una vez hecho el análisis se plantean diversas alternativas que permitan solucionar el problema y así seleccionar la más adecuada.

- Definir el resultado deseado.
- Determinar los datos a ingresar o generar para obtener el resultado deseado.
- Determinar la forma de procesamiento de los datos para transformarlos en información.

El resultado de esta etapa son las ESPECIFICACIONES para la solución del problema.

Etapas 3: Diseño de la solución (modelado).

Esta etapa contempla el diseño de la solución a través del modelado y desarrollo de algoritmos. Para el modelado de la solución del problema en la Programación Orientada a Objetos se hace uso del Lenguaje Unificado de Modelado (UML) por medio del cual se describen las clases, objetos y sus relaciones. De la misma manera se hace uso de los algoritmos a través de diagramas de flujo para diseñar los métodos de una clase.

- Definir el problema para el proyecto.
- Definición de diagramas, relaciones y clases.
- Desarrollo de algoritmos.

El resultado de esta etapa es el MODELADO (diagramas y algoritmos) que especifican la solución del problema.

Etapas 4: Desarrollo de la solución (codificación).

En esta etapa se lleva a cabo la implementación de la solución del problema en un lenguaje de programación orientado a objetos que involucra traducir los diagramas y las especificaciones de UML a dicho lenguaje generando un programa que se almacenará en un archivo lógico que la computadora podrá ejecutar.

- Codificar el proyecto en un lenguaje de programación.
- Desarrollar comentarios internos en los programas de computadoras.
- Desarrollar copias de seguridad de los programas de computadoras.

El resultado de esta etapa es la CODIFICACIÓN (programa implementado en un lenguaje de programación) que permite solucionar el problema.

Etapas 5: Depuración y pruebas (pruebas).

Después de la codificación del programa éste debe ser probado mediante su ejecución conocida como corrida del programa. Durante esta etapa pueden surgir errores de lógica y sintaxis que deberán ser corregidos ya sea modificándolo o creando nuevas sentencias. Una vez hechas las correcciones correspondientes se vuelve a probar hasta conseguir el resultado deseado.

- Realizar la depuración y verificar la correcta estructura de programas.
- Realizar pruebas de sintaxis.
- Realizar pruebas de lógica.

El resultado de esta etapa son las PRUEBAS hasta que registren el adecuado funcionamiento de solución del problema.

Etapas 6: Documentación (manuales).

Finalmente en esta etapa se recopila toda la información generada en las etapas anteriores que sirve como base para la elaboración del manual técnico.

- Recopilar el material generado de cada una de las etapas anteriores.
- Generar el manual del programa
- Generar el manual de usuario

El resultado de esta etapa es el MANUAL que permite un adecuado manejo de la solución desarrollada. El manual técnico debe incluir:

- 1.- Descripción del problema.
- 2.- Resultados esperados y datos necesarios para generar dichos resultados.
- 3.- Diagramas UML y algoritmos.
- 4.- Pruebas desarrolladas y el listado del programa con comentarios internos.

PRÁCTICA 4. CONVERTIR DÓLARES A PESOS.

OBJETIVO ESPECÍFICO.

Conocer y utilizar los operadores aritméticos para la solución de problemas del contexto utilizando el modelo de las 6'D para definir las etapas para solución de problemas, algoritmos y un lenguaje orientado a objetos aplicando la Metodología para la solución de problemas.

TEMAS CON LOS QUE SE RELACIONA.

- 2.1 Descripción del problema (enunciado).
- 2.2 Definición de solución (especificaciones).
- 2.3 Diseño de la solución (modelado).
- 2.4 Desarrollo de la solución (codificación).
- 2.5 Depuración y pruebas (pruebas).
- 2.6 Documentación (manuales).

DESARROLLO DE LA PRÁCTICA.

- a) Analizar la descripción de la siguiente situación que se desea resolver a través del desarrollo de un programa de cómputo.

1. *La casa de cambio “Los Aces, S.A. de C.V.” se encuentra localizada en Tijuana y desea abrir sus puertas ofreciendo al cliente un servicio confiable en la conversión de distintas monedas a pesos mexicanos. Sus operaciones iniciales están enfocadas en la captación de dólares únicamente a un tipo de cambio muy competitivo. Al encontrarse en un mercado cambiario fluctuante dicha casa de cambio debe adaptarse a esa situación para evitar confusiones al momento de estar en operaciones. A los socios de la casa de cambio le interesa tener la flexibilidad para poder modificar el tipo de cambio de acuerdo al comportamiento del mercado cambiario y a los clientes únicamente les interesa saber cuánto dinero van a recibir de la cantidad que ellos desean cambiar.*

- b) Una vez analizado el planteamiento de la situación anterior y llevar a cabo la identificación y descripción del problema generando un enunciado claro que lo especifique.

2. Identifica el problema: _____

3. Descripción del problema: _____

4. Anota claramente y con precisión el enunciado del problema: _____

c) Partiendo del enunciado del problema llevar a cabo un análisis o a fondo para saber exactamente en qué consiste y así poder descomponerlo en cada una de sus partes facilitando su comprensión y posterior solución.

5. Utilizar el Modelo de Solución para el análisis del problema donde se desglosan tres partes muy importantes en la solución del problema: DATOS DE ENTRADA (Datos que se necesitan para resolver el problema), PROCESO (Procesos para solucionar el problema, por ejemplo los cálculos, etc.) y DATOS DE SALIDA (Datos que serán la salida, o sea, los resultados). Utilizar todos los modelos necesarios para cada alternativa de solución.

DATOS DE ENTRADA	PROCESOS	DATOS DE SALIDA

6. Identificar el resultado deseado: _____

7. Identificar claramente los datos a ingresar y transformar así como aquellos datos a generar: _____

8. Identificar claramente los procesos necesarios para transformar los datos en información y así lograr llegar al resultado esperado: _____

9. Identificar alternativas que permitan solucionar el problema: _____

10. Seleccionar la alternativa más adecuada para solucionar el problema : _____

d) Partiendo del enunciado claro y preciso del problema a solucionar diseñar la solución a través del modelado utilizando UML y desarrollo de algoritmos.

11. Diagramas UML

12. Diagrama del algoritmo

e) Una vez diseñada la solución se deberá implementar en un lenguaje de programación o se puede desarrollar la siguiente alternativa de solución si se desea

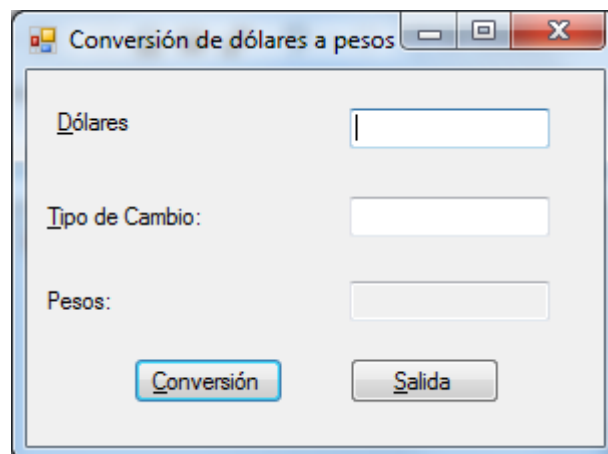
- 1.- Diseñar diagramas UML, algoritmo y posteriormente iniciar Visual C#
- 2.- Elegir **Aplicación de Windows Forms**
- 3.- Elegir **Nuevo Proyecto**
- 4.- Elegir **Aplicación de Windows Forms**
- 5.- Dentro de la caja que aparece ir a la sección de Especificaciones de guardado del archivo del programa e introducir lo siguiente:

Nombre: **Práctica 4**

Ubicación: (Ir a **Examinar** y elegir ubicación, usar unidad de almacenamiento móvil)

Nombre de la solución: (Automáticamente se copia el **Nombre**, dejarlo con el mismo)

- 6.- Diseñar la forma, que deberá quedar como se muestra en la siguiente figura:

The image shows a screenshot of a Windows Forms application window. The title bar of the window reads "Conversión de dólares a pesos". Inside the window, there are three text input fields arranged vertically. The first field is labeled "Dólares", the second is labeled "Tipo de Cambio:", and the third is labeled "Pesos:". Below these fields, there are two buttons. The left button is labeled "Conversión" and the right button is labeled "Salida". The window has a standard Windows XP-style border with minimize, maximize, and close buttons in the title bar.

- I) Detalles de la forma.

- 7.- Dar click en la forma e ir a la **Ventana propiedades**, localizar la propiedad **Name** y dar un click a su derecha donde dice **Form1** y sustituir por **frmPractica4** (no acepta espacios en blanco) para así asignarle nombre a la forma.
- 8.- Estando la forma seleccionada ir a la **Ventana propiedades**, localizar la propiedad **Text** y dar un click a su derecha donde dice **Form1** y sustituir por **Conversión de dólares a pesos** para así agregarle el título que aparecerá en la forma.
- 9.- Ir la propiedad **BackColor** y elegir un color para la forma.
- 10.- El tamaño de la forma se puede cambiar como cualquier ventana de Windows con arrastrar cualquier parte de su contorno.

II) Detalles de las etiquetas.

11.- Para crear la etiqueta dar click en la **Cuadro de herramientas** en **Label**  y dar un click en la forma donde se desea insertar la etiqueta.

12.- Estando seleccionada la etiqueta ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Label1** y sustituir por **lblDolares** (no acepta espacios en blanco) para así asignarle nombre a la etiqueta.

13.- Estando la etiqueta seleccionada ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Label1** y sustituir por **&Dólares** para así agregarle el texto que aparecerá en la etiqueta (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución).

14.- Ir la propiedad **ForeColor** y elegir un color para la etiqueta.

15.- Ir la propiedad **Font** y elegir un Tipo de letra para la etiqueta.

III) Detalles de las cajas de texto.

16.- Para insertar una caja de texto dar click en la **Cuadro de herramientas** en  **Text Box** y dar click en la forma donde se desea insertar la caja de texto.

17.- Estando seleccionada la caja de texto ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **textBox1** y sustituir por **txtDolares** (no acepta espacios en blanco) para así asignarle nombre a la caja de texto.

18.- Ir la propiedad **BackColor** y elegir un color para la caja de texto.

19.- Ir la propiedad **ForeColor** y elegir un color para el texto que posteriormente aparecerá en la caja de texto.

20.- Ir la propiedad **Font** y elegir un Tipo de letra para el texto que posteriormente aparecerá en la caja de texto.

21.- Repetir los pasos del 11 al 20 para las siguientes etiquetas y caja de texto (crear primero la etiqueta y después su correspondiente caja de texto):

Propiedades		
<i>Name</i>	<i>Text</i>	<i>ReadOnly</i>
lblTC	&Tipo de cambio:	-
lblPesos	Pesos:	-
txtTC	(dejar vacío)	False
txtPesos	(dejar vacío)	True

IV) Detalles de los botones de comando.

22.- Para insertar un botón de comando dar un click en la **Cuadro de herramientas** en  **Button** y dar click en la forma donde se desea insertar el botón de comando.

23.- Estando seleccionado el botón de comando ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Button1** y sustituir por **cmdConversion** (no acepta espacios en blanco) para así asignarle nombre.

24.- Ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Button1** y sustituir por **&Conversión** para así agregarle el texto que aparecerá en el botón (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución). Puedes cambiar tipo de letra color, etc.

25.- Agregar botón de comando repitiendo los pasos 22 al 24 para la salida del programa dejando que también se active con la tecla **Esc**, así como establecer el botón que se ejecutará al presionar la tecla **Enter**, para lo que se modificarán dos propiedades de la forma **frmPractica4**. Utilizar las siguientes propiedades:

Propiedades			
Name	Text	AcceptButton	CancelButton
cmdSalida	&Salida	-	-
frmPractica4	-	cmdConversion	cmdSalida

V) Introducir código.

26.- Dar doble click sobre el botón **cmdConversion** y escribir el siguiente código:

```
private void cmdConversion_Click(object sender, EventArgs e)
{
    // Programa que convierte dólares a pesos
    // Práctica 4

    // Declaración de variables
    float pesos, dolares, tc;

    //Asignación de valores a las variables
    dolares = System.Single.Parse(txtDolares.Text);
    tc = System.Single.Parse(txtTC.Text);

    //Cálculo
    pesos = dolares * tc;

    //Salida del resultado
    txtPesos.Text = pesos.ToString();
}
```

27.- Para regresar al diseño de la forma ir a la caja **Explorador de soluciones** y dar doble click sobre Form1.cs o ir a la opción del menú **VER** y elegir **Diseñador**.

28.- Dar doble click sobre el botón **cmdSalida** y el cursor aparecerá dentro de las llaves del siguiente subprograma:

```
private void cmdSalida_Click(object sender, EventArgs e)
{

}
}
```

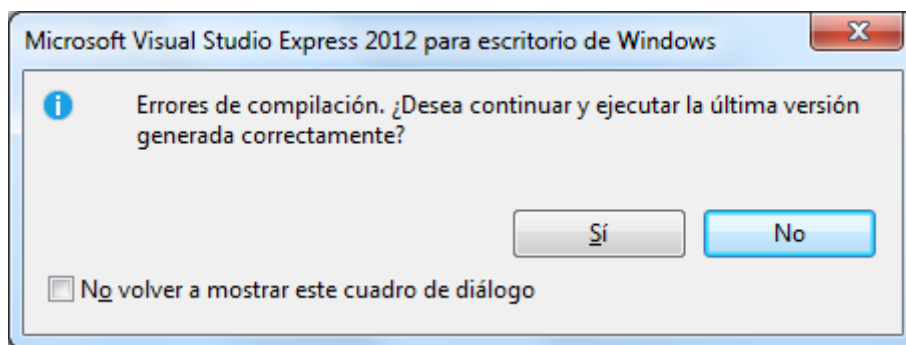
29.- Sin mover el cursor introducir el siguiente código:

```
//Salida del programa
Close();
```

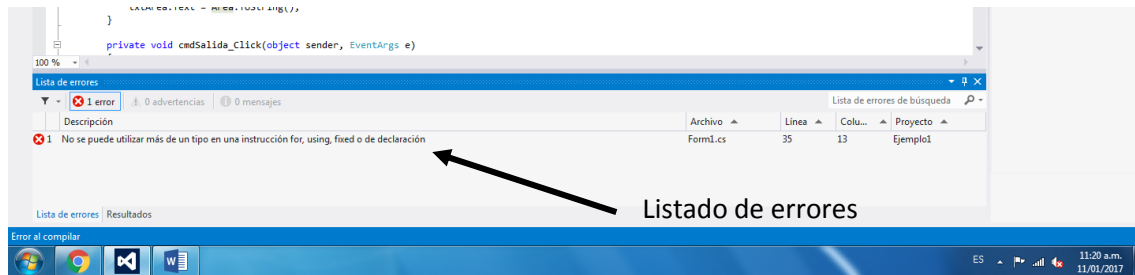
30.- Guardar el programa **Práctica 4**. En caso de que no hayas especificado desde el inicio el nombre y ubicación del programa ir a la opción del menú **ARCHIVO** y posteriormente ir a **Guardar WindowsFormsApplication1 como**. Asegurarse que en el **Explorador de soluciones** se encuentre elegido **WindowsFormsApplication1** para guardar correctamente.

31.- Para compilar y ejecutar el programa dar click en , presionar F5 o ir a la opción del menú **DEPURAR** y elegir **Iniciar depuración**.

32.- En caso que no se ejecute quiere decir que se presentaron errores durante la compilación o traducción y mostrará la siguiente caja:



Dar click en **No** para que permita revisar e identificar los errores de sintaxis y así poder corregirlos. Mostrará una división en la ventana donde se enlistan los errores (los errores varían):



Notas práctica 4:

- Dentro de la práctica 4 se vieron las clases Form, Label, Text, Button y las instancias de dichas clases a través de los objetos frmPractica4, lblPesos, lblDolares, lblTC, txtPesos, txtDolares, txtTC, cmdConversion, cmdSalida, respectivamente, que como ya se dijo, un objeto es una unidad dentro de un programa que consta de un estado (propiedades) y de un comportamiento (métodos), que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.
- Se modificaron las propiedades de Name para los objetos usando prefijos de acuerdo a la clase: Form (prefijo frm), Label (prefijo lbl), Text (prefijo txt), Button (prefijo cmd).
- Se ligaron métodos a los objetos de la clase Button (cmdConversion, cmdSalida).
- Las `//` se utilizan para dar inicio a un comentario dentro del código del programa. Estos comentarios no afectan el funcionamiento del mismo.

`// Programa que convierte dólares a pesos`

- Al estar trabajando en forma visual el objeto **Text Box** permite la entrada de datos que posteriormente se le asignarán a las variables correspondientes. Como la cajas de texto almacenan precisamente texto, es necesario convertir los caracteres numéricos a valores numéricos, en este caso como la variable que recibe es tipo **float** su convertidor es **Single**:

```
dolares = System.Single.Parse(txtDolares.Text);
```

- Por otro lado al momento de mostrar un valor numérico en una caja de texto es necesario convertirlo a cadena:

```
txtPesos.Text = pesos.ToString();
```

PRÁCTICA 5. SUMA DE DOS NÚMEROS.

OBJETIVO ESPECÍFICO.

Conocer y utilizar los operadores aritméticos para la solución de problemas del contexto utilizando el modelo de las 6'D para definir las etapas para solución de problemas, algoritmos y un lenguaje orientado a objetos aplicando la Metodología para la solución de problemas.

TEMAS CON LOS QUE SE RELACIONA.

- 2.1 Descripción del problema (enunciado).
- 2.2 Definición de solución (especificaciones).
- 2.3 Diseño de la solución (modelado).

DESARROLLO DE LA PRÁCTICA.

- a) Analizar la descripción de la siguiente situación que se desea resolver a través del desarrollo de un programa de cómputo.

- 1. En la escuela preescolar “El niño feliz” desean enseñar a los niños a realizar operaciones aritméticas básicas introduciéndolos inicialmente al operador aritmético (+) de la suma. Para hacer más atractivo el desarrollo de esta habilidad se hará uso de un programa que les permita introducir dos números que deseen sumar y les muestre el resultado. El niño podrá introducir las veces que desee los números a sumar y el programa les arrojará el resultado de manera sencilla y gráfica.

- b) Una vez analizado el planteamiento de la situación anterior y llevar a cabo la identificación y descripción del problema generando un enunciado claro que lo especifique.

- 2. Identifica el problema: _____

- 3. Descripción del problema: _____

- 4. Anota claramente y con precisión el enunciado del problema: _____

c) Partiendo del enunciado del problema llevar a cabo un análisis o a fondo para saber exactamente en qué consiste y así poder descomponerlo en cada una de sus partes facilitando su comprensión y posterior solución.

5. Utilizar el Modelo de Solución para el análisis del problema donde se desglosan tres partes muy importantes en la solución del problema: DATOS DE ENTRADA (Datos que se necesitan para resolver el problema), PROCESO (Procesos para solucionar el problema, por ejemplo los cálculos, etc.) y DATOS DE SALIDA (Datos que serán la salida, o sea, los resultados). Utilizar todos los modelos necesarios para cada alternativa de solución.

DATOS DE ENTRADA	PROCESOS	DATOS DE SALIDA

6. Identificar el resultado deseado: _____

7. Identificar claramente los datos a ingresar y transformar así como aquellos datos a generar: _____

8. Identificar claramente los procesos necesarios para transformar los datos en información y así lograr llegar al resultado esperado: _____

9. Identificar alternativas que permitan solucionar el problema: _____

10. Seleccionar la alternativa más adecuada para solucionar el problema : _____

- d) Partiendo del enunciado claro y preciso del problema a solucionar diseñar la solución a través del modelado utilizando UML y desarrollo de algoritmos.

11. Diagramas UML

Clase

	Nombre
	Atributos
	Métodos

12. Diagrama del algoritmo

HERRAMIENTAS DE PROGRAMACIÓN

COMPETENCIAS A DESARROLLAR

- Diseñar diagramas UML, algoritmos y pseudocódigo empleando el paradigma de POO para la solución de problemas.
- Capacidad de abstracción y análisis.
- Capacidad de aplicar los conocimientos en la práctica.
- Habilidades en el uso de las tecnologías de la información y de la comunicación.
- Habilidad para buscar, procesar y analizar información procedente de fuentes diversas.
- Capacidad para identificar, plantear y resolver problemas.
- Capacidad de trabajar en equipo.
- Habilidades interpersonales.
- Capacidad creativa.
- Habilidad para trabajar en forma autónoma.
- Compromiso ético.
- Compromiso con la calidad.

PRÁCTICA 6. CONVERTIR PESOS A DÓLARES.

OBJETIVO ESPECÍFICO.

Diseñar diagramas UML, algoritmos y pseudocódigo empleando el paradigma de POO y un lenguaje orientado a objetos así como conocer y utilizar los operadores aritméticos para la solución de problemas del contexto.

TEMAS CON LOS QUE SE RELACIONA.

3.1 Simbología.

3.2 Reglas para la construcción de diagramas.

3.3 Pseudocódigo.

3.4 Tipos de datos y expresiones.

4.1 Características del modelo orientado a objetos.

4.2 Elementos primordiales en el modelo de objetos.

4.3 Representación gráfica del diseño.

4.4 Relación entre la programación orientada a objetos y la estructurada.

DESARROLLO DE LA PRÁCTICA.

1.- Diseñar diagramas UML, algoritmo y posteriormente iniciar Visual C#

2.- Elegir **Aplicación de Windows Forms**

3.- Elegir **Nuevo Proyecto**

4.- Elegir **Aplicación de Windows Forms**

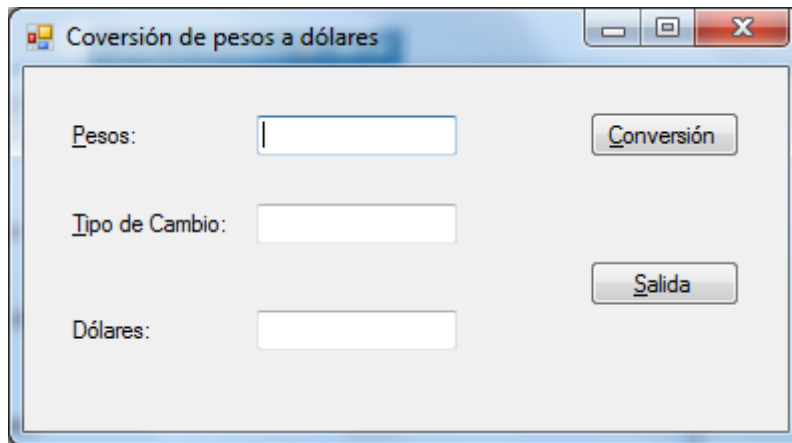
5.- Dentro de la caja que aparece ir a la sección de Especificaciones de guardado del archivo del programa e introducir lo siguiente:

Nombre: **Práctica 6**

Ubicación: (Ir a **Examinar** y elegir ubicación, usar unidad de almacenamiento móvil)

Nombre de la solución: (Automáticamente se copia el **Nombre**, dejarlo con el mismo)

6.- Diseñar la forma, que deberá quedar como se muestra en la siguiente figura:



a) Detalles de la forma.

7.- Dar click en la forma e ir a la **Ventana propiedades**, localizar la propiedad **Name** y dar un click a su derecha donde dice **Form1** y sustituir por **frmPractica5** (no acepta espacios en blanco) para así asignarle nombre a la forma.

8.- Estando la forma seleccionada ir a la **Ventana propiedades**, localizar la propiedad **Text** y dar un click a su derecha donde dice **Form1** y sustituir por **Conversión de pesos a dólares** para así agregarle el título que aparecerá en la forma.

9.- Ir la propiedad **BackColor** y elegir un color para la forma.

10.- El tamaño de la forma se puede cambiar como cualquier ventana de Windows con arrastrar cualquier parte de su contorno.

b) Detalles de las etiquetas.

11.- Para crear la etiqueta dar click en la **Cuadro de herramientas** en **Label**  y dar un click en la forma donde se desea insertar la etiqueta.

12.- Estando seleccionada la etiqueta ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Label1** y sustituir por **lblPesos** (no acepta espacios en blanco) para así asignarle nombre a la etiqueta.

13.- Estando la etiqueta seleccionada ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Label1** y sustituir por **&Pesos** para así agregarle el texto que aparecerá en la etiqueta (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución).

14.- Ir la propiedad **ForeColor** y elegir un color para la etiqueta.

15.- Ir la propiedad **Font** y elegir un Tipo de letra para la etiqueta.

c) Detalles de las cajas de texto.

16.- Para insertar una caja de texto dar click en la **Cuadro de herramientas** en  **Text Box** y dar click en la forma donde se desea insertar la caja de texto.

17.- Estando seleccionada la caja de texto ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **textBox1** y sustituir por **txtPesos** (no acepta espacios en blanco) para así asignarle nombre a la caja de texto.

18.- Ir la propiedad **BackColor** y elegir un color para la caja de texto.

19.- Ir la propiedad **ForeColor** y elegir un color para el texto que posteriormente aparecerá en la caja de texto.

20.- Ir la propiedad **Font** y elegir un Tipo de letra para el texto que posteriormente aparecerá en la caja de texto.

21.- Repetir los pasos del 11 al 20 para las siguientes etiquetas y caja de texto (crear primero la etiqueta y después su correspondiente caja de texto):

Propiedades		
<i>Name</i>	<i>Text</i>	<i>ReadOnly</i>
lblTC	&Tipo de cambio:	-
lbl Dolares	Dólares:	-
txtTC	(dejar vacío)	False
txt Dolares	(dejar vacío)	True

d) Detalles de los botones de comando.

22.-Para insertar un botón de comando dar un click en la **Cuadro de herramientas** en  **Button** y dar click en la forma donde se desea insertar el botón de comando.

23.- Estando seleccionado el botón de comando ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Button1** y sustituir por **cmdConversion** (no acepta espacios en blanco) para así asignarle nombre.

24.- Ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Button1** y sustituir por **&Conversión** para así agregarle el texto que aparecerá en el botón (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución). Puedes cambiar tipo de letra color, etc.

25.- Agregar botón de comando repitiendo los pasos 22 al 24 para la salida del programa dejando que también se active con la tecla **Esc**, así como establecer el botón que se ejecutará al presionar la tecla **Enter**, para lo que se modificarán dos propiedades de la forma **frmPractica5**. Utilizar las siguientes propiedades:

Propiedades			
<i>Name</i>	<i>Text</i>	<i>AcceptButton</i>	<i>CancelButton</i>
cmdSalida	&Salida	-	-
frmPractica6	-	cmdConversion	cmdSalida

e) Introducir código.

26.- Dar doble click sobre el botón **cmdConversion** y escribir el siguiente código:

```
private void cmdConversion_Click(object sender, EventArgs e)
{
    // Programa que convierte dólares a pesos
    // Práctica 6

    // Declaración de variables
    float pesos, dolares, tc;

    //Asignación de valores a las variables
    pesos = System.Single.Parse(txtPesos.Text);
    tc = System.Single.Parse(txtTC.Text);

    //Cálculo
    dolares = pesos / tc;

    //Salida del resultado
    txtDolares.Text = pesos.ToString();
}
```

27.- Para regresar al diseño de la forma ir a la caja **Explorador de soluciones** y dar doble click sobre Form1.cs o ir a la opción del menú **VER** y elegir **Diseñador**.

28.- Dar doble click sobre el botón **cmdSalida** y el cursor aparecerá dentro de las llaves del siguiente subprograma:

```
private void cmdSalida_Click(object sender, EventArgs e)
{

}
```

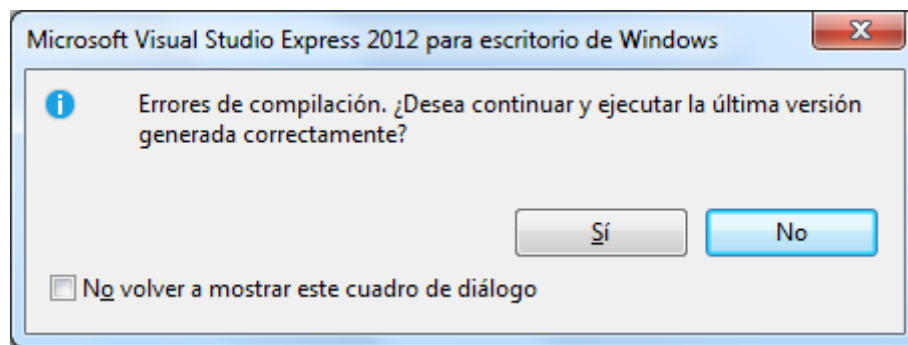
29.- Sin mover el cursor introducir el siguiente código:

```
//Salida del programa  
Close();
```

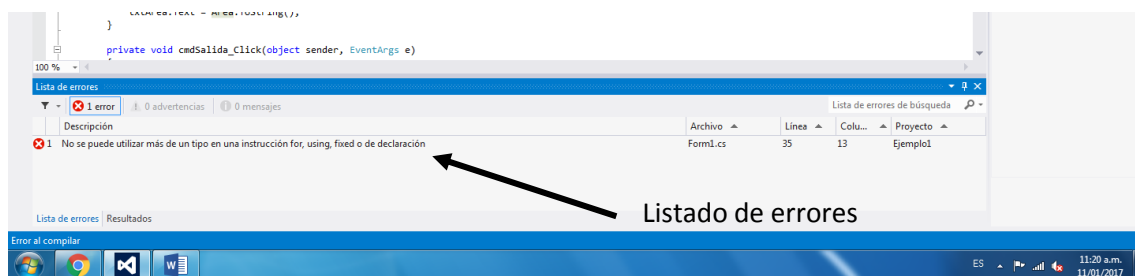
30.- Guardar el programa **Práctica 6**. En caso de que no hayas especificado desde el inicio el nombre y ubicación del programa ir a la opción del menú **ARCHIVO** y posteriormente ir a **Guardar WindowsFormsApplication1 como**. Asegurarse que en el **Explorador de soluciones** se encuentre elegido **WindowsFormsApplication1** para guardar correctamente.

31.- Para compilar y ejecutar el programa dar click en , presionar F5 o ir a la opción del menú **DEPURAR** y elegir **Iniciar depuración**.

32.- En caso que no se ejecute quiere decir que se presentaron errores durante la compilación o traducción y mostrará la siguiente caja:



Dar click en **No** para que permita revisar e identificar los errores de sintaxis y así poder corregirlos. Mostrará una división en la ventana donde se enlistan los errores (los errores varían):



Notas práctica 6:

- Dentro de la práctica 6 se vieron las clases Form, Label, Text, Button y las instancias de dichas clases a través de los objetos frmPractica4, lblPesos, lblDolares, lblTC, txtPesos, txtDolares, txtTC, cmdConversion, cmdSalida, respectivamente, que como ya se dijo, un objeto es una unidad dentro de un programa que consta de un estado (propiedades) y

de un comportamiento (métodos), que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.

- Se modificaron las propiedades de Name para los objetos usando prefijos de acuerdo a la clase: Form (prefijo frm), Label (prefijo lbl), Text (prefijo txt), Button (prefijo cmd).
- Se ligaron métodos a los objetos de la clase Button (cmdConversion, cmdSalida).
- Esta práctica introduce la declaración de variables de tipo real **float**.
- Las **//** se utilizan para dar inicio a un comentario dentro del código del programa. Estos comentarios no afectan el funcionamiento del mismo.

```
// Programa que convierte dólares a pesos
```

- Al estar trabajando en forma visual el objeto **Text Box** permite la entrada de datos que posteriormente se le asignarán a las variables correspondientes. Como las cajas de texto almacenan precisamente texto, es necesario convertir los caracteres numéricos a valores numéricos, en este caso como la variable que recibe es tipo **float** su convertidor es **Single**:

```
pesos = System.Single.Parse(txtPesos.Text);
```

- Por otro lado al momento de mostrar un valor numérico en una caja de texto es necesario convertirlo a cadena:

```
txtDolares.Text = dolares.ToString();
```

PRÁCTICA 7. SUMA DE DOS NÚMEROS.

OBJETIVO ESPECÍFICO.

Conocer y utilizar los operadores aritméticos para la solución de problemas del contexto utilizando el modelo de las 6'D para definir las etapas para solución de problemas, algoritmos y un lenguaje orientado a objetos.

TEMAS CON LOS QUE SE RELACIONA.

- 3.1 Simbología.
- 3.2 Reglas para la construcción de diagramas.
- 3.3 Pseudocódigo.
- 3.4 Tipos de datos y expresiones.
- 4.1 Características del modelo orientado a objetos.
- 4.2 Elementos primordiales en el modelo de objetos.
- 4.3 Representación gráfica del diseño.
- 4.4 Relación entre la programación orientada a objetos y la estructurada.

DESARROLLO DE LA PRÁCTICA.

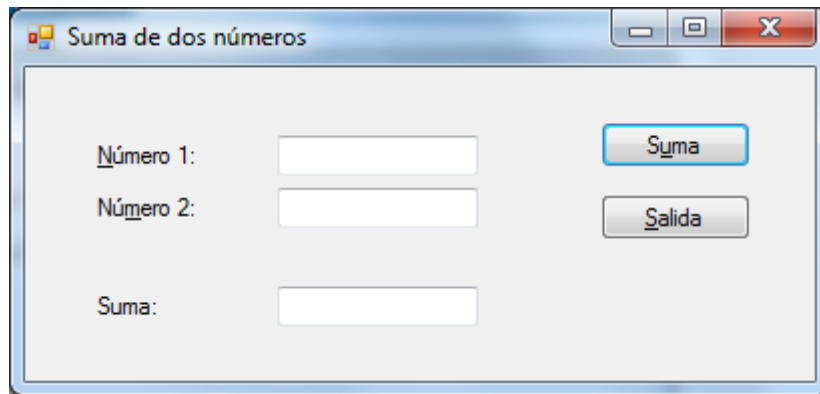
- 1.- Diseñar diagramas UML, algoritmo y posteriormente iniciar Visual C#
- 2.- Elegir **Aplicación de Windows Forms**
- 3.- Elegir **Nuevo Proyecto**
- 4.- Elegir **Aplicación de Windows Forms**
- 5.- Dentro de la caja que aparece ir a la sección de Especificaciones de guardado del archivo del programa e introducir lo siguiente:

Nombre: **Práctica 7**

Ubicación: (Ir a **Examinar** y elegir ubicación, usar unidad de almacenamiento móvil)

Nombre de la solución: (Automáticamente se copia el **Nombre**, dejarlo con el mismo)

- 6.- Diseñar la forma, que deberá quedar como se muestra en la siguiente figura:



a) Detalles de la forma.

7.- Dar click en la forma e ir a la **Ventana propiedades**, localizar la propiedad **Name** y dar un click a su derecha donde dice **Form1** y sustituir por **frmPractica6** (no acepta espacios en blanco) para así asignarle nombre a la forma.

8.- Estando la forma seleccionada ir a la **Ventana propiedades**, localizar la propiedad **Text** y dar un click a su derecha donde dice **Form1** y sustituir por **Suma de dos números** para así agregarle el título que aparecerá en la forma.

9.- Ir la propiedad **BackColor** y elegir un color para la forma.

10.- El tamaño de la forma se puede cambiar como cualquier ventana de Windows con arrastrar cualquier parte de su contorno.

b) Detalles de las etiquetas.

11.- Crear la etiqueta utilizando la herramienta **Label** y  dar un click donde se desea insertar la etiqueta.

12.- Estando seleccionada la etiqueta ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Label1** y sustituir por **lblNumero1** (no acepta espacios en blanco) para así asignarle nombre a la etiqueta.

13.- Estando la etiqueta seleccionada ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Label1** y sustituir por **&Número 1:** para así agregarle el texto que aparecerá en la etiqueta (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución).

14.- Ir la propiedad **ForeColor** y elegir un color para la etiqueta.

15.- Ir la propiedad **Font** y elegir un Tipo de letra para la etiqueta.

c) Detalles de las cajas de texto.

16.- Para insertar una caja de texto dar click en la **Cuadro de herramientas** en  **Text Box** y dar click en la forma donde se desea insertar la caja de texto.

17.- Estando seleccionada la caja de texto ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **textBox1** y sustituir por **txtNumero1** (no acepta espacios en blanco) para así asignarle nombre a la caja de texto.

18.- Ir la propiedad **BackColor** y elegir un color para la caja de texto.

19.- Ir la propiedad **ForeColor** y elegir un color para el texto que posteriormente aparecerá en la caja de texto.

20.- Ir la propiedad **Font** y elegir un Tipo de letra para el texto que posteriormente aparecerá en la caja de texto.

21.- Repetir los pasos del 11 al 20 para las siguientes etiquetas y caja de texto (crear primero la etiqueta y después su correspondiente caja de texto):

Propiedades		
Name	Text	ReadOnly
lblNumero2	Nú&mero 2:	
lblSuma	La suma es:	
txtNumero2	(dejar vacío)	False
txtSuma	(dejar vacío)	True

d) Detalles de los botones de comando.

22.-Para insertar un botón de comando dar un click en la **Cuadro de herramientas** en  **Button** y dar click en la forma donde se desea insertar el botón de comando.

23.- Estando seleccionado el botón de comando ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Button1** y sustituir por **cmdSuma** (no acepta espacios en blanco) para así asignarle nombre.

24.- Ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Button1** y sustituir por **S&uma** para así agregarle el texto que aparecerá en el botón (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución). Puedes cambiar tipo de letra color, etc.

25.- Agregar botón de comando repitiendo los pasos 22 al 24 para la salida del programa dejando que también se active con la tecla **Esc**, así como establecer el botón que se ejecutará al presionar

la tecla **Enter**, para lo que se modificarán dos propiedades de la forma **frmPractica7**. Utilizar las siguientes propiedades:

Propiedades			
<i>Name</i>	<i>Text</i>	<i>AcceptButton</i>	<i>CancelButton</i>
cmdSalida	&Salida	-	-
frmPractica7	-	cmdSuma	cmdSalida

e) Introducir código.

26.- Dar doble click sobre el botón **cmdSuma** y escribir el código correspondiente quedando finalmente como a continuación:

```
// Creado automáticamente para el botón cmdSuma
private void cmdSuma_Click(object sender, EventArgs e)
{
    // Programa que suma dos números
    // Práctica 7

    // Declaración de variables
    int N1, N2, Res;

    // Asignación de valores a las variables
    N1 = System.Int32.Parse(txtNumero1.Text);
    N2 = System.Int32.Parse(txtNumero2.Text);

    // Llamada a la función Suma ()
    Res = Num1 + Num2;

    //Despliegue del resultado de la suma
    txtSuma.Text = Res.ToString();
}
```

27.- Para regresar al diseño de la forma ir a la caja **Explorador de soluciones** y dar doble click sobre Form1.cs o ir a la opción del menú **VER** y elegir **Diseñador**.

28.- Dar doble click sobre el botón **cmdSalida** y el cursor aparecerá dentro de las llaves del siguiente subprograma:

```
private void cmdSalida_Click(object sender, EventArgs e)
{

}
```

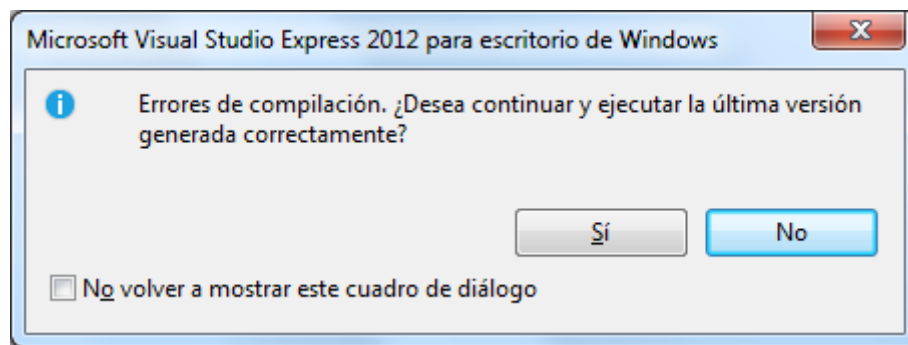
29.- Sin mover el cursor introducir el siguiente código:

```
//Salida del programa  
Close();
```

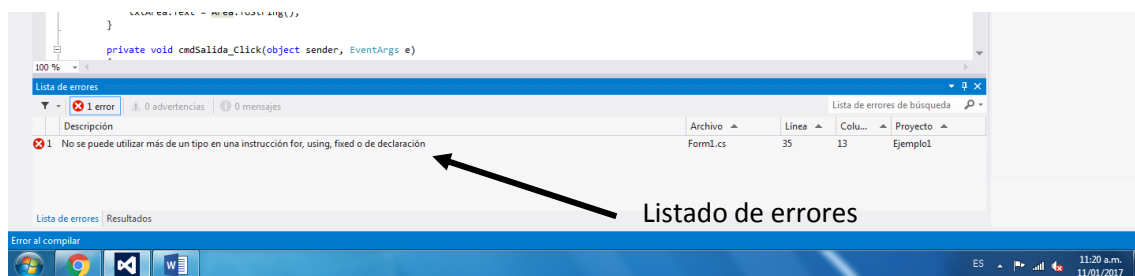
30.- Guardar el programa **Práctica 6**. En caso de que no hayas especificado desde el inicio el nombre y ubicación del programa ir a la opción del menú **ARCHIVO** y posteriormente ir a **Guardar WindowsFormsApplication1 como**. Asegurarse que en el **Explorador de soluciones** se encuentre elegido **WindowsFormsApplication1** para guardar correctamente.

31.- Para compilar y ejecutar el programa dar click en , presionar F5 o ir a la opción del menú **DEPURAR** y elegir **Iniciar depuración**.

32.- En caso que no se ejecute quiere decir que se presentaron errores durante la compilación o traducción y mostrará la siguiente caja:



Dar click en **No** para que permita revisar e identificar los errores de sintaxis y así poder corregirlos. Mostrará una división en la ventana donde se enlistan los errores (los errores varían):



33.- Probar el programa y realizar modificaciones si es necesario.

Notas práctica 7:

- Dentro de la práctica 7 se vieron las clases Form, Label, Text, Button y las instancias de dichas clases a través de los objetos frmPractica7, lblNumero1, lblNumero2, lblSuma, txtNumero1, txtNumero2, txtSuma, cmdSuma, cmdSalida, respectivamente, que como ya se dijo, un objeto es una unidad dentro de un programa que consta de un estado (propiedades) y de un comportamiento (métodos), que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.
- Se modificaron las propiedades de Name para los objetos usando prefijos de acuerdo a la clase: Form (prefijo frm), Label (prefijo lbl), Text (prefijo txt), Button (prefijo cmd).
- Se ligaron métodos a los objetos de la clase Button (cmdSuma, cmdSalida).
- Esta práctica introduce la declaración de variables de tipo real **int**.
- Las **//** se utilizan para dar inicio a un comentario dentro del código del programa. Estos comentarios no afectan el funcionamiento del mismo.

// Programa que suma dos números

- Al estar trabajando en forma visual el objeto **Text Box** permite la entrada de datos que posteriormente se le asignarán a las variables correspondientes. Como las cajas de texto almacenan precisamente texto, es necesario convertir los caracteres numéricos a valores numéricos, en este caso como la variable que recibe es tipo **int** su convertidor es **Int32**:

```
N1 = System.Int32.Parse(txtNumero1.Text);
```

- Por otro lado al momento de mostrar un valor numérico en una caja de texto es necesario convertirlo a cadena:

```
txtSuma.Text = Res.ToString();
```

PROGRAMACIÓN ORIENTADA A OBJETOS Y MODELADO

COMPETENCIAS A DESARROLLAR

- Aplica estructuras de datos y estructuras lógicas basándose en la POO para desarrollar la lógica de programación.
- Capacidad de abstracción y análisis.
- Capacidad de aplicar los conocimientos en la práctica.
- Habilidades en el uso de las tecnologías de la información y de la comunicación.
- Habilidad para buscar, procesar y analizar información procedente de fuentes diversas.
- Capacidad para identificar, plantear y resolver problemas.
- Capacidad de trabajar en equipo.
- Habilidades interpersonales.
- Capacidad creativa.
- Habilidad para trabajar en forma autónoma.
- Compromiso ético.
- Compromiso con la calidad.

PRÁCTICA 8. DETERMINAR SI UNA PERSONA ES MAYOR DE EDAD UTILIZANDO LA SELECTIVA SIMPLE.

OBJETIVO ESPECÍFICO.

Conocer y utilizar la estructura de control “selectiva simple” como parte de las estructuras de datos y estructuras lógicas basándose en la POO para desarrollar la lógica de programación por medio de algoritmos y el lenguaje de programación Visual C#.

TEMAS CON LOS QUE SE RELACIONA.

- 4.1 Características del modelo orientado a objetos.
- 4.2 Elementos primordiales en el modelo de objetos.
- 4.3 Representación gráfica del diseño.
- 4.4 Relación entre la programación orientada a objetos y la estructurada.

DESARROLLO DE LA PRÁCTICA.

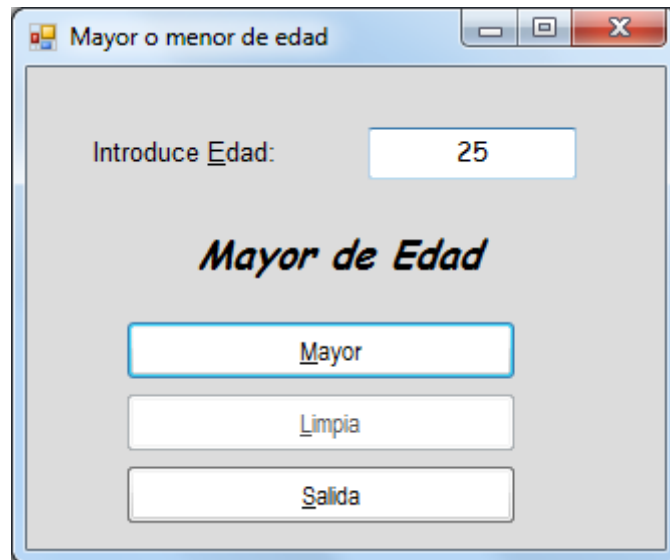
- 1.- Diseñar diagramas UML, algoritmo y posteriormente iniciar Visual C#
- 2.- Elegir **Aplicación de Windows Forms**
- 3.- Elegir **Nuevo Proyecto**
- 4.- Elegir **Aplicación de Windows Forms**
- 5.- Dentro de la caja que aparece ir a la sección de Especificaciones de guardado del archivo del programa e introducir lo siguiente:

Nombre: **Práctica 8**

Ubicación: (Ir a **Examinar** y elegir ubicación, usar unidad de almacenamiento móvil)

Nombre de la solución: (Automáticamente se copia el **Nombre**, dejarlo con el mismo)

- 6.- Diseñar la forma, que deberá quedar como se muestra en la siguiente figura:



a) Detalles de la forma.

7.- Dar click en la forma e ir a la **Ventana propiedades**, localizar la propiedad **Name** y dar un click a su derecha donde dice **Form1** y sustituir por **frmPractica8** (no acepta espacios en blanco) para así asignarle nombre a la forma.

8.- Estando la forma seleccionada ir a la **Ventana propiedades**, localizar la propiedad **Text** y dar un click a su derecha donde dice **Form1** y sustituir por **Mayor o menor de edad** para así agregarle el título que aparecerá en la forma.

9.- Ir la propiedad **BackColor** y elegir un color para la forma.

10.- El tamaño de la forma se puede cambiar como cualquier ventana de Windows con arrastrar cualquier parte de su contorno.

b) Detalles de las etiquetas.

11.- Crear etiqueta con **Label**  y dar un click en la forma donde se desea insertar la etiqueta.

12.- Estando seleccionada la etiqueta ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Label1** y sustituir por **lblEdad** (no acepta espacios en blanco) para así asignarle nombre a la etiqueta.

13.- Estando la etiqueta seleccionada ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Label1** y sustituir por **Introduce Edad:** para así agregarle el texto que aparecerá en la etiqueta.

14.- Ir la propiedad **ForeColor** y elegir un color para la etiqueta.

15.- Ir la propiedad **Font** y elegir un Tipo de letra para la etiqueta.

16.- Insertar una nueva etiqueta y cambiar la propiedad Name: **lblMayMen** y en su propiedad Text: **(dejar vacío)**. Modificar tamaño de letra.

c) Detalles de las cajas de texto.

17.- Para insertar una caja de texto dar click en la **Cuadro de herramientas** en  **Text Box** y dar click en la forma donde se desea insertar la caja de texto.

18.- Estando seleccionada la caja de texto ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **textBox1** y sustituir por **txtEdad** (no acepta espacios en blanco) para así asignarle nombre a la caja de texto.

19.- Ir la propiedad **BackColor** y elegir un color para la caja de texto.

20.- Ir la propiedad **ForeColor** y elegir un color para el texto que posteriormente aparecerá en la caja de texto.

21.- Ir la propiedad **Font** y elegir un Tipo de letra para el texto que posteriormente aparecerá en la caja de texto.

d) Detalles de los botones de comando.

22.-Para insertar un botón de comando dar un click en la **Cuadro de herramientas** en  **Button** y dar click en la forma donde se desea insertar el botón de comando.

23.- Estando seleccionado el botón de comando ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Button1** y sustituir por **cmdMayMen** (no acepta espacios en blanco) para así asignarle nombre.

24.- Ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Button1** y sustituir por **&Mayor** para así agregarle el texto que aparecerá en el botón (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución). Puedes cambiar tipo de letra color, etc.

25.- Agregar botón de comando repitiendo los pasos 22 al 24 para la salida del programa dejando que también se active con la tecla **Esc**, así como establecer el botón que se ejecutará al presionar la tecla **Enter**, para lo que se modificarán dos propiedades de la forma **frmPractica8**. Utilizar las siguientes propiedades:

Propiedades			
<i>Name</i>	<i>Text</i>	<i>AcceptButton</i>	<i>CancelButton</i>
cmdSalida	&Salida	-	-
cmdLimpia	&Limpia	-	-
frmPractica8	-	cmdMayMen	cmdSalida

e) Introducir código.

26.- Dar doble click sobre el botón **cmdMayMen** y escribir el código correspondiente quedando finalmente como a continuación:

```
// Programa que determina si una persona es mayor de edad
// utilizando la selectiva simple
// Práctica 8

//Codigo para boton cmdMayMen
private void cmdMayMen_Click(object sender, EventArgs e)
{
    //Declaracion de variables
    int Edad;

    //Asignacion de valores a las variables
    Edad = System.Int32.Parse(txtEdad.Text);

    //Condicion que determina mayor de edad
    if (Edad >= 18)
        lblMayMen.Text = "Mayor de Edad";
}
```

27.- Dar doble click sobre el botón **cmdLimpia** y escribir el código correspondiente quedando finalmente como a continuación:

```
//Codigo para boton cmdLimpia
private void cmdLimpia_Click(object sender, EventArgs e)
{
    lblMayMen.Text = "";
    txtEdad.Clear();
    txtEdad.Focus();
}
```

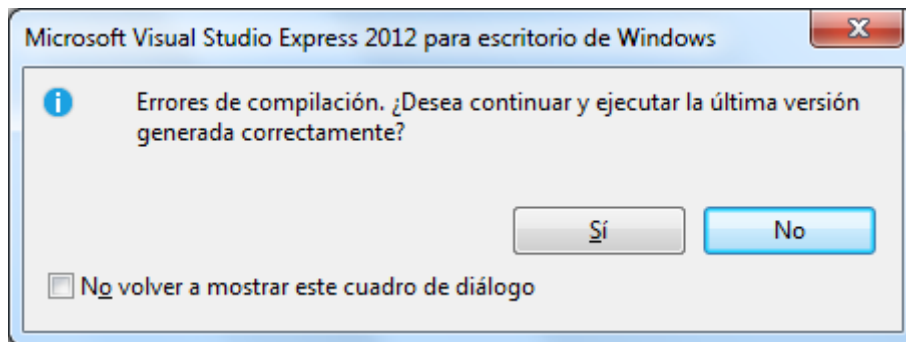
28.- Dar doble click al botón cmdSalida y sin mover el cursor introducir el código correspondiente quedando así finalmente:

```
//Creado automáticamente para el botón cmdSalida
private void cmdSalida_Click(object sender, EventArgs e)
{
    Close();
}
```

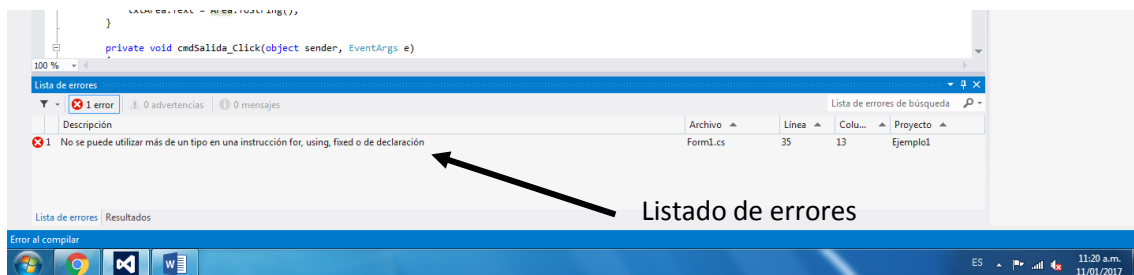
29.- Guardar el programa **Práctica 8**. En caso de que no hayas especificado desde el inicio el nombre y ubicación del programa ir a la opción del menú **ARCHIVO** y posteriormente ir a **Guardar WindowsFormsApplication1 como**. Asegurarse que en el **Explorador de soluciones** se encuentre elegido **WindowsFormsApplication1** para guardar correctamente.

30.- Para compilar y ejecutar el programa dar click en , presionar F5 o ir a la opción del menú **DEPURAR** y elegir **Iniciar depuración**.

31.- En caso que no se ejecute quiere decir que se presentaron errores durante la compilación o traducción y mostrará la siguiente caja:



Dar click en **No** para que permita revisar e identificar los errores de sintaxis y así poder corregirlos. Mostrará una división en la ventana donde se enlistan los errores (los errores varían):



32.- Probar el programa y realizar modificaciones si es necesario.

Notas práctica 8:

- Dentro de la práctica 8 se vieron las clases Form, Label, Text, Button y las instancias de dichas clases a través de los objetos frmPractica8, lblEdad, lblMayMen, txtEdad, cmdMayMen, cmdLimpia, cmdSalida, respectivamente, que como ya se dijo, un objeto es una unidad dentro de un programa que consta de un estado (propiedades) y de un comportamiento (métodos), que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.
- Se modificaron las propiedades de Name para los objetos usando prefijos de acuerdo a la clase: Form (prefijo frm), Label (prefijo lbl), Text (prefijo txt), Button (prefijo cmd).
- Se ligaron métodos a los objetos de la clase Button (cmdMayMen, cmdSalida y cmdLimpia).
- Aquí se utiliza la estructura de control **if()** para establecer condiciones y así elegir un curso de acción dentro del programa.

```
if (Edad >= 18)
```

- Se utilizan operadores relacionales para lograr establecer los criterios.
- Para esta práctica solamente la selectiva realiza una acción cuando la condición es verdadera, en caso contrario no realiza acción alguna.

```
lblMayMen.Text = "Mayor de Edad";
```

- Se integra la acción de borrar la salida de datos para iniciar con la siguiente evaluación.

lblMayMen.Text = "";	Asigna vacío a la etiqueta
txtEdad.Clear();	Limpia la caja de texto
txtEdad.Focus();	Centra el foco en la caja de texto

PRÁCTICA 9. DETERMINAR SI UNA PERSONA ES MAYOR O MENOR DE EDAD, UTILIZANDO LA SELECTIVA DOBLE.

OBJETIVO ESPECÍFICO.

Conocer y utilizar la estructura de control “selectiva doble” como parte de las estructuras de datos y estructuras lógicas basándose en la POO para desarrollar la lógica de programación por medio de algoritmos y el lenguaje de programación Visual C#.

TEMAS CON LOS QUE SE RELACIONA.

- 4.1 Características del modelo orientado a objetos.
- 4.2 Elementos primordiales en el modelo de objetos.
- 4.3 Representación gráfica del diseño.
- 4.4 Relación entre la programación orientada a objetos y la estructurada.

DESARROLLO DE LA PRÁCTICA.

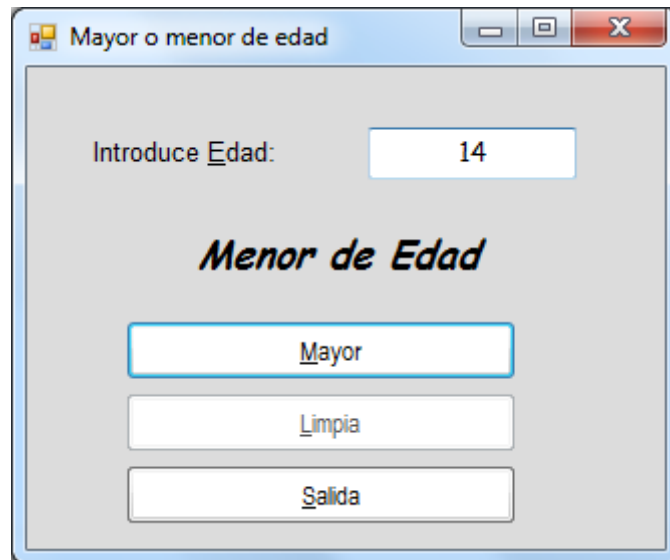
- 1.- Iniciar Visual C#
- 2.- Elegir **Aplicación de Windows Forms**
- 3.- Elegir **Nuevo Proyecto**
- 4.- Elegir **Aplicación de Windows Forms**
- 5.- Dentro de la caja que aparece ir a la sección de Especificaciones de guardado del archivo del programa e introducir lo siguiente:

Nombre: **Práctica 9**

Ubicación: (Ir a **Examinar** y elegir ubicación, usar unidad de almacenamiento móvil)

Nombre de la solución: (Automáticamente se copia el **Nombre**, dejarlo con el mismo)

- 6.- Diseñar la forma, que deberá quedar como se muestra en la siguiente figura:



a) Detalles de la forma.

7.- Dar click en la forma e ir a la **Ventana propiedades**, localizar la propiedad **Name** y dar un click a su derecha donde dice **Form1** y sustituir por **frmPractica9** (no acepta espacios en blanco) para así asignarle nombre a la forma.

8.- Estando la forma seleccionada ir a la **Ventana propiedades**, localizar la propiedad **Text** y dar un click a su derecha donde dice **Form1** y sustituir por **Mayor o menor de edad** para así agregarle el título que aparecerá en la forma.

9.- Ir la propiedad **BackColor** y elegir un color para la forma.

10.- El tamaño de la forma se puede cambiar como cualquier ventana de Windows con arrastrar cualquier parte de su contorno.

b) Detalles de las etiquetas.

11.- Crear etiqueta con **Label**  y dar un click en la forma donde se desea insertar la etiqueta.

12.- Estando seleccionada la etiqueta ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Label1** y sustituir por **lblEdad** (no acepta espacios en blanco) para así asignarle nombre a la etiqueta.

13.- Estando la etiqueta seleccionada ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Label1** y sustituir por **Introduce Edad:** para así agregarle el texto que aparecerá en la etiqueta.

14.- Ir la propiedad **ForeColor** y elegir un color para la etiqueta.

15.- Ir la propiedad **Font** y elegir un Tipo de letra para la etiqueta.

16.- Insertar una nueva etiqueta y cambiar la propiedad Name: **lblMayMen** y en su propiedad Text: **(dejar vacío)**. Modificar tamaño de letra.

c) Detalles de las cajas de texto.

17.- Para insertar una caja de texto dar click en la **Cuadro de herramientas** en  **Text Box** y dar click en la forma donde se desea insertar la caja de texto.

18.- Estando seleccionada la caja de texto ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **textBox1** y sustituir por **txtEdad** (no acepta espacios en blanco) para así asignarle nombre a la caja de texto.

19.- Ir la propiedad **BackColor** y elegir un color para la caja de texto.

20.- Ir la propiedad **ForeColor** y elegir un color para el texto que posteriormente aparecerá en la caja de texto.

21.- Ir la propiedad **Font** y elegir un Tipo de letra para el texto que posteriormente aparecerá en la caja de texto.

d) Detalles de los botones de comando.

22.-Para insertar un botón de comando dar un click en la **Cuadro de herramientas** en  **Button** y dar click en la forma donde se desea insertar el botón de comando.

23.- Estando seleccionado el botón de comando ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Button1** y sustituir por **cmdMayMen** (no acepta espacios en blanco) para así asignarle nombre.

24.- Ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Button1** y sustituir por **&Mayor** para así agregarle el texto que aparecerá en el botón (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución). Puedes cambiar tipo de letra color, etc.

25.- Agregar botón de comando repitiendo los pasos 22 al 24 para la salida del programa dejando que también se active con la tecla **Esc**, así como establecer el botón que se ejecutará al presionar la tecla **Enter**, para lo que se modificarán dos propiedades de la forma **frmPractica9**. Utilizar las siguientes propiedades:

Propiedades			
<i>Name</i>	<i>Text</i>	<i>AcceptButton</i>	<i>CancelButton</i>
cmdSalida	&Salida	-	-
cmdLimpia	&Limpia	-	-
frmPractica9	-	cmdMayMen	cmdSalida

e) Introducir código.

26.- Dar doble click sobre el botón **cmdMayMen** y escribir el código correspondiente quedando finalmente como a continuación:

```
// Programa que determina si una persona es mayor o menor de edad
// utilizando la selectiva doble
// Práctica 9

//Codigo para boton cmdMayMen
private void cmdMayMen_Click(object sender, EventArgs e)
{
    //Declaracion de variables
    int Edad;

    //Asignacion de valores a las variables
    Edad = System.Int32.Parse(txtEdad.Text);

    //Condicion que determina mayor de edad
    if (Edad >= 18)
        lblMayMen.Text = "Mayor de Edad";
    else
        lblMayMen.Text = "Menor de Edad";
}
```

27.- Dar doble click sobre el botón **cmdLimpia** y escribir el código correspondiente quedando finalmente como a continuación:

```
//Codigo para boton cmdLimpia
private void cmdLimpia_Click(object sender, EventArgs e)
{
    lblMayMen.Text = "";
    txtEdad.Clear();
    txtEdad.Focus();
}
```

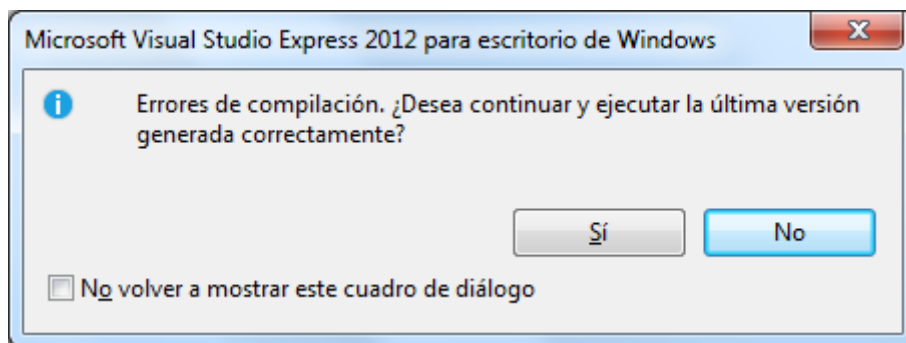
28.- Dar doble click al botón cmdSalida y sin mover el cursor introducir el código correspondiente quedando así finalmente:

```
//Creado automáticamente para el botón cmdSalida
private void cmdSalida_Click(object sender, EventArgs e)
{
    Close();
}
```

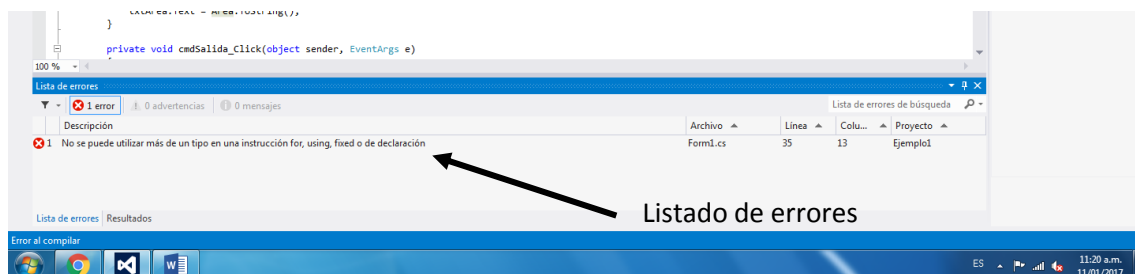
29.- Guardar el programa **Práctica 9**. En caso de que no hayas especificado desde el inicio el nombre y ubicación del programa ir a la opción del menú **ARCHIVO** y posteriormente ir a **Guardar WindowsFormsApplication1 como**. Asegurarse que en el **Explorador de soluciones** se encuentre elegido **WindowsFormsApplication1** para guardar correctamente.

30.- Para compilar y ejecutar el programa dar click en , presionar F5 o ir a la opción del menú **DEPURAR** y elegir **Iniciar depuración**.

31.- En caso que no se ejecute quiere decir que se presentaron errores durante la compilación o traducción y mostrará la siguiente caja:



Dar click en **No** para que permita revisar e identificar los errores de sintaxis y así poder corregirlos. Mostrará una división en la ventana donde se enlistan los errores (los errores varían):



32.- Probar el programa y realizar modificaciones si es necesario.

Notas práctica 9:

- Dentro de la práctica 8 se vieron las clases Form, Label, Text, Button y las instancias de dichas clases a través de los objetos frmPractica8, lblEdad, lblMayMen, txtEdad, cmdMayMen, cmdLimpia, cmdSalida, respectivamente, que como ya se dijo, un objeto es una unidad dentro de un programa que consta de un estado (propiedades) y de un comportamiento (métodos), que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.
- Se modificaron las propiedades de Name para los objetos usando prefijos de acuerdo a la clase: Form (prefijo frm), Label (prefijo lbl), Text (prefijo txt), Button (prefijo cmd).
- Se ligaron métodos a los objetos de la clase Button (cmdMayMen, cmdSalida y cmdLimpia).
- Aquí se utiliza la estructura de control **if()** para establecer condiciones y así elegir un curso de acción dentro del programa.

```
if (Edad >= 18)
```

- Se utilizan operadores relacionales para lograr establecer los criterios.
- La selectiva doble realiza una acción cuando la condición es verdadera y de igual forma cuando la condición es falsa en la sentencia **else**.

```
if (Edad >= 18)
    lblMayMen.Text = "Mayor de Edad";           //Verdadero
else
    lblMayMen.Text = "Menor de Edad";           //Falso
```

- Se integra la acción de borrar la salida de datos para iniciar con la siguiente evaluación.

```
lblMayMen.Text = "";           Asigna vacío a la etiqueta
txtEdad.Clear();              Limpia la caja de texto
txtEdad.Focus();              Centra el foco en la caja de texto
```

PRÁCTICA 10. SELECCIÓN DE ALUMNO PARA PARTICIPACIÓN EN CONCURSO, LOS REQUISITOS SON: PROMEDIO DE 90 O MÁS Y SER DE 5TO. SEMESTRE UTILIZANDO LA SELECTIVA COMPUESTA.

OBJETIVO ESPECÍFICO.

Conocer y utilizar la estructura de control “selectiva compuesta” como parte de las estructuras de datos y estructuras lógicas basándose en la POO para desarrollar la lógica de programación por medio de algoritmos y el lenguaje de programación Visual C#.

TEMAS CON LOS QUE SE RELACIONA.

- 4.1 Características del modelo orientado a objetos.
- 4.2 Elementos primordiales en el modelo de objetos.
- 4.3 Representación gráfica del diseño.
- 4.4 Relación entre la programación orientada a objetos y la estructurada.

DESARROLLO DE LA PRÁCTICA.

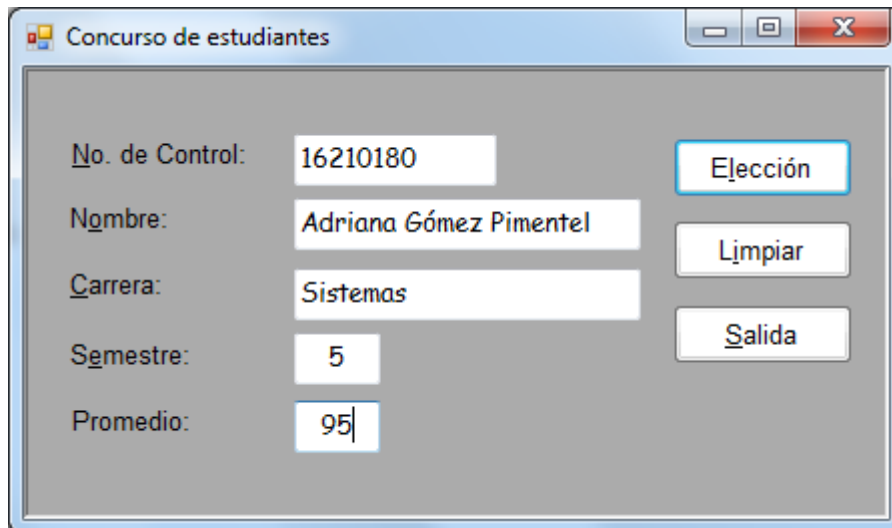
- 1.- Iniciar Visual C#
- 2.- Elegir **Aplicación de Windows Forms**
- 3.- Elegir **Nuevo Proyecto**
- 4.- Elegir **Aplicación de Windows Forms**
- 5.- Dentro de la caja que aparece ir a la sección de Especificaciones de guardado del archivo del programa e introducir lo siguiente:

Nombre: **Práctica 10**

Ubicación: (Ir a **Examinar** y elegir ubicación, usar unidad de almacenamiento móvil)

Nombre de la solución: (Automáticamente se copia el **Nombre**, dejarlo con el mismo)

- 6.- Diseñar la forma, que deberá quedar como se muestra en la siguiente figura:



No. de Control:	16210180	Elección
Nombre:	Adriana Gómez Pimentel	Limpiar
Carrera:	Sistemas	Salida
Semestre:	5	
Promedio:	95	

a) Detalles de la forma.

7.- Dar click en la forma e ir a la **Ventana propiedades**, y localizar la propiedad **Name** y dar un click a su derecha donde dice **Form1** y sustituir por **frmPractica10** (no acepta espacios en blanco) para así asignarle nombre a la forma.

8.- Estando la forma seleccionada ir a la **Ventana propiedades**, y localizar la propiedad **Text** y dar un click a su derecha donde dice **Form1** y sustituir por **Concurso de estudiantes** para así agregarle el título que aparecerá en la forma.

9.- Ir la propiedad **BackColor** y elegir un color para la forma.

10.- El tamaño de la forma se puede cambiar como cualquier ventana de Windows con arrastrar cualquier parte de su contorno.

b) Detalles de las etiquetas.

11.- Crear la etiqueta utilizando la herramienta **Label**  y dar un click donde se desea insertar la etiqueta.

12.- Estando seleccionada la etiqueta ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Label1** y sustituir por **lblNoControl** (no acepta espacios en blanco) para así asignarle nombre a la etiqueta.

13.- Estando la etiqueta seleccionada ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Label1** y sustituir por **&No. De Control:** para así agregarle el texto que aparecerá en la etiqueta (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución).

14.- Ir la propiedad **ForeColor** y elegir un color para la etiqueta.

15.- Ir la propiedad **Font** y elegir un Tipo de letra para la etiqueta.

c) Detalles de las cajas de texto.

16.- Para insertar una caja de texto dar click en la **Cuadro de herramientas** en  **Text Box** y dar click en la forma donde se desea insertar la caja de texto.

17.- Estando seleccionada la caja de texto ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **textBox1** y sustituir por **txtNoControl** (no acepta espacios en blanco) para así asignarle nombre a la caja de texto.

18.- Ir la propiedad **BackColor** y elegir un color para la caja de texto.

19.- Ir la propiedad **ForeColor** y elegir un color para el texto que posteriormente aparecerá en la caja de texto.

20.- Ir la propiedad **Font** y elegir un Tipo de letra para el texto que posteriormente aparecerá en la caja de texto.

21.- Repetir los pasos del 11 al 20 para las siguientes etiquetas y caja de texto (crear primero la etiqueta y después su correspondiente caja de texto):

Propiedades	
<i>Name</i>	<i>Text</i>
lblNombre	N&ombre
lblCarrera	&Carrera:
lblSemestre	&Semestre:
lblPromedio	&Promedio:
lblEleccion	(dejar vacío)
txtNombre	(dejar vacío)
txtCarrera	(dejar vacío)
txtSemestre	(dejar vacío)
txtPromedio	(dejar vacío)

d) Detalles de los botones de comando.

22.-Para insertar un botón de comando dar un click en la **Cuadro de herramientas** en  **Button** y dar click en la forma donde se desea insertar el botón de comando.

23.- Estando seleccionado el botón de comando ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Button1** y sustituir por **cmdEleccion** (no acepta espacios en blanco) para así asignarle nombre.

24.- Ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Button1** y sustituir por **&Elección** para así agregarle el texto que aparecerá en el botón (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución). Puedes cambiar tipo de letra color, etc.

25.- Agregar botón de comando repitiendo los pasos 22 al 24 para la salida del programa dejando que también se active con la tecla **Esc**, así como establecer el botón que se ejecutará al presionar la tecla **Enter**, para lo que se modificarán dos propiedades de la forma **frmPractica10**. Utilizar las siguientes propiedades:

Propiedades			
<i>Name</i>	<i>Text</i>	<i>AcceptButton</i>	<i>CancelButton</i>
cmdSalida	&Salida	-	-
cmdLimpiar	&Limpiar	-	-
frmPractica10	-	cmdEleccion	cmdSalida

e) Introducir código.

26.- Insertar el siguiente código para el botón cmdEleccion:

```
private void cmdEleccion_Click(object sender, EventArgs e)
{
    //Programa que selecciona un alumno para participar en concurso
    //Práctica 10

    //Declaración de variables
    int NoControl, Semestre;
    float Promedio;
    string Nombre, Carrera;

    // Asignación de valores a las variables
    NoControl = System.Int32.Parse(txtNoControl.Text);
    Nombre = txtNombre.Text;
    Carrera = txtCarrera.Text;
    Semestre = System.Int32.Parse(txtSemestre.Text);
    Promedio = System.Single.Parse(txtPromedio.Text);

    // Condición Compuesta
    if (Promedio >= 90 && Semestre == 5)
        lblEleccion.Text = "ACEPTADO";
    else
        lblEleccion.Text = "NO ACEPTADO";
}
```

27.- Dar doble click al botón cmdLimpiar y sin mover el cursor introducir el código correspondiente quedando así finalmente:

```
//Creado automáticamente para el botón cmdLimpiar
private void cmdLimpiar_Click(object sender, EventArgs e)
{
    txtNoControl.Clear();
    txtNoControl.Focus();
    txtNombre.Clear();
    txtCarrera.Clear();
    txtSemestre.Clear();
    txtPromedio.Clear();
    lblEleccion.Text = "";
}
```

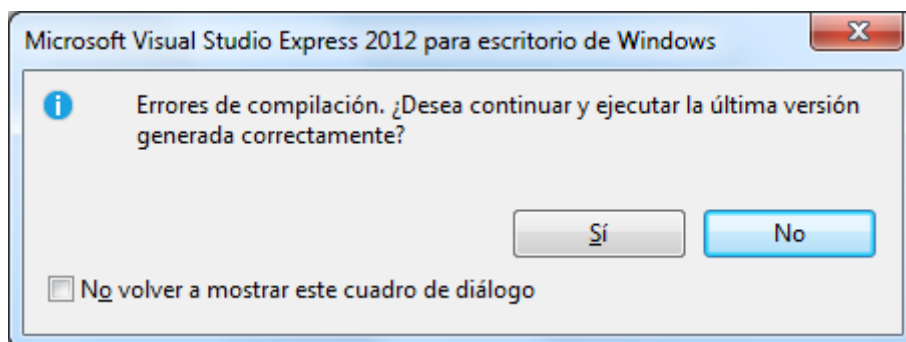
28.- Dar doble click al botón cmdSalida y sin mover el cursor introducir el código correspondiente, quedando así finalmente:

```
//Creado automáticamente para el botón cmdSalida
private void cmdSalida_Click(object sender, EventArgs e)
{
    Close();
}
```

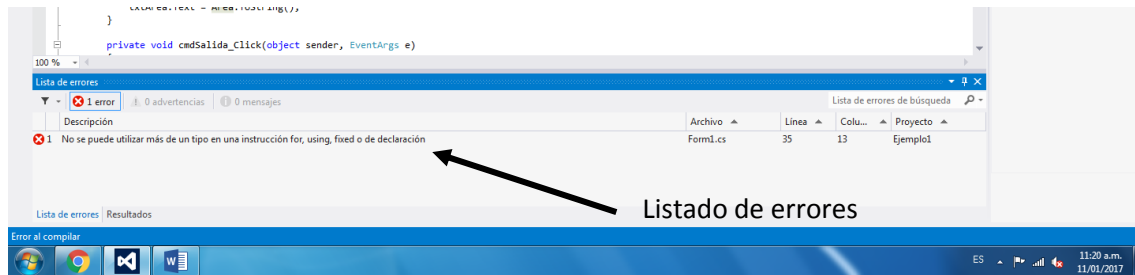
29.- Guardar el programa **Práctica 10**. En caso de que no hayas especificado desde el inicio el nombre y ubicación del programa ir a la opción del menú **ARCHIVO** y posteriormente ir a **Guardar WindowsFormsApplication1 como**. Asegurarse que en el **Explorador de soluciones** se encuentre elegido **WindowsFormsApplication1** para guardar correctamente.

30.- Para compilar y ejecutar el programa dar click en , presionar F5 o ir a la opción del menú **DEPURAR** y elegir **Iniciar depuración**.

31.- En caso que no se ejecute quiere decir que se presentaron errores durante la compilación o traducción y mostrará la siguiente caja:



Dar click en **No** para que permita revisar e identificar los errores de sintaxis y así poder corregirlos. Mostrará una división en la ventana donde se enlistan los errores (los errores varían):



32.- Probar el programa y realizar modificaciones si es necesario.

Notas práctica 10:

- Dentro de la práctica 10 se vieron las clases Form, Label, Text, Button y las instancias de dichas clases a través de los objetos frmPractica10, lblNoControl, lblNombre, lblCarrera, lblSemestre, lblPromedio, lblEleccion, txtNoControl, txtNombre, txtCarrera, txtSemestre, txtPromedio, cmdEleccion, cmdLimpiar, cmdSalida, respectivamente, que como ya se dijo, un objeto es una unidad dentro de un programa que consta de un estado (propiedades) y de un comportamiento (métodos), que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.
- Se modificaron las propiedades de Name para los objetos usando prefijos de acuerdo a la clase: Form (prefijo frm), Label (prefijo lbl), Text (prefijo txt), Button (prefijo cmd).
- Se ligaron métodos a los objetos de la clase Button (cmdEleccion, cmdSalida y cmdLimpiar).
- En este ejercicio se está utilizando las condiciones compuestas. Para lograr esto se requiere de los operadores lógicos.

```
if (Promedio >= 90 && Semestre == 5)
```

- Para centrar el foco en un objeto (activar desde código) es necesario especificar el nombre de este y agregar el método **Focus()**.

```
txtNoControl.Focus();
```

- Para limpiar el contenido el objeto caja de texto **Text Box** es necesario especificar el nombre de este y agregar el método **Clear()**.

```
txtNoControl.Clear();
```

PRÁCTICA 11. DETERMINAR EL MAYOR DE DOS NUMEROS CONSIDERANDO SER IGUALES UTILIZANDO LA SELECTIVA ANIDADA.

OBJETIVO ESPECÍFICO.

Conocer y utilizar la estructura de control “selectiva anidada” como parte de las estructuras de datos y estructuras lógicas basándose en la POO para desarrollar la lógica de programación por medio de algoritmos y el lenguaje de programación Visual C#.

TEMAS CON LOS QUE SE RELACIONA.

- 4.1 Características del modelo orientado a objetos.
- 4.2 Elementos primordiales en el modelo de objetos.
- 4.3 Representación gráfica del diseño.
- 4.4 Relación entre la programación orientada a objetos y la estructurada.

DESARROLLO DE LA PRÁCTICA.

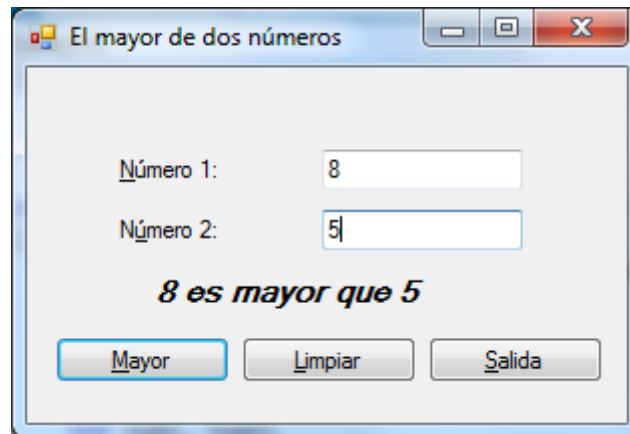
- 1.- Iniciar Visual C#
- 2.- Elegir **Aplicación de Windows Forms**
- 3.- Elegir **Nuevo Proyecto**
- 4.- Elegir **Aplicación de Windows Forms**
- 5.- Dentro de la caja que aparece ir a la sección de Especificaciones de guardado del archivo del programa e introducir lo siguiente:

Nombre: **Práctica 11**

Ubicación: (Ir a **Examinar** y elegir ubicación, usar unidad de almacenamiento móvil)

Nombre de la solución: (Automáticamente se copia el **Nombre**, dejarlo con el mismo)

- 6.- Diseñar la forma, que deberá quedar como se muestra en la siguiente figura:



a) Detalles de la forma.

7.- Dar click en la forma e ir a la **Ventana propiedades**, localizar la propiedad **Name** y dar un click a su derecha donde dice **Form1** y sustituir por **frmPractica11** (no acepta espacios en blanco) para así asignarle nombre a la forma.

8.- Estando la forma seleccionada ir a la **Ventana propiedades**, localizar la propiedad **Text** y dar un click a su derecha donde dice **Form1** y sustituir por **El mayor de dos números** para así agregarle el título que aparecerá en la forma.

9.- Ir la propiedad **BackColor** y elegir un color para la forma.

10.- El tamaño de la forma se puede cambiar como cualquier ventana de Windows con arrastrar cualquier parte de su contorno.

b) Detalles de las etiquetas.

11.- Crear la etiqueta utilizando la herramienta **Label** y  dar un click donde se desea insertar la etiqueta.

12.- Estando seleccionada la etiqueta ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Label1** y sustituir por **lblNumero1** (no acepta espacios en blanco) para así asignarle nombre a la etiqueta.

13.- Estando la etiqueta seleccionada ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Label1** y sustituir por **&Número 1:** para así agregarle el texto que aparecerá en la etiqueta (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución).

14.- Ir la propiedad **ForeColor** y elegir un color para la etiqueta.

15.- Ir la propiedad **Font** y elegir un Tipo de letra para la etiqueta.

c) Detalles de las cajas de texto.

16.- Para insertar una caja de texto dar click en la **Cuadro de herramientas** en  **Text Box** y dar click en la forma donde se desea insertar la caja de texto.

17.- Estando seleccionada la caja de texto ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **textBox1** y sustituir por **txtNumero1** (no acepta espacios en blanco) para así asignarle nombre a la caja de texto.

18.- Ir la propiedad **BackColor** y elegir un color para la caja de texto.

19.- Ir la propiedad **ForeColor** y elegir un color para el texto que posteriormente aparecerá en la caja de texto.

20.- Ir la propiedad **Font** y elegir un Tipo de letra para el texto que posteriormente aparecerá en la caja de texto.

21.- Repetir los pasos del 11 al 20 para las siguientes etiquetas y caja de texto (crear primero la etiqueta y después su correspondiente caja de texto):

Propiedades	
<i>Name</i>	<i>Text</i>
lblNumero2	N&úmero 2:
txtNumero2	(dejar vacío)
lblDespliegue	(dejar vacío)

d) Detalles de los botones de comando.

22.-Para insertar un botón de comando dar un click en la **Cuadro de herramientas** en  **Button** y dar click en la forma donde se desea insertar el botón de comando.

23.- Estando seleccionado el botón de comando ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Button1** y sustituir por **cmdMayor** (no acepta espacios en blanco) para así asignarle nombre.

24.- Ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Button1** y sustituir por **&Mayor** para así agregarle el texto que aparecerá en el botón (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución). Puedes cambiar tipo de letra color, etc.

25.- Agregar botón de comando repitiendo los pasos 22 al 24 para la salida del programa dejando que también se active con la tecla **Esc**, así como establecer el botón que se ejecutará al presionar

la tecla **Enter**, para lo que se modificarán dos propiedades de la forma **frmPractica11**. Utilizar las siguientes propiedades:

Propiedades			
<i>Name</i>	<i>Text</i>	<i>AcceptButton</i>	<i>CancelButton</i>
cmdSalida	&Salida	-	-
cmdLimpiar	&Limpiar	-	-
frmPractica11	-	cmdMayor	cmdSalida

e) Introducir código.

26.- Insertar el siguiente código para el botón cmdMayor:

```
private void cmdMayor_Click(object sender, EventArgs e)
{
    // Programa que determina el número mayor de dos elementos
    // considerando que pudieran ser iguales
    // Práctica 11

    //Declaración de variables
    int Numero1, Numero2;

    //Asignación de valores a las variables
    Numero1 = System.Int32.Parse(txtNumero1.Text);
    Numero2 = System.Int32.Parse(txtNumero2.Text);

    //Condición anidada
    if (Numero1 == Numero2)
        lblDespliegue.Text = "Son iguales";
    else if(Numero1>Numero2)
        lblDespliegue.Text = Numero1.ToString()+" es mayor que " + Numero2.ToString();
    else
        lblDespliegue.Text = Numero2.ToString()+" es mayor que " + Numero1.ToString();
}
```

27.- Insertar el siguiente código para el botón cmdLimpiar:

```
//Creado automáticamente para el botón cmdLimpiar
private void cmdLimpiar_Click(object sender, EventArgs e)
{
    lblDespliegue.Text = " ";
    txtNumero1.Clear();
    txtNumero2.Clear();
    txtNumero1.Clear();
}
```

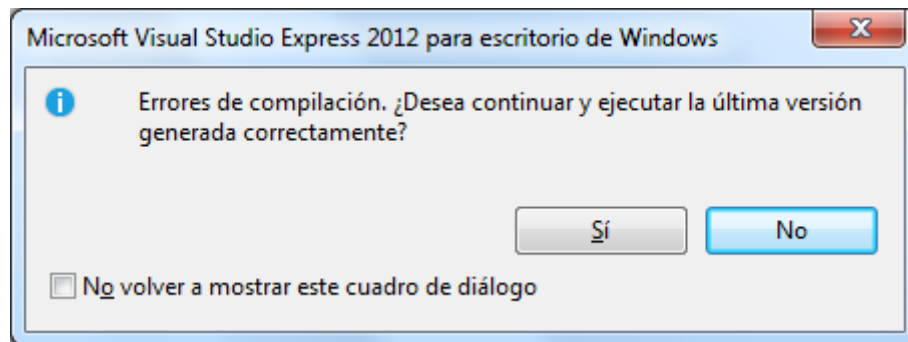
28.- Insertar el siguiente código para el botón cmdSalida:

```
//Creado automáticamente para el botón cmdSalida
private void cmdSalida_Click(object sender, EventArgs e)
{
    Close();
}
```

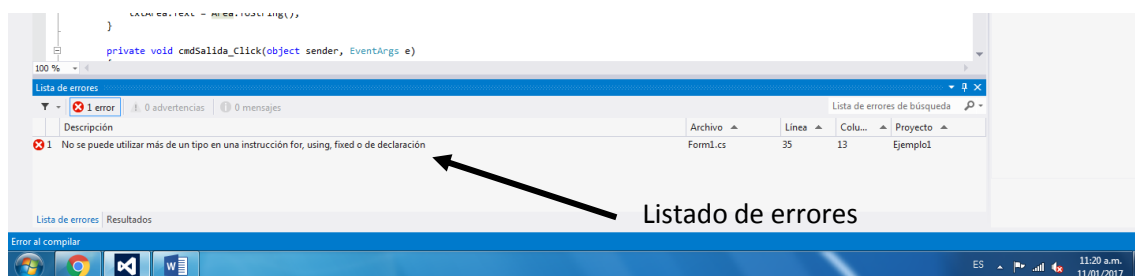
29.- Guardar el programa **Práctica 11**. En caso de que no hayas especificado desde el inicio el nombre y ubicación del programa ir a la opción del menú **ARCHIVO** y posteriormente ir a **Guardar WindowsFormsApplication1 como**. Asegurarse que en el **Explorador de soluciones** se encuentre elegido **WindowsFormsApplication1** para guardar correctamente.

30.- Para compilar y ejecutar el programa dar click en , presionar F5 o ir a la opción del menú **DEPURAR** y elegir **Iniciar depuración**.

31.- En caso que no se ejecute quiere decir que se presentaron errores durante la compilación o traducción y mostrará la siguiente caja:



Dar click en **No** para que permita revisar e identificar los errores de sintaxis y así poder corregirlos. Mostrará una división en la ventana donde se enlistan los errores (los errores varían):



32.- Probar el programa y realizar modificaciones si es necesario.

Notas práctica 11:

- Dentro de la práctica 11 se vieron las clases Form, Label, Text, Button y las instancias de dichas clases a través de los objetos frmPractica11, lblNumero1, lblNumero2, lblDespliegue, txtNumero1, txtNumero2, cmdMayor, cmdLimpiar, cmdSalida, respectivamente, que como ya se dijo, un objeto es una unidad dentro de un programa que consta de un estado (propiedades) y de un comportamiento (métodos), que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.
- Se modificaron las propiedades de Name para los objetos usando prefijos de acuerdo a la clase: Form (prefijo frm), Label (prefijo lbl), Text (prefijo txt), Button (prefijo cmd).
- Se ligaron métodos a los objetos de la clase Button (cmdMayor, cmdSalida y cmdLimpiar).
- En este ejercicio se está utilizando las condiciones anidadas donde una condición se encuentra dentro de otra ya sea en el falso o en el verdadero, esto es, prueba la primera condición y posteriormente prueba la siguiente que se encuentra dentro de la primera. Para esta práctica la primera condición se prueba `if (Numero1 == Numero2)` y si no se cumple se va al falso y prueba la siguiente condición `if(Numero1>Numero2)`.

```
if (Numero1 == Numero2)
    lblDespliegue.Text = "Son iguales";
else if(Numero1>Numero2)
```

- Se introduce el concepto de concatenación de cadenas de caracteres haciendo uso del operador +.

```
lblDespliegue.Text = Numero1.ToString() + " es mayor que " + Numero2.ToString();
```

- Para centrar el foco en un objeto (activar desde código) es necesario especificar el nombre de este y agregar el método **Focus()**.

```
txtNumero1.Focus();
```

- Para limpiar el contenido el objeto caja de texto **Text Box** es necesario especificar el nombre de este y agregar el método **Clear()**.

```
txtNumero1.Clear();
```

PRÁCTICA 12. ASIGNAR EL VALOR NUMÉRICO A LAS CALIFICACIONES DADAS EN LITERAL. A=10, M=9, B=8, R=7, S=6 Y N=5 UTILIZANDO LA SELECTIVA MÚLTIPLE.

OBJETIVO ESPECÍFICO.

Conocer y utilizar la estructura de control “selectiva múltiple” como parte de las estructuras de datos y estructuras lógicas basándose en la POO para desarrollar la lógica de programación por medio de algoritmos y el lenguaje de programación Visual C#.

TEMAS CON LOS QUE SE RELACIONA.

- 4.1 Características del modelo orientado a objetos.
- 4.2 Elementos primordiales en el modelo de objetos.
- 4.3 Representación gráfica del diseño.
- 4.4 Relación entre la programación orientada a objetos y la estructurada.

DESARROLLO DE LA PRÁCTICA.

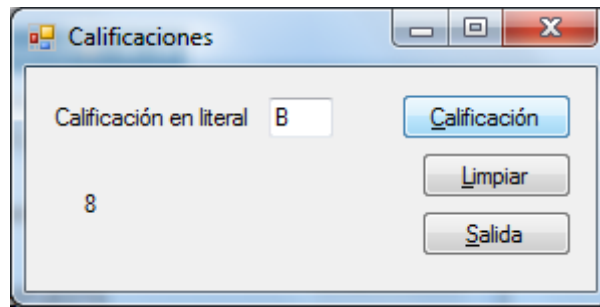
- 1.- Iniciar Visual C#
- 2.- Elegir **Aplicación de Windows Forms**
- 3.- Elegir **Nuevo Proyecto**
- 4.- Elegir **Aplicación de Windows Forms**
- 5.- Dentro de la caja que aparece ir a la sección de Especificaciones de guardado del archivo del programa e introducir lo siguiente:

Nombre: **Práctica 12**

Ubicación: (Ir a **Examinar** y elegir ubicación, usar unidad de almacenamiento móvil)

Nombre de la solución: (Automáticamente se copia el **Nombre**, dejarlo con el mismo)

- 6.- Diseñar la forma, que deberá quedar como se muestra en la siguiente figura:



a) Detalles de la forma.

7.- Dar click en la forma e ir a la **Ventana propiedades**, localizar la propiedad **Name** y dar un click a su derecha donde dice **Form1** y sustituir por **frmPractica12** (no acepta espacios en blanco) para así asignarle nombre a la forma.

8.- Estando la forma seleccionada ir a la **Ventana propiedades**, localizar la propiedad **Text** y dar un click a su derecha donde dice **Form1** y sustituir por **Calificaciones** para así agregarle el título que aparecerá en la forma.

9.- Ir la propiedad **BackColor** y elegir un color para la forma.

10.- El tamaño de la forma se puede cambiar como cualquier ventana de Windows con arrastrar cualquier parte de su contorno.

b) Detalles de las etiquetas.

11.- Crear la etiqueta utilizando la herramienta **Label** y  dar un click donde se desea insertar la etiqueta.

12.- Estando seleccionada la etiqueta ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Label1** y sustituir por **lblCaL** (no acepta espacios en blanco) para así asignarle nombre a la etiqueta.

13.- Estando la etiqueta seleccionada ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Label1** y sustituir por **Calificación en literal** para así agregarle el texto que aparecerá en la etiqueta (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución).

14.- Ir la propiedad **ForeColor** y elegir un color para la etiqueta.

15.- Ir la propiedad **Font** y elegir un Tipo de letra para la etiqueta.

c) Detalles de las cajas de texto.

16.- Para insertar una caja de texto dar click en la **Cuadro de herramientas** en  **Text Box** y dar click en la forma donde se desea insertar la caja de texto.

17.- Estando seleccionada la caja de texto ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **textBox1** y sustituir por **txtCaL** (no acepta espacios en blanco) para así asignarle nombre a la caja de texto.

18.- Seleccionar la caja de texto **txtCaL** e ir a la Propiedad **CharacterCasing** y elegir **Upper** para convertir la literal a mayúscula y así asegurar que aun cuando el usuario introduzca en minúscula al momento de pasar el valor a la variable ya estará en mayúscula.

19.- Ir la propiedad **BackColor** y elegir un color para la caja de texto.

20.- Ir la propiedad **ForeColor** y elegir un color para el texto que posteriormente aparecerá en la caja de texto.

21.- Ir la propiedad **Font** y elegir un Tipo de letra para el texto que posteriormente aparecerá en la caja de texto.

22.- Repetir los pasos del 11 al 15 para las siguientes etiquetas:

Propiedades	
<i>Name</i>	<i>Text</i>
lblCaN	(dejar vacío)
lblError	(dejar vacío)

d) Detalles de los botones de comando.

23.-Para insertar un botón de comando dar un click en la **Cuadro de herramientas** en  **Button** y dar click en la forma donde se desea insertar el botón de comando.

24.- Estando seleccionado el botón de comando ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Button1** y sustituir por **cmdCa** (no acepta espacios en blanco) para así asignarle nombre.

25.- Ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Button1** y sustituir por **&Calificación** para así agregarle el texto que aparecerá en el botón (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución). Puedes cambiar tipo de letra color, etc.

26.- Agregar botón de comando repitiendo los pasos 23 al 25 para la salida del programa dejando que también se active con la tecla **Esc**, así como establecer el botón que se ejecutará al presionar la tecla **Enter**, para lo que se modificarán dos propiedades de la forma **frmPractica12**. Utilizar las siguientes propiedades:

Propiedades			
<i>Name</i>	<i>Text</i>	<i>AcceptButton</i>	<i>CancelButton</i>
cmdSalida	&Salida	-	-
cmdLimpiar	&Limpiar	-	-
frmPractica12	-	cmdCa	cmdSalida

e) Introducir código.

27- Insertar el siguiente código para el botón cmdCa:

```
private void cmdCa_Click(object sender, EventArgs e)
{
    // Programa que asigne el valor numérico a las calificaciones dadas //
    // en literal.
    // A=10, M=9, B=8, R=7, S=6 y N=5
    // Práctica 12

    // Declaración de variables
    int CN;
    char CL;

    //Asignación de valores a variables
    CL = System.Char.Parse(txtCaL.Text);
    CN = 0;

    // Selección múltiple
    switch (CL)
    {
        case 'A':
            CN = 10;
            break;
        case 'M':
            CN = 9;
            break;
        case 'B':
            CN = 8;
            break;
        case 'R':
            CN = 7;
            break;
        case 'S':
            CN = 6;
            break;
        case 'N':
            CN = 5;
            break;
        default:
            lblError.Text = "No válida";
            break;
    }
}
```

```
    }  
  
    lblCaN.Text = CN.ToString();  
}
```

28.- Insertar el siguiente código para el botón cmdLimpia:

```
//Creado automáticamente para el botón cmdLimpia  
private void cmdLimpia_Click(object sender, EventArgs e)  
{  
    lblCaN.Text = " ";  
    lblError.Text = " ";  
}
```

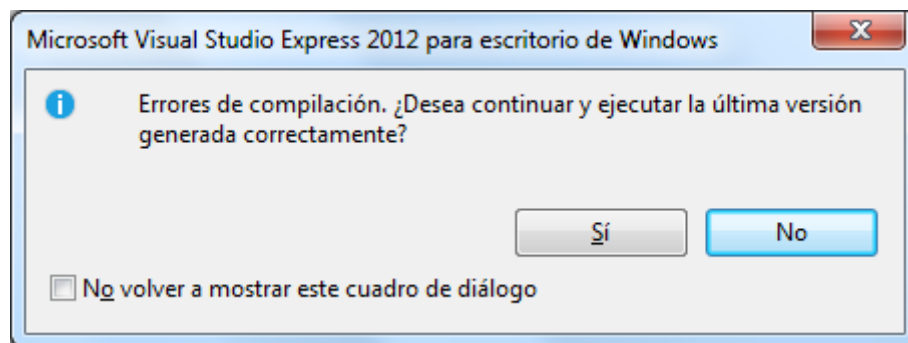
29.- Insertar el siguiente código para el botón cmdSalida:

```
//Creado automáticamente para el botón cmdSalida  
private void cmdSalida_Click(object sender, EventArgs e)  
{  
    Close();  
}
```

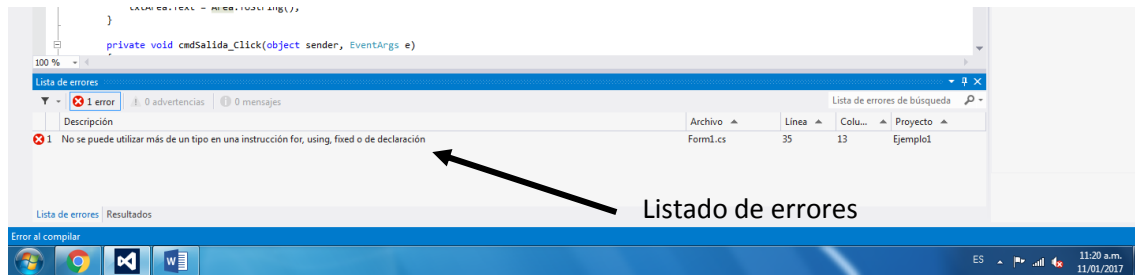
30.- Guardar el programa **Práctica 12**. En caso de que no hayas especificado desde el inicio el nombre y ubicación del programa ir a la opción del menú **ARCHIVO** y posteriormente ir a **Guardar WindowsFormsApplication1 como**. Asegurarse que en el **Explorador de soluciones** se encuentre elegido **WindowsFormsApplication1** para guardar correctamente.

31.- Para compilar y ejecutar el programa dar click en , presionar F5 o ir a la opción del menú **DEPURAR** y elegir **Iniciar depuración**.

32.- En caso que no se ejecute quiere decir que se presentaron errores durante la compilación o traducción y mostrará la siguiente caja:



Dar click en **No** para que permita revisar e identificar los errores de sintaxis y así poder corregirlos. Mostrará una división en la ventana donde se enlistan los errores (los errores varían):



33.- Probar el programa y realizar modificaciones si es necesario.

Notas práctica 12:

- Dentro de la práctica 12 se vieron las clases Form, Label, Text, Button y las instancias de dichas clases a través de los objetos frmPractica12, lblCaL, lblCaN, lblError, txtCaL, cmdCa, cmdLimpiar, cmdSalida, respectivamente, que como ya se dijo, un objeto es una unidad dentro de un programa que consta de un estado (propiedades) y de un comportamiento (métodos), que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.
- Se modificaron las propiedades de Name para los objetos usando prefijos de acuerdo a la clase: Form (prefijo frm), Label (prefijo lbl), Text (prefijo txt), Button (prefijo cmd).
- Se ligaron métodos a los objetos de la clase Button (cmdCa, cmdSalida y cmdLimpiar).
- En esta práctica se utiliza la selectiva anidada **switch()** donde se cuenta con la variable selector **CL** y una serie de casos que irá probando hasta encontrar el que se cumpla.

```
switch (CL)
{
    case 'A':
        CN = 10;
        break;
    case 'M':
        .
        .
        .
}
```

- Si no se cumple alguno de los casos realizará el default.

```
default:
    lblError.Text = "No válida";
    break;
```

- Se modificó en la caja de texto **txtCaL** la propiedad **CharacterCasing** y eligiendo **Upper** para convertir su contenido a mayúscula.

IMPLEMENTACIÓN ORIENTADA A OBJETOS.

COMPETENCIAS A DESARROLLAR.

- Aplica estructuras de datos y estructuras lógicas basándose en la POO para desarrollar la lógica de programación.
- Capacidad de abstracción y análisis.
- Capacidad de aplicar los conocimientos en la práctica.
- Habilidades en el uso de las tecnologías de la información y de la comunicación.
- Habilidad para buscar, procesar y analizar información procedente de fuentes diversas.
- Capacidad para identificar, plantear y resolver problemas.
- Capacidad de trabajar en equipo.
- Habilidades interpersonales.
- Capacidad creativa.
- Habilidad para trabajar en forma autónoma.
- Compromiso ético.
- Compromiso con la calidad.

PRÁCTICA 13. IMPRIME LOS NÚMEROS DEL 1 AL 10 UTILIZANDO EL CICLO DO-WHILE

OBJETIVO ESPECÍFICO.

Conocer y utilizar la estructura de control repetitiva “repetir-mientras” para la solución de problemas del contexto por medio de algoritmos y un lenguaje orientado a objetos.

TEMAS CON LOS QUE SE RELACIONA.

- 5.1 Estructura de una clase.
- 5.2 Elementos de una clase.
- 5.3 Clase principal.
- 5.4 Crear objetos.

DESARROLLO DE LA PRÁCTICA.

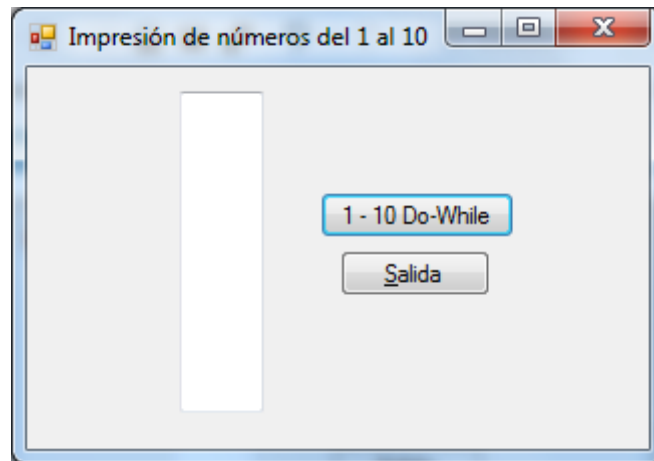
- 1.- Iniciar Visual C#
- 2.- Elegir **Aplicación de Windows Forms**
- 3.- Elegir **Nuevo Proyecto**
- 4.- Elegir **Aplicación de Windows Forms**
- 5.- Dentro de la caja que aparece ir a la sección de Especificaciones de guardado del archivo del programa e introducir lo siguiente:

Nombre: **Práctica 13**

Ubicación: (Ir a **Examinar** y elegir ubicación, usar unidad de almacenamiento móvil)

Nombre de la solución: (Automáticamente se copia el **Nombre**, dejarlo con el mismo)

- 6.- Diseñar la forma, que deberá quedar como se muestra en la siguiente figura:



a) Detalles de la forma.

7.- Dar click en la forma e ir a la **Ventana propiedades**, localizar la propiedad **Name** y dar un click a su derecha donde dice **Form1** y sustituir por **frmPractica13** (no acepta espacios en blanco) para así asignarle nombre a la forma.

8.- Estando la forma seleccionada ir a la **Ventana propiedades**, localizar la propiedad **Text** y dar un click a su derecha donde dice **Form1** y sustituir por **Impresión de números del 1 al 10** para así agregarle el título que aparecerá en la forma.

9.- Ir la propiedad **BackColor** y elegir un color para la forma.

10.- El tamaño de la forma se puede cambiar como cualquier ventana de Windows con arrastrar cualquier parte de su contorno.

b) Detalles de la caja de texto.

11.- Para insertar una caja de texto dar click en la **Cuadro de herramientas** en **abl** **Text Box** y dar click en la forma donde se desea insertar la caja de texto.

12.- Estando seleccionada la caja de texto ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **textBox1** y sustituir por **txtDespliegue** (no acepta espacios en blanco) para así asignarle nombre a la caja de texto.

13.- Para poder desplegar varios datos en la forma es necesario hacer lo siguiente: Ir a la propiedad **Multiline** del objeto Caja de Texto **txtDespliegue** y activar **True** para que permita varias líneas en la caja.

14.- Verificar que la propiedad **WordWrap** esté también en **True**.

15.- Ir a **TextAlign** y seleccionar **Center** para centrar el texto en la caja de texto **txtDespliegue**.

16.- Para que el usuario no pueda modificar la salida de datos ir a la propiedad **ReadOnly** del objeto Caja de Texto **txtDespliegue** y activar **True**.

c) Detalles de los botones de comando.

17.- Para insertar un botón de comando dar un click en la **Cuadro de herramientas** en  **Button** y dar click en la forma donde se desea insertar el botón de comando.

18.- Estando seleccionado el botón de comando ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Button1** y sustituir por **cmdDespliegue** (no acepta espacios en blanco) para así asignarle nombre.

19.- Ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Button1** y sustituir por **1-10 Do While** para así agregarle el texto que aparecerá en el botón. Puedes cambiar tipo de letra color, etc.

20.- Agregar botón de comando repitiendo los pasos 17 al 19 para la salida del programa dejando que también se active con la tecla **Esc**, así como establecer el botón que se ejecutará al presionar la tecla **Enter**, para lo que se modificarán dos propiedades de la forma **frmPractica18**. Utilizar las siguientes propiedades:

Propiedades			
<i>Name</i>	<i>Text</i>	<i>AcceptButton</i>	<i>CancelButton</i>
cmdSalida	&Salida	-	-
frmPractica13	-	cmdDespliegue	cmdSalida

d) Introducir código.

21.- Agregar el siguiente código al botón **cmdDespliegue**:

```
private void cmdDespliegue_Click(object sender, EventArgs e)
{
    //Programa que imprime los números del 1 al 10
    //utilizando el ciclo Do-While
    //Práctica 13

    //Declaración de variables
    int C=0;

    //Despliegue de resultados
    do
    {
        C = C + 1;
        txtDespliegue.Text = txtDespliegue.Text+C.ToString()+"\r\n";
    } while (C < 10);
}
```

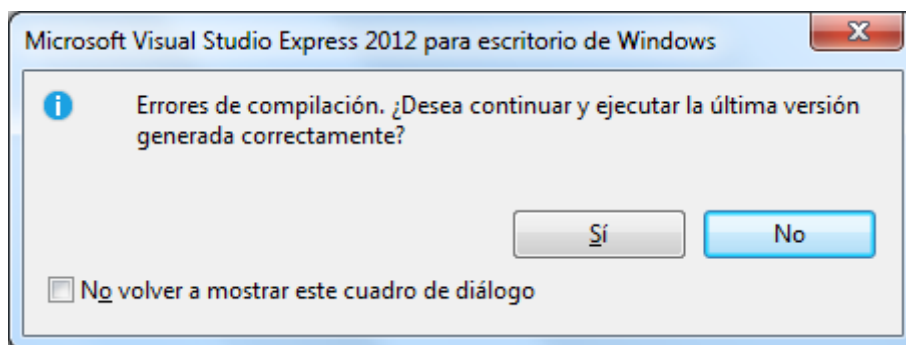
22.- Agregar el siguiente código al botón **cmdSalida**:

```
private void cmdSalida_Click(object sender, EventArgs e)
{
    Close();
}
```

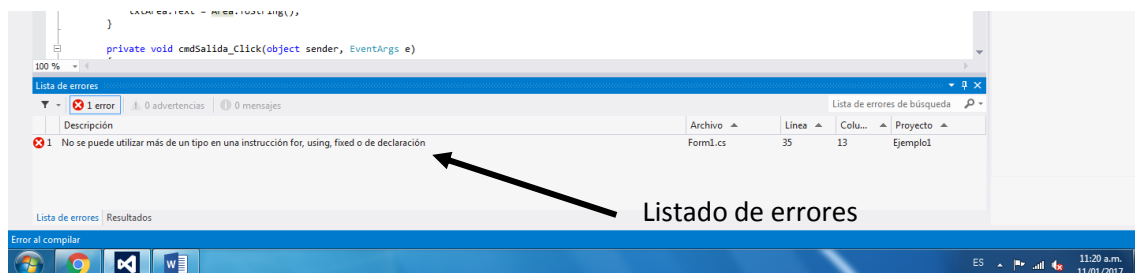
23.- Guardar el programa **Práctica 13**. En caso de que no hayas especificado desde el inicio el nombre y ubicación del programa ir a la opción del menú **ARCHIVO** y posteriormente ir a **Guardar WindowsFormsApplication1 como**. Asegurarse que en el **Explorador de soluciones** se encuentre elegido **WindowsFormsApplication1** para guardar correctamente.

24.- Para compilar y ejecutar el programa dar click en , presionar F5 o ir a la opción del menú **DEPURAR** y elegir **Iniciar depuración**.

25.- En caso que no se ejecute quiere decir que se presentaron errores durante la compilación o traducción y mostrará la siguiente caja:



Dar click en **No** para que permita revisar e identificar los errores de sintaxis y así poder corregirlos. Mostrará una división en la ventana donde se enlistan los errores (los errores varían):



26.- Probar el programa y realizar modificaciones si es necesario.

Notas práctica 13:

- Dentro de la práctica 13 se vieron las clases Form, Text, Button y las instancias de dichas clases a través de los objetos frmPractica13, txtDespliegue, cmdDespliegue, cmdSalida, respectivamente, que como ya se dijo, un objeto es una unidad dentro de un programa que consta de un estado (propiedades) y de un comportamiento (métodos), que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.
- Se modificaron las propiedades de Name para los objetos usando prefijos de acuerdo a la clase: Form (prefijo frm), Text (prefijo txt), Button (prefijo cmd).
- Se ligaron métodos a los objetos de la clase Button (cmdDespliegue, cmdSalida).
- En esta práctica se introduce el uso de la **estructura de repetición o ciclo** que consiste en un segmento de un algoritmo o programa cuyas instrucciones se repetirán un número determinado de veces mientras una la condición establecida es verdadera.
- En esta práctica se introduce el uso de un **contador** que consiste en una variable cuyo valor se va a incrementar o decrementar en una cantidad constante en cada iteración o repetición, en este caso la variable **C** incrementará en **1** su valor de manera constante en cada iteración.

$C = C + 1;$

- En esta práctica se utiliza una de las estructuras de repetición llamada **Do – While** o Repetir-Mientras, es conocido también como ciclo que prueba por abajo ya que por lo menos ejecuta una vez el bloque de instrucciones antes de probar la condición para continuar o salir del ciclo, en este caso se mantendrá iterando mientras **C < 10**.

```
do
{
    C = C + 1;
    txtDespliegue.Text = txtDespliegue.Text+C.ToString()+"\r\n";
} while (C < 10);
```

- Una característica muy importante del ciclo Do-While es que se mantendrá iterando mientras la condición sea verdadera, esto significa que cuando la condición sea falsa saldrá del ciclo. La condición que se defina para detener el ciclo puede manejar cualquier tipo de valor, ya sea numérico o alfanumérico.

PRÁCTICA 14. IMPRIME LOS NÚMEROS DEL 1 AL 10 UTILIZANDO EL CICLO WHILE.

OBJETIVO ESPECÍFICO.

Conocer y utilizar la estructura de control repetitiva “mientras” para la solución de problemas del contexto por medio de algoritmos y un lenguaje orientado a objetos.

TEMAS CON LOS QUE SE RELACIONA.

- 5.1 Estructura de una clase.
- 5.2 Elementos de una clase.
- 5.3 Clase principal.
- 5.4 Crear objetos.

DESARROLLO DE LA PRÁCTICA.

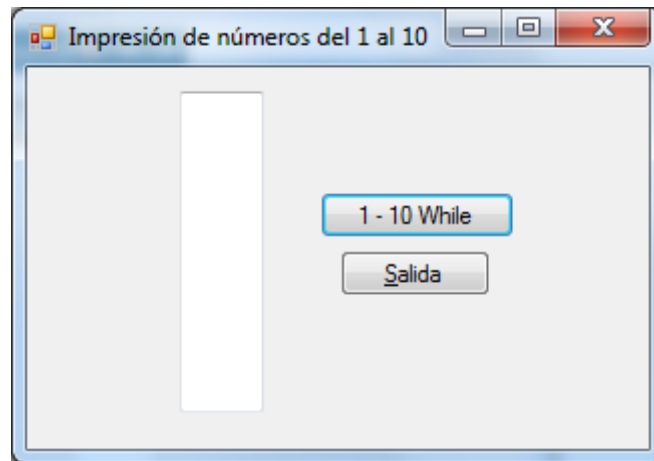
- 1.- Iniciar Visual C#
- 2.- Elegir **Aplicación de Windows Forms**
- 3.- Elegir **Nuevo Proyecto**
- 4.- Elegir **Aplicación de Windows Forms**
- 5.- Dentro de la caja que aparece ir a la sección de Especificaciones de guardado del archivo del programa e introducir lo siguiente:

Nombre: **Práctica 14**

Ubicación: (Ir a **Examinar** y elegir ubicación, usar unidad de almacenamiento móvil)

Nombre de la solución: (Automáticamente se copia el **Nombre**, dejarlo con el mismo)

- 6.- Diseñar la forma, que deberá quedar como se muestra en la siguiente figura:



a) Detalles de la forma.

7.- Dar click en la forma e ir a la **Ventana propiedades**, localizar la propiedad **Name** y dar un click a su derecha donde dice **Form1** y sustituir por **frmPractica14** (no acepta espacios en blanco) para así asignarle nombre a la forma.

8.- Estando la forma seleccionada ir a la **Ventana propiedades**, localizar la propiedad **Text** y dar un click a su derecha donde dice **Form1** y sustituir por **Impresión de números del 1 al 10** para así agregarle el título que aparecerá en la forma.

9.- Ir la propiedad **BackColor** y elegir un color para la forma.

10.- El tamaño de la forma se puede cambiar como cualquier ventana de Windows.

b) Detalles de la caja de texto.

11.- Para insertar una caja de texto dar click en la **Cuadro de herramientas** en  **Text Box** y dar click en la forma donde se desea insertar la caja de texto.

12.- Estando seleccionada la Caja de Texto ir a las propiedades y modificar **Name** dando un click y sustituir **textBox1** por **txtDespliegue** para así asignarle nombre a la caja de texto.

13.- Para desplegar varias líneas de datos en la caja de texto es necesario ir a la propiedad **Multiline** del objeto Caja de Texto **txtDespliegue** y activar **True**.

14.- Verificar que la propiedad **WordWrap** esté también en **True**.

15.- Ir a **TextAlign** y seleccionar **Center** para centrar el texto en la caja de texto **txtDespliegue**.

16.- Para que el usuario no pueda modificar la salida de datos ir a la propiedad **ReadOnly** del objeto Caja de Texto **txtDespliegue** y activar **True**.

c) Detalles de los botones de comando.

17.- Para insertar un botón de comando dar un click en la **Cuadro de herramientas** en  **Button** y dar click en la forma donde se desea insertar el botón de comando.

18.- Estando seleccionado el botón de comando ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Button1** y sustituir por **cmdDespliegue** (no acepta espacios en blanco) para así asignarle nombre.

19.- Ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Button1** y sustituir por **1-10 While** para así agregarle el texto que aparecerá en el botón. Puedes cambiar tipo de letra color, etc.

20.- Agregar botón de comando repitiendo los pasos 17 al 19 para la salida del programa dejando que también se active con la tecla **Esc**, así como establecer el botón que se ejecutará al presionar la tecla **Enter**, para lo que se modificarán dos propiedades de la forma **frmPractica14**. Utilizar las siguientes propiedades:

Propiedades			
<i>Name</i>	<i>Text</i>	<i>AcceptButton</i>	<i>CancelButton</i>
cmdSalida	&Salida	-	-
frmPractica14	-	cmdDespliegue	cmdSalida

d) Introducir código.

21.- Agregar el siguiente código al botón **cmdDespliegue**:

```
private void cmdDespliegue_Click(object sender, EventArgs e)
{
    //Programa que imprime los números del 1 al 10 utilizando el ciclo While
    //Práctica 14

    //Declaración de variables
    int C=0;

    //Despliegue de resultados utilizando el ciclo While
    while (C < 10)
    {
        C = C + 1;
        txtDespliegue.Text = txtDespliegue.Text+C.ToString()+"\r\n";
    }
}
```

22.- Agregar el siguiente código al botón **cmdSalida**:

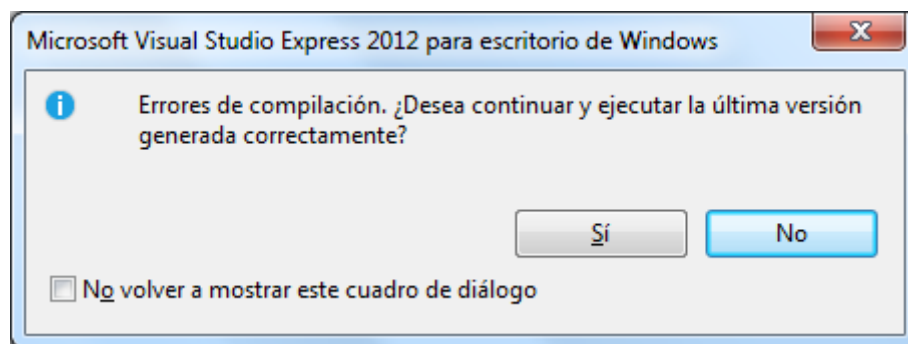
```
private void cmdSalida_Click(object sender, EventArgs e)
{
```

```
        Close();  
    }
```

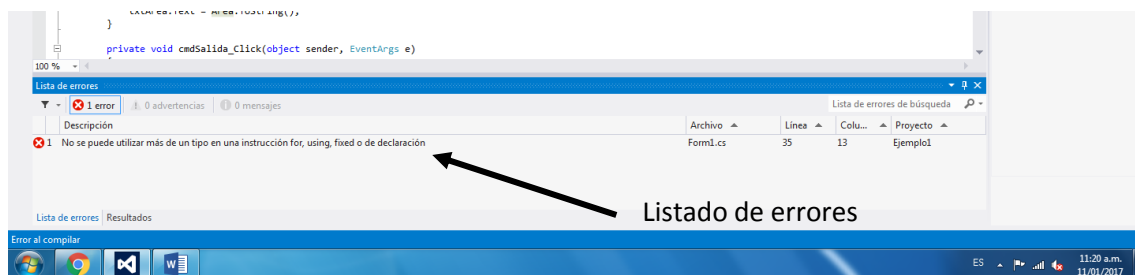
23.- Guardar el programa **Práctica 14**. En caso de que no hayas especificado desde el inicio el nombre y ubicación del programa ir a la opción del menú **ARCHIVO** y posteriormente ir a **Guardar WindowsFormsApplication1 como**. Asegurarse que en el **Explorador de soluciones** se encuentre elegido **WindowsFormsApplication1** para guardar correctamente.

24.- Para compilar y ejecutar el programa dar click en , presionar F5 o ir a la opción del menú **DEPURAR** y elegir **Iniciar depuración**.

25.- En caso que no se ejecute quiere decir que se presentaron errores durante la compilación o traducción y mostrará la siguiente caja:



Dar click en **No** para que permita revisar e identificar los errores de sintaxis y así poder corregirlos. Mostrará una división en la ventana donde se enlistan los errores (los errores varían):



26.- Probar el programa y realizar modificaciones si es necesario.

Notas práctica 14:

- Dentro de la práctica 14 se vieron las clases Form, Text, Button y las instancias de dichas clases a través de los objetos frmPractica14, txtDespliegue, cmdDespliegue, cmdSalida, respectivamente, que como ya se dijo, un objeto es una unidad dentro de un programa

que consta de un estado (propiedades) y de un comportamiento (métodos), que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.

- Se modificaron las propiedades de Name para los objetos usando prefijos de acuerdo a la clase: Form (prefijo frm), Text (prefijo txt), Button (prefijo cmd).
- Se ligaron métodos a los objetos de la clase Button (cmdDespliegue, cmdSalida).
- En esta práctica se hace uso de la **estructura de repetición o ciclo** que consiste en un segmento de un algoritmo o programa cuyas instrucciones se repetirán un número determinado de veces mientras una la condición establecida es verdadera.
- En esta práctica se hace uso de un **contador** que consiste en una variable cuyo valor se va a incrementar o decrementar en una cantidad constante en cada iteración o repetición, en este caso la variable **C** incrementará en **1** su valor de manera constante en cada iteración.

$$C = C + 1;$$

- En esta práctica se utiliza una de las estructuras de repetición llamada **While** o Mientras, es conocido también como ciclo que prueba por arriba ya que ejecuta el bloque de instrucciones sólo si al probar la condición al inicio ésta es verdadera, en este caso se mantendrá iterando mientras **C < 10**.

```
while (C < 10)
{
    C = C + 1;
    txtDespliegue.Text = txtDespliegue.Text+C.ToString()+"\r\n";
}
```

- Una característica muy importante del ciclo **While** es que se mantendrá iterando mientras la condición sea verdadera, esto significa que cuando la condición sea falsa saldrá del ciclo. La condición que se defina para detener el ciclo puede manejar cualquier tipo de valor, ya sea numérico o alfanumérico.

PRÁCTICA 15. IMPRIME LOS NÚMERO DEL 1 AL 10 UTILIZANDO EL CICLO FOR.

OBJETIVO ESPECÍFICO.

Conocer y utilizar la estructura de control repetitiva “Repetir-Desde” para la solución de problemas del contexto por medio de algoritmos y un lenguaje orientado a objetos.

TEMAS CON LOS QUE SE RELACIONA.

- 5.1 Estructura de una clase.
- 5.2 Elementos de una clase.
- 5.3 Clase principal.
- 5.4 Crear objetos.

DESARROLLO DE LA PRÁCTICA.

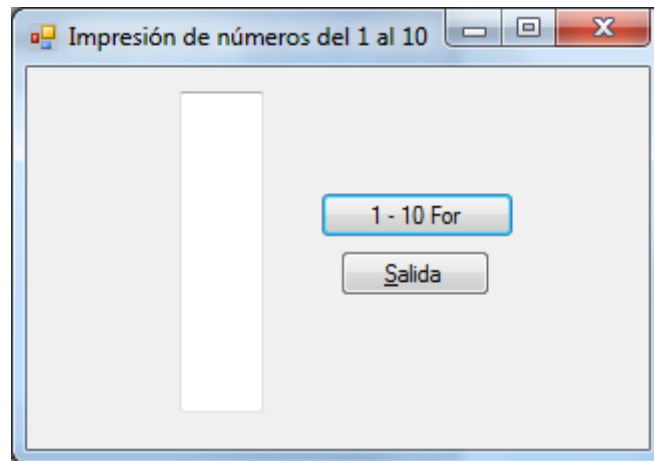
- 1.- Iniciar Visual C#
- 2.- Elegir **Aplicación de Windows Forms**
- 3.- Elegir **Nuevo Proyecto**
- 4.- Elegir **Aplicación de Windows Forms**
- 5.- Dentro de la caja que aparece ir a la sección de Especificaciones de guardado del archivo del programa e introducir lo siguiente:

Nombre: **Práctica 15**

Ubicación: (Ir a **Examinar** y elegir ubicación, usar unidad de almacenamiento móvil)

Nombre de la solución: (Automáticamente se copia el **Nombre**, dejarlo con el mismo)

- 6.- Diseñar la forma, que deberá quedar como se muestra en la siguiente figura:



a) Detalles de la forma.

7.- Dar click en la forma e ir a la **Ventana propiedades**, localizar la propiedad **Name** y dar un click a su derecha donde dice **Form1** y sustituir por **frmPractica15** (no acepta espacios en blanco) para así asignarle nombre a la forma.

8.- Estando la forma seleccionada ir a la **Ventana propiedades**, localizar la propiedad **Text** y dar un click a su derecha donde dice **Form1** y sustituir por **Impresión de números del 1 al 10** para así agregarle el título que aparecerá en la forma.

9.- Ir la propiedad **BackColor** y elegir un color para la forma.

10.- El tamaño de la forma se puede cambiar como cualquier ventana de Windows.

b) Detalles de la caja de texto.

11.- Para insertar una caja de texto dar click en la **Cuadro de herramientas** en  **Text Box** y dar click en la forma donde se desea insertar la caja de texto.

12.- Estando seleccionada la Caja de Texto ir a las propiedades y modificar **Name** dando un click y sustituir **textBox1** por **txtDespliegue** para así asignarle nombre a la caja de texto.

13.- Para desplegar varias líneas de datos en la caja de texto es necesario ir a la propiedad **Multiline** del objeto Caja de Texto **txtDespliegue** y activar **True**.

14.- Verificar que la propiedad **WordWrap** esté también en **True**.

15.- Ir a **TextAlign** y seleccionar **Center** para centrar el texto en la caja de texto **txtDespliegue**.

16.- Para que el usuario no pueda modificar la salida de datos ir a la propiedad **ReadOnly** del objeto Caja de Texto **txtDespliegue** y activar **True**.

c) Detalles de los botones de comando.

17.- Para insertar un botón de comando dar un click en la **Cuadro de herramientas** en  **Button** y dar click en la forma donde se desea insertar el botón de comando.

18.- Estando seleccionado el botón de comando ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Button1** y sustituir por **cmdDespliegue** (no acepta espacios en blanco) para así asignarle nombre.

19.- Ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Button1** y sustituir por **1-10 For** para así agregarle el texto que aparecerá en el botón. Puedes cambiar tipo de letra color, etc.

20.- Agregar botón de comando repitiendo los pasos 17 al 19 para la salida del programa dejando que también se active con la tecla **Esc**, así como establecer el botón que se ejecutará al presionar la tecla **Enter**, para lo que se modificarán dos propiedades de la forma **frmPractica15**. Utilizar las siguientes propiedades:

Propiedades			
Name	Text	AcceptButton	CancelButton
cmdSalida	&Salida	-	-
FrmPractica15	-	cmdDespliegue	cmdSalida

d) Introducir código.

21.- Agregar el siguiente código al botón **cmdDespliegue**:

```
private void cmdDespliegue_Click(object sender, EventArgs e)
{
    //Programa que imprime los números del 1 al 10 utilizando el ciclo For
    //Práctica 15

    //Declaración de variables
    int C;

    //Despliegue de resultados utilizando el ciclo For
    for (C = 1; C <= 10; C = C + 1)
    {
        txtDespliegue.Text = txtDespliegue.Text + C.ToString() + "\r\n";
    }
}
```

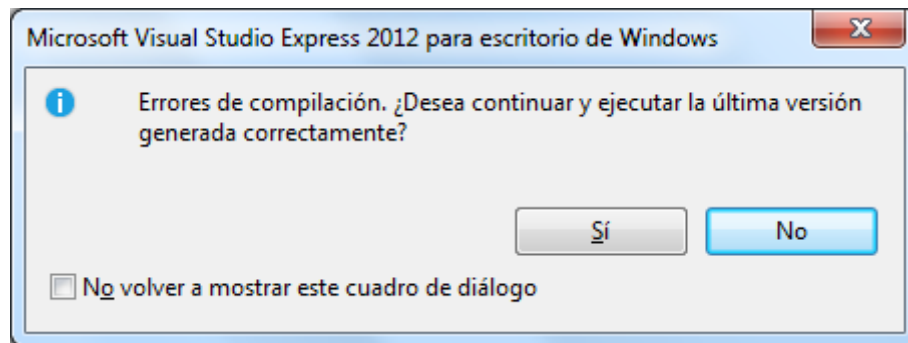
22.- Agregar el siguiente código al botón **cmdSalida**:

```
private void cmdSalida_Click(object sender, EventArgs e)
{
    Close();
}
```

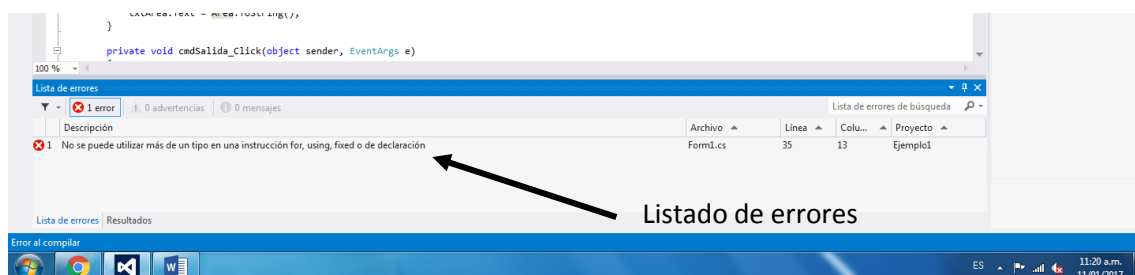
23.- Guardar el programa **Práctica 15**. En caso de que no hayas especificado desde el inicio el nombre y ubicación del programa ir a la opción del menú **ARCHIVO** y posteriormente ir a **Guardar WindowsFormsApplication1 como**. Asegurarse que en el **Explorador de soluciones** se encuentre elegido **WindowsFormsApplication1** para guardar correctamente.

24.- Para compilar y ejecutar el programa dar click en , presionar F5 o ir a la opción del menú **DEPURAR** y elegir **Iniciar depuración**.

25.- En caso que no se ejecute quiere decir que se presentaron errores durante la compilación o traducción y mostrará la siguiente caja:



Dar click en **No** para que permita revisar e identificar los errores de sintaxis y así poder corregirlos. Mostrará una división en la ventana donde se enlistan los errores (los errores varían):



26.- Probar el programa y realizar modificaciones si es necesario.

Notas práctica 15:

- Dentro de la práctica 15 se vieron las clases Form, Text, Button y las instancias de dichas clases a través de los objetos frmPractica15, txtDespliegue, cmdDespliegue, cmdSalida, respectivamente, que como ya se dijo, un objeto es una unidad dentro de un programa que consta de un estado (propiedades) y de un comportamiento (métodos), que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.
- Se modificaron las propiedades de Name para los objetos usando prefijos de acuerdo a la clase: Form (prefijo frm), Text (prefijo txt), Button (prefijo cmd).
- Se ligaron métodos a los objetos de la clase Button (cmdDespliegue, cmdSalida).
- En esta práctica se hace uso de la **estructura de repetición o ciclo** que consiste en un segmento de un algoritmo o programa cuyas instrucciones se repetirán un número determinado de veces mientras una la condición establecida es verdadera.
- En esta práctica se utiliza una de las estructuras de repetición llamada Repetir-Desde o **For**, es el más sencillo de los ciclos y es conveniente utilizarlo cuando ya se conoce el número de iteraciones que deberá realizar, en caso contrario se deberá utilizar otro tipo de ciclo donde no sea necesario indicar desde el principio las veces que iterará, en este caso el contador se inicializa **C = 1** únicamente la primera vez, se mantendrá iterando mientras **C <= 10** y el incremento del contador será **C = C + 1** en cada iteración.

`for (C = 1; C <= 10; C = C + 1)`

- Una característica muy importante del ciclo **For** es que se mantendrá iterando mientras la condición sea verdadera, esto significa que cuando la condición sea falsa saldrá del ciclo. La condición que se defina para detener el ciclo debe ser tipo de valor numérico.
- Como se puede observar, en esta práctica se hace uso de un **contador** que como ya se sabe consiste en una variable cuyo valor se va a incrementar o decrementar en una cantidad constante en cada iteración o repetición, en este caso la variable **C** incrementará en **1** su valor de manera constante en cada iteración.

`C = C + 1;`

- En C# también se puede utilizar un contador con incrementos en 1 como **C++** en lugar de **C = C + 1**, en el caso de incrementos diferentes a 1 es necesario utilizar **C += 2** para incrementos de 2 en 2, **C += 3** para incrementos de 3 en 3, etc. en lugar de utilizar **C = C + 2** o **C = C + 3**.
- Para el caso de un contador en decrementos de 1 en 1 se puede usar como **C--** en lugar de **C = C - 1** y para los decrementos diferentes a 1 es necesario utilizar **C -= 2** para decrementos de 2 en 2, **C -= 3** para decrementos de 3 en 3, etc. en lugar de utilizar el formato **C = C - 2** o **C = C - 3**.

PRÁCTICA 16. PROGRAMA QUE IMPRIME LOS NÚMEROS PARES ENTRE 1 Y 40 UTILIZANDO EL CICLO FOR.

OBJETIVO ESPECÍFICO.

Conocer y utilizar la estructura de control repetitiva “Repetir-Desde” para la solución de problemas del contexto por medio de algoritmos y un lenguaje orientado a objetos.

TEMAS CON LOS QUE SE RELACIONA.

- 5.1 Estructura de una clase.
- 5.2 Elementos de una clase.
- 5.3 Clase principal.
- 5.4 Crear objetos.

DESARROLLO DE LA PRÁCTICA.

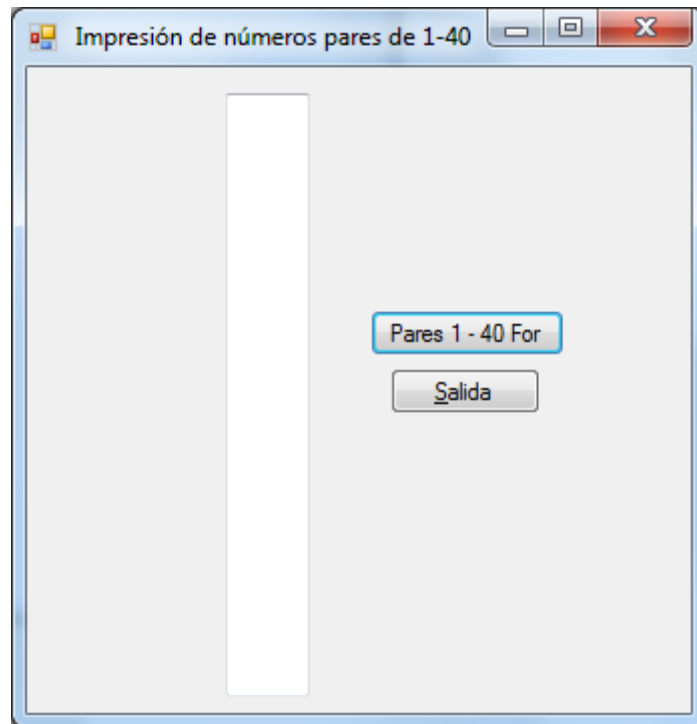
- 1.- Iniciar Visual C#
- 2.- Elegir **Aplicación de Windows Forms**
- 3.- Elegir **Nuevo Proyecto**
- 4.- Elegir **Aplicación de Windows Forms**
- 5.- Dentro de la caja que aparece ir a la sección de Especificaciones de guardado del archivo del programa e introducir lo siguiente:

Nombre: **Práctica 16**

Ubicación: (Ir a **Examinar** y elegir ubicación, usar unidad de almacenamiento móvil)

Nombre de la solución: (Automáticamente se copia el **Nombre**, dejarlo con el mismo)

- 6.- Diseñar la forma, que deberá quedar como se muestra en la siguiente figura:



a) Detalles de la forma.

7.- Dar click en la forma e ir a la **Ventana propiedades**, localizar la propiedad **Name** y dar un click a su derecha donde dice **Form1** y sustituir por **frmPractica16** (no acepta espacios en blanco) para así asignarle nombre a la forma.

8.- Estando la forma seleccionada ir a la **Ventana propiedades**, localizar la propiedad **Text** y dar un click a su derecha donde dice **Form1** y sustituir por **Impresión de números pares de 1 al 40** para así agregarle el título que aparecerá en la forma.

9.- Ir la propiedad **BackColor** y elegir un color para la forma.

10.- El tamaño de la forma se puede cambiar como cualquier ventana de Windows.

b) Detalles de la caja de texto.

11.- Para insertar una caja de texto dar click en la **Cuadro de herramientas** en **Text Box** y dar click en la forma donde se desea insertar la caja de texto.

12.- Estando seleccionada la Caja de Texto ir a las propiedades y modificar **Name** dando un click y sustituir **textBox1** por **txtDespliegue** para así asignarle nombre a la caja de texto.

13.- Para desplegar varias líneas de datos en la caja de texto es necesario ir a la propiedad **Multiline** del objeto Caja de Texto **txtDespliegue** y activar **True**.

14.- Verificar que la propiedad **WordWrap** esté también en **True**.

15.- Ir a **TextAlign** y seleccionar **Center** para centrar el texto en la caja de texto **txtDespliegue**.

16.- Para que el usuario no pueda modificar la salida de datos ir a la propiedad **ReadOnly** del objeto Caja de Texto **txtDespliegue** y activar **True**.

c) Detalles de los botones de comando.

17.-Para insertar un botón de comando dar un click en la **Cuadro de herramientas** en  **Button** y dar click en la forma donde se desea insertar el botón de comando.

18.- Estando seleccionado el botón de comando ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Button1** y sustituir por **cmdDespliegue** (no acepta espacios en blanco) para así asignarle nombre.

19.- Ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Button1** y sustituir por **Pares 1-40 For** para así agregarle el texto que aparecerá en el botón. Puedes cambiar tipo de letra color, etc.

20.- Agregar botón de comando repitiendo los pasos 17 al 19 para la salida del programa dejando que también se active con la tecla **Esc**, así como establecer el botón que se ejecutará al presionar la tecla **Enter**, para lo que se modificarán dos propiedades de la forma **frmPractica16**. Utilizar las siguientes propiedades:

Propiedades			
<i>Name</i>	<i>Text</i>	<i>AcceptButton</i>	<i>CancelButton</i>
cmdSalida	&Salida	-	-
frmPractica16	-	cmdDespliegue	cmdSalida

d) Introducir código.

21.- Agregar el siguiente código al botón **cmdDespliegue**:

```
private void cmdDespliegue_Click(object sender, EventArgs e)
{
    //Programa que imprime números pares entre 1 y 40 utilizando el ciclo For
    //Práctica 16

    //Declaración de variables
    int C;

    //Despliegue de resultados utilizando el ciclo For
    for (C = 2; C <= 40;C=C+2)
    {
        txtDespliegue.Text = txtDespliegue.Text + C.ToString() + "\r\n";
    }
}
```

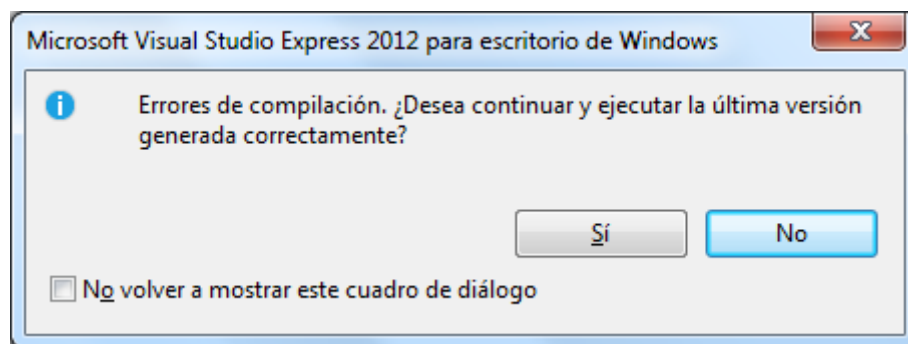
22.- Agregar el siguiente código al botón **cmdSalida**:

```
private void cmdSalida_Click(object sender, EventArgs e)
{
    Close();
}
```

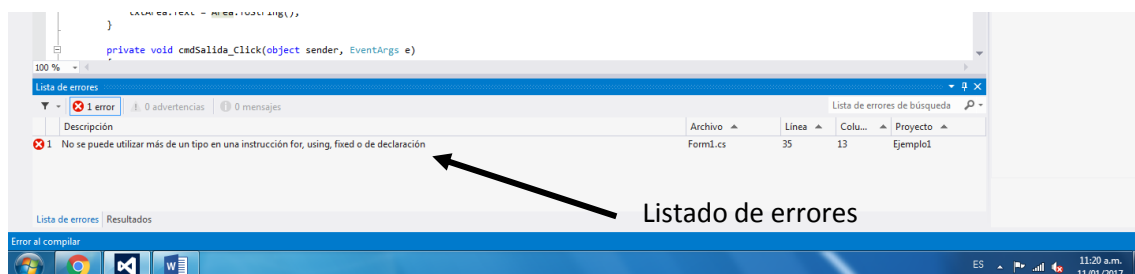
23.- Guardar el programa **Práctica 16**. En caso de que no hayas especificado desde el inicio el nombre y ubicación del programa ir a la opción del menú **ARCHIVO** y posteriormente ir a **Guardar WindowsFormsApplication1 como**. Asegurarse que en el **Explorador de soluciones** se encuentre elegido **WindowsFormsApplication1** para guardar correctamente.

24.- Para compilar y ejecutar el programa dar click en , presionar F5 o ir a la opción del menú **DEPURAR** y elegir **Iniciar depuración**.

25.- En caso que no se ejecute quiere decir que se presentaron errores durante la compilación o traducción y mostrará la siguiente caja:



Dar click en **No** para que permita revisar e identificar los errores de sintaxis y así poder corregirlos. Mostrará una división en la ventana donde se enlistan los errores (los errores varían):



26.- Probar el programa y realizar modificaciones si es necesario.

Notas práctica 16:

- Dentro de la práctica 16 se vieron las clases Form, Text, Button y las instancias de dichas clases a través de los objetos frmPractica16, txtDespliegue, cmdDespliegue, cmdSalida, respectivamente, que como ya se dijo, un objeto es una unidad dentro de un programa que consta de un estado (propiedades) y de un comportamiento (métodos), que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.
- Se modificaron las propiedades de Name para los objetos usando prefijos de acuerdo a la clase: Form (prefijo frm), Text (prefijo txt), Button (prefijo cmd).
- Se ligaron métodos a los objetos de la clase Button (cmdDespliegue, cmdSalida).
- En esta práctica se hace uso de la **estructura de repetición o ciclo** que consiste en un segmento de un algoritmo o programa cuyas instrucciones se repetirán un número determinado de veces mientras una la condición establecida es verdadera.
- En esta práctica se utiliza una de las estructuras de repetición llamada Repetir-Desde o **For**, es el más sencillo de los ciclos y es conveniente utilizarlo cuando ya se conoce el número de iteraciones que deberá realizar, en caso contrario se deberá utilizar otro tipo de ciclo donde no sea necesario indicar desde el principio las veces que iterará, en este caso el contador se inicializa **C = 2** únicamente la primera vez, se mantendrá iterando mientras **C <= 40** sea verdadera y el incremento del contador será **C = C + 2** en cada iteración.

`for (C = 2; C <= 40; C = C + 2)`

- Una característica muy importante del ciclo **For** es que se mantendrá iterando mientras la condición sea verdadera, esto significa que cuando la condición sea falsa saldrá del ciclo. La condición que se defina para detener el ciclo debe ser tipo de valor numérico.
- En esta práctica se hace uso de un **contador** que como ya se consiste en una variable cuyo valor se va a incrementar o decrementar en una cantidad constante en cada iteración o repetición, en este caso la variable **C** incrementará en **2** su valor de manera constante en cada iteración.

`C = C + 2;`

- En C# también se puede utilizar un contador con incrementos en 1 como **C++** en lugar de **C = C + 1**, en el caso de incrementos diferentes a 1 es necesario utilizar **C += 2** para incrementos de 2 en 2, **C += 3** para incrementos de 3 en 3, etc. en lugar de utilizar **C = C + 2** o **C = C + 3**.
- Para el caso de un contador en decrementos de 1 en 1 se puede usar como **C--** en lugar de **C = C - 1** y para los decrementos diferentes a 1 es necesario utilizar **C -= 2** para decrementos de 2 en 2, **C -= 3** para decrementos de 3 en 3, etc. en lugar de utilizar el formato **C = C - 2** o **C = C - 3**.

IMPLEMENTACIÓN ORIENTADA A OBJETOS.

COMPETENCIAS A DESARROLLAR.

- Implementa métodos para diseñar módulos independientes y robustos y que correspondan a soluciones en el mundo real.
- Capacidad de abstracción y análisis.
- Capacidad de aplicar los conocimientos en la práctica.
- Habilidades en el uso de las tecnologías de la información y de la comunicación.
- Habilidad para buscar, procesar y analizar información procedente de fuentes diversas.
- Capacidad para identificar, plantear y resolver problemas.
- Capacidad de trabajar en equipo.
- Habilidades interpersonales.
- Capacidad creativa.
- Habilidad para trabajar en forma autónoma.
- Compromiso ético.
- Compromiso con la calidad.

PRÁCTICA 17. SUMA DE DOS NÚMEROS UTILIZANDO FUNCIONES QUE NO RETORNAN VALOR (PROCEDIMIENTO) Y PASE DE PARÁMETROS

OBJETIVO ESPECÍFICO.

Implementar métodos para diseñar módulos independientes y robustos y que correspondan a soluciones en el mundo real.

TEMAS CON LOS QUE SE RELACIONA.

6.1 Declaración de métodos.

6.2 Métodos de clase.

6.3 Métodos de instancia.

DESARROLLO DE LA PRÁCTICA.

1.- Iniciar Visual C#

2.- Elegir **Aplicación de Windows Forms**

3.- Elegir **Nuevo Proyecto**

4.- Elegir **Aplicación de Windows Forms**

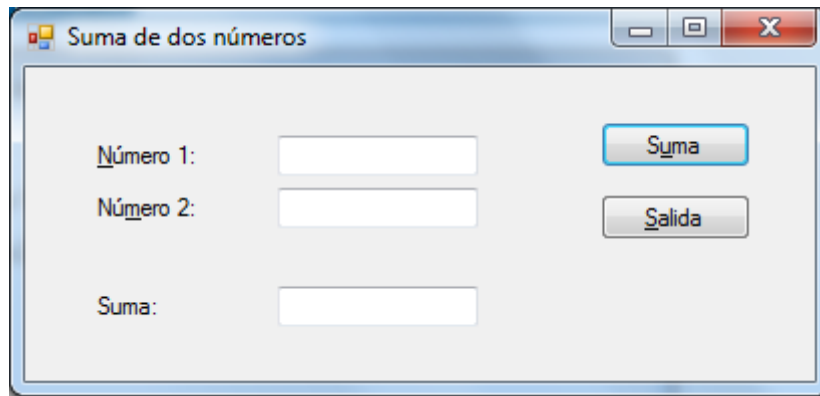
5.- Dentro de la caja que aparece ir a la sección de Especificaciones de guardado del archivo del programa e introducir lo siguiente:

Nombre: **Práctica 17**

Ubicación: (Ir a **Examinar** y elegir ubicación, usar unidad de almacenamiento móvil)

Nombre de la solución: (Automáticamente se copia el **Nombre**, dejarlo con el mismo)

6.- Diseñar la forma, que deberá quedar como se muestra en la siguiente figura:



a) Detalles de la forma.

7.- Dar click en la forma e ir a la **Ventana propiedades**, y localizar la propiedad **Name** y dar un click a su derecha donde dice **Form1** y sustituir por **frmPractica17** (no acepta espacios en blanco) para así asignarle nombre a la forma.

8.- Estando la forma seleccionada ir a la **Ventana propiedades**, y localizar la propiedad **Text** y dar un click a su derecha donde dice **Form1** y sustituir por **Suma de dos números** para así agregarle el título que aparecerá en la forma.

9.- Ir la propiedad **BackColor** y elegir un color para la forma.

10.- El tamaño de la forma se puede cambiar como cualquier ventana de Windows con arrastrar cualquier parte de su contorno.

b) Detalles de las etiquetas.

11.- Crear la etiqueta utilizando la herramienta **Label**  y dar un click donde se desea insertar la etiqueta.

12.- Estando seleccionada la etiqueta ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Label1** y sustituir por **lblNumero1** (no acepta espacios en blanco) para así asignarle nombre a la etiqueta.

13.- Estando la etiqueta seleccionada ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Label1** y sustituir por **&Número 1:** para así agregarle el texto que aparecerá en la etiqueta (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución).

14.- Ir la propiedad **ForeColor** y elegir un color para la etiqueta.

15.- Ir la propiedad **Font** y elegir un Tipo de letra para la etiqueta.

c) Detalles de las cajas de texto.

16.- Para insertar una caja de texto dar click en la **Cuadro de herramientas** en  **Text Box** y dar click en la forma donde se desea insertar la caja de texto.

17.- Estando seleccionada la caja de texto ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **textBox1** y sustituir por **txtNumero1** (no acepta espacios en blanco) para así asignarle nombre a la caja de texto.

18.- Ir la propiedad **BackColor** y elegir un color para la caja de texto.

19.- Ir la propiedad **ForeColor** y elegir un color para el texto que posteriormente aparecerá en la caja de texto.

20.- Ir la propiedad **Font** y elegir un Tipo de letra para el texto que posteriormente aparecerá en la caja de texto.

21.- Repetir los pasos del 11 al 20 para las siguientes etiquetas y caja de texto (crear primero la etiqueta y después su correspondiente caja de texto):

Propiedades		
<i>Name</i>	<i>Text</i>	<i>ReadOnly</i>
lblNumero2	Nú&mero 2:	
lblSuma	La suma es:	
txtNumero1		False
txtSuma		True

d) Detalles de los botones de comando.

22.-Para insertar un botón de comando dar un click en la **Cuadro de herramientas** en  **Button** y dar click en la forma donde se desea insertar el botón de comando.

23.- Estando seleccionado el botón de comando ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Button1** y sustituir por **cmdSuma** (no acepta espacios en blanco) para así asignarle nombre.

24.- Ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Button1** y sustituir por **S&uma** para así agregarle el texto que aparecerá en el botón (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución). Puedes cambiar tipo de letra color, etc.

25.- Agregar botón de comando repitiendo los pasos 22 al 24 para la salida del programa dejando que también se active con la tecla **Esc**, así como establecer el botón que se ejecutará al presionar

la tecla **Enter**, para lo que se modificarán dos propiedades de la forma **frmPractica17**. Utilizar las siguientes propiedades:

Propiedades			
<i>Name</i>	<i>Text</i>	<i>AcceptButton</i>	<i>CancelButton</i>
cmdSalida	&Salida	-	-
FrmPractica17	-	cmdSuma	cmdSalida

e) Introducir código.

26.- Dar doble click sobre el botón **cmdSuma** y escribir el código correspondiente quedando finalmente como a continuación:

```
// Creado automáticamente para el botón cmdSuma
private void cmdSuma_Click(object sender, EventArgs e)
{
    // Programa que suma dos número utilizando una función que no retorna
    // valor
    // Práctica 17

    // Declaración de variables
    int N1, N2;

    // Asignación de valores a las variables
    N1 = System.Int32.Parse(txtNumero1.Text);
    N2 = System.Int32.Parse(txtNumero2.Text);

    // Llamada a la función que no retorna valor (procedimiento) Suma()
    Suma(N1,N2);
}
```

27.- Posteriormente escribir fuera del subprograma o función que no retorna valor (procedimiento) para el botón cmdSuma el código del procedimiento definido por el programador que hará la suma.

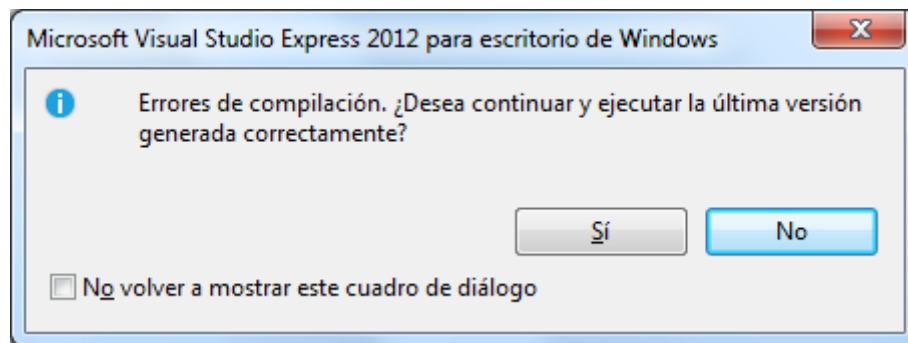
```
//Subprograma de la función que no retorna valor (procedimiento) definido
//por el Programador
void Suma(int Num1, int Num2)
{
    //Declaración de variable local
    int Sum;
```

```
//Suma de los dos números  
Sum = Num1 + Num2;  
  
//Despliegue del resultado de la suma  
txtSuma.Text = Sum.ToString();  
}
```

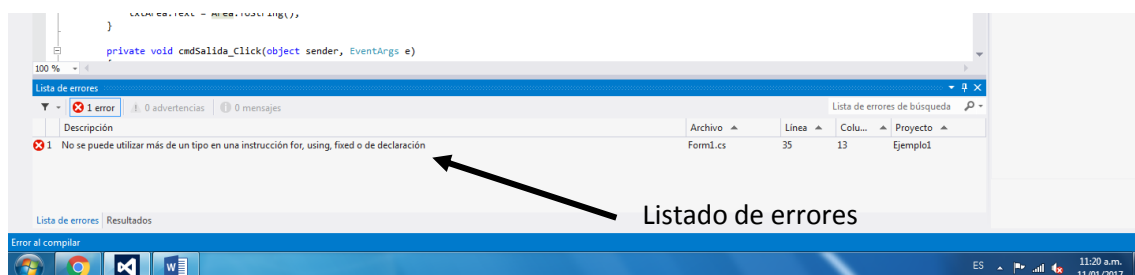
28.- Guardar el programa **Práctica 17**. En caso de que no hayas especificado desde el inicio el nombre y ubicación del programa ir a la opción del menú **ARCHIVO** y posteriormente ir a **Guardar WindowsFormsApplication1 como**. Asegurarse que en el **Explorador de soluciones** se encuentre elegido **WindowsFormsApplication1** para guardar correctamente.

29.- Para compilar y ejecutar el programa dar click en , presionar F5 o ir a la opción del menú **DEPURAR** y elegir **Iniciar depuración**.

30.- En caso que no se ejecute quiere decir que se presentaron errores durante la compilación o traducción y mostrará la siguiente caja:



Dar click en **No** para que permita revisar e identificar los errores de sintaxis y así poder corregirlos. Mostrará una división en la ventana donde se enlistan los errores (los errores varían):



31.- Probar el programa y realizar modificaciones si es necesario.

Notas práctica 17:

- Dentro de la práctica 17 se vieron las clases Form, Label, Text, Button y las instancias de dichas clases a través de los objetos frmPractica17, lblNumero1, lblNumero2, lblSuma, txtNumero1, txtNumero2, txtSuma, cmdSuma, cmdSalida, respectivamente, que como ya se dijo, un objeto es una unidad dentro de un programa que consta de un estado (propiedades) y de un comportamiento (métodos), que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.
- Se modificaron las propiedades de Name para los objetos usando prefijos de acuerdo a la clase: Form (prefijo frm), Label (prefijo lbl), Text (prefijo txt), Button (prefijo cmd).
- Se ligaron métodos a los objetos de la clase Button (cmdSuma, cmdSalida).
- En este programa se introduce el concepto de módulo definido por el usuario y particularmente el módulo que no retorna valor.
- Para hacer la llamada solo se especifica el nombre del módulo que no retorna valor y los parámetros entre paréntesis si es que existen.

Suma(N1,N2);

- Para definir el inicio del módulo que no retorna valor se debe indicar **void** ya que no devolverá valor al momento de retornar. Posteriormente se declaran los parámetros, como variables normales, entre paréntesis si es que existen.

```
void Suma(int Num1, int Num2)
{
}
}
```

PRÁCTICA 18. SUMA DE DOS NÚMEROS POR MEDIO DEL USO DE UNA FUNCIÓN Y PASE DE PARÁMETROS.

OBJETIVO ESPECÍFICO.

Conocer y utilizar los métodos que retornan valor para la solución de problemas del contexto utilizando modularidad y un lenguaje orientado a objetos..

TEMAS CON LOS QUE SE RELACIONA.

6.1 Declaración de métodos.

6.2 Métodos de clase.

6.3 Métodos de instancia.

DESARROLLO DE LA PRÁCTICA.

1.- Iniciar Visual C#

2.- Elegir **Aplicación de Windows Forms**

3.- Elegir **Nuevo Proyecto**

4.- Elegir **Aplicación de Windows Forms**

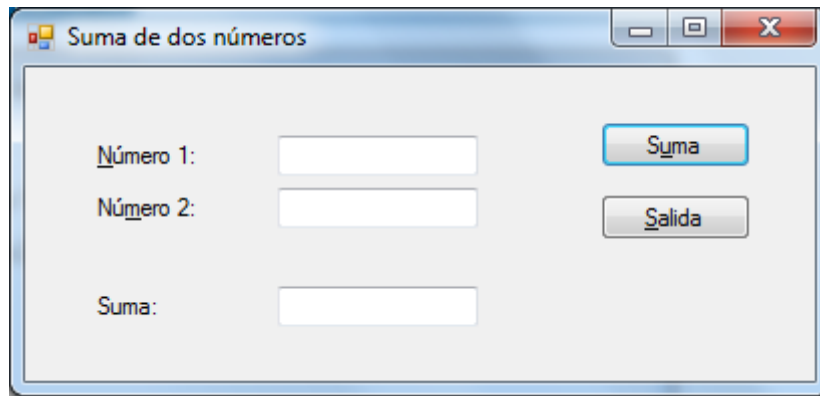
5.- Dentro de la caja que aparece ir a la sección de Especificaciones de guardado del archivo del programa e introducir lo siguiente:

Nombre: **Práctica 18**

Ubicación: (Ir a **Examinar** y elegir ubicación, usar unidad de almacenamiento móvil)

Nombre de la solución: (Automáticamente se copia el **Nombre**, dejarlo con el mismo)

6.- Diseñar la forma, que deberá quedar como se muestra en la siguiente figura:



a) Detalles de la forma.

7.- Dar click en la forma e ir a la **Ventana propiedades**, y localizar la propiedad **Name** y dar un click a su derecha donde dice **Form1** y sustituir por **frmPractica18** (no acepta espacios en blanco) para así asignarle nombre a la forma.

8.- Estando la forma seleccionada ir a la **Ventana propiedades**, y localizar la propiedad **Text** y dar un click a su derecha donde dice **Form1** y sustituir por **Suma de dos números** para así agregarle el título que aparecerá en la forma.

9.- Ir la propiedad **BackColor** y elegir un color para la forma.

10.- El tamaño de la forma se puede cambiar como cualquier ventana de Windows con arrastrar cualquier parte de su contorno.

b) Detalles de las etiquetas.

11.- Crear la etiqueta utilizando la herramienta **Label**  y dar un click donde se desea insertar la etiqueta.

12.- Estando seleccionada la etiqueta ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Label1** y sustituir por **lblNumero1** (no acepta espacios en blanco) para así asignarle nombre a la etiqueta.

13.- Estando la etiqueta seleccionada ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Label1** y sustituir por **&Número 1:** para así agregarle el texto que aparecerá en la etiqueta (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución).

14.- Ir la propiedad **ForeColor** y elegir un color para la etiqueta.

15.- Ir la propiedad **Font** y elegir un Tipo de letra para la etiqueta.

c) Detalles de las cajas de texto.

16.- Para insertar una caja de texto dar click en la **Cuadro de herramientas** en  **Text Box** y dar click en la forma donde se desea insertar la caja de texto.

17.- Estando seleccionada la caja de texto ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **textBox1** y sustituir por **txtNumero1** (no acepta espacios en blanco) para así asignarle nombre a la caja de texto.

18.- Ir la propiedad **BackColor** y elegir un color para la caja de texto.

19.- Ir la propiedad **ForeColor** y elegir un color para el texto que posteriormente aparecerá en la caja de texto.

20.- Ir la propiedad **Font** y elegir un Tipo de letra para el texto que posteriormente aparecerá en la caja de texto.

21.- Repetir los pasos del 11 al 20 para las siguientes etiquetas y caja de texto (crear primero la etiqueta y después su correspondiente caja de texto):

Propiedades		
<i>Name</i>	<i>Text</i>	<i>ReadOnly</i>
lblNumero2	Nú&mero 2:	
lblSuma	La suma es:	
txtNumero1		False
txtSuma		True

d) Detalles de los botones de comando.

22.-Para insertar un botón de comando dar un click en la **Cuadro de herramientas** en  **Button** y dar click en la forma donde se desea insertar el botón de comando.

23.- Estando seleccionado el botón de comando ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Button1** y sustituir por **cmdSuma** (no acepta espacios en blanco) para así asignarle nombre.

24.- Ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Button1** y sustituir por **S&uma** para así agregarle el texto que aparecerá en el botón (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución). Puedes cambiar tipo de letra color, etc.

25.- Agregar botón de comando repitiendo los pasos 22 al 24 para la salida del programa dejando que también se active con la tecla **Esc**, así como establecer el botón que se ejecutará al presionar

la tecla **Enter**, para lo que se modificarán dos propiedades de la forma **frmPractica18**. Utilizar las siguientes propiedades:

Propiedades			
<i>Name</i>	<i>Text</i>	<i>AcceptButton</i>	<i>CancelButton</i>
cmdSalida	&Salida	-	-
FrmPractica18	-	cmdSuma	cmdSalida

e) Introducir código.

26.- Dar doble click sobre el botón **cmdSuma** y escribir el código correspondiente quedando finalmente como a continuación:

```
// Creado automáticamente para el botón cmdSuma
private void cmdSuma_Click(object sender, EventArgs e)
{
    // Programa que suma dos número utilizando una función
    // Práctica 18

    // Declaración de variables
    int N1, N2, Res;

    // Asignación de valores a las variables
    N1 = System.Int32.Parse(txtNumero1.Text);
    N2 = System.Int32.Parse(txtNumero2.Text);

    // Llamada a la función Suma ()
    Res = Suma(N1, N2);

    //Despliegue del resultado de la suma
    txtSuma.Text = Res.ToString();
}
```

27.- Posteriormente escribir fuera del subprograma para el botón cmdSuma el código de la función definida por el programador que hará la suma.

```
//Subprograma (Función)definido por el Programador
int Suma(int Num1, int Num2)
{
    //Calculo de la suma y retorno del total
    return Num1 + Num2;
}
```

28.- En la caja **Solution Explorer** dar un click para regresar al diseño de la forma.

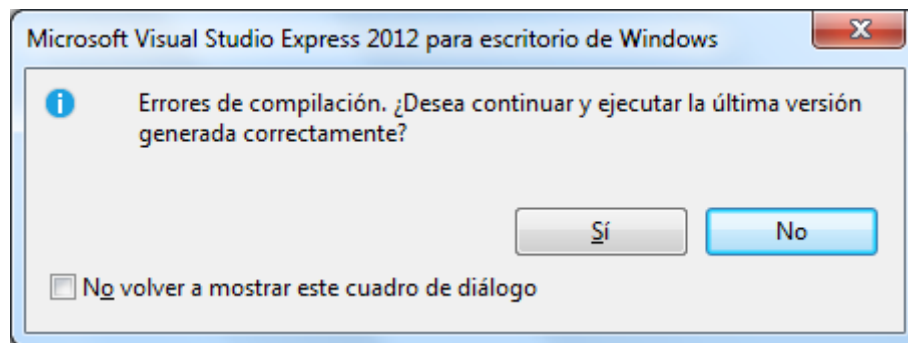
29.- Dar doble click al botón cmdSalida y sin mover el cursor introducir el código correspondiente quedando así finalmente:

```
//Creado automáticamente para el botón cmdSalida
private void cmdSalida_Click(object sender, EventArgs e)
{
    Close();
}
```

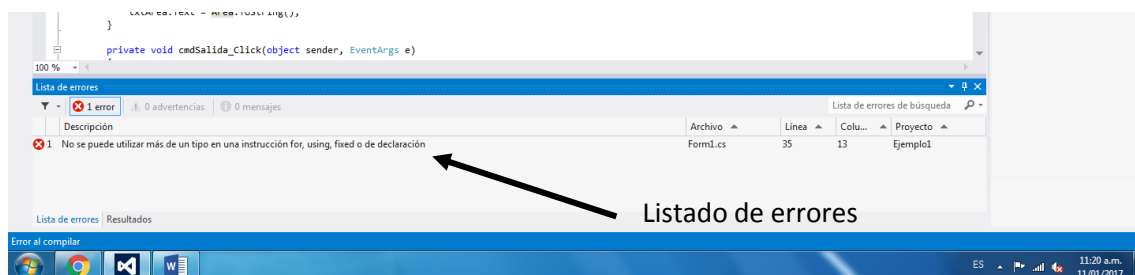
30.- Guardar el programa **Práctica 18**. En caso de que no hayas especificado desde el inicio el nombre y ubicación del programa ir a la opción del menú **ARCHIVO** y posteriormente ir a **Guardar WindowsFormsApplication1 como**. Asegurarse que en el **Explorador de soluciones** se encuentre elegido **WindowsFormsApplication1** para guardar correctamente.

31.- Para compilar y ejecutar el programa dar click en , presionar F5 o ir a la opción del menú **DEPURAR** y elegir **Iniciar depuración**.

32.- En caso que no se ejecute quiere decir que se presentaron errores durante la compilación o traducción y mostrará la siguiente caja:



Dar click en **No** para que permita revisar e identificar los errores de sintaxis y así poder corregirlos. Mostrará una división en la ventana donde se enlistan los errores (los errores varían):



33.- Probar el programa y realizar modificaciones si es necesario.

Notas práctica 18:

- Dentro de la práctica 18 se vieron las clases Form, Label, Text, Button y las instancias de dichas clases a través de los objetos frmPractica18, lblNumero1, lblNumero2, lblSuma, txtNumero1, txtNumero2, txtSuma, cmdSuma, cmdSalida, respectivamente, que como ya se dijo, un objeto es una unidad dentro de un programa que consta de un estado (propiedades) y de un comportamiento (métodos), que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.
- Se modificaron las propiedades de Name para los objetos usando prefijos de acuerdo a la clase: Form (prefijo frm), Label (prefijo lbl), Text (prefijo txt), Button (prefijo cmd).
- Se ligaron métodos a los objetos de la clase Button (cmdSuma, cmdSalida).
- En este programa se introduce el concepto de módulo definido por el usuario y particularmente es aquel módulo que si permite retornar un valor explícito.
- Para hacer la llamada solo se especifica el nombre del módulo y los parámetros entre paréntesis si es que existen y también la variable que recibirá el valor con el que retornará.

`Res = Suma(N1,N2);`

- Para definir el inicio del módulo se debe indicar **int** ya que el valor que devolverá será un entero. Posteriormente se declaran los parámetros, como variables normales, entre paréntesis si es que existen.

```
int Suma(int Num1, int Num2)
{
}
}
```

- Para indicar el valor que retornará se utiliza la palabra reservada **return** y enseguida el valor de retorno.

`return Num1+Num2;`

PRÁCTICA 19. SUMA DE DOS NÚMEROS UTILIZANDO FUNCIONES Y VARIABLES GLOBALES.

OBJETIVO ESPECÍFICO.

Conocer y utilizar las funciones que retornan valor y variables globales para la solución de problemas del contexto utilizando el modelo de las 6'D para definir las etapas para solución de problemas, algoritmos y un lenguaje orientado a objetos y un lenguaje orientado a objetos.

TEMAS CON LOS QUE SE RELACIONA.

6.1 Declaración de métodos.

6.2 Métodos de clase.

6.3 Métodos de instancia.

DESARROLLO DE LA PRÁCTICA.

1.- Iniciar Visual C#

2.- Elegir **Aplicación de Windows Forms**

3.- Elegir **Nuevo Proyecto**

4.- Elegir **Aplicación de Windows Forms**

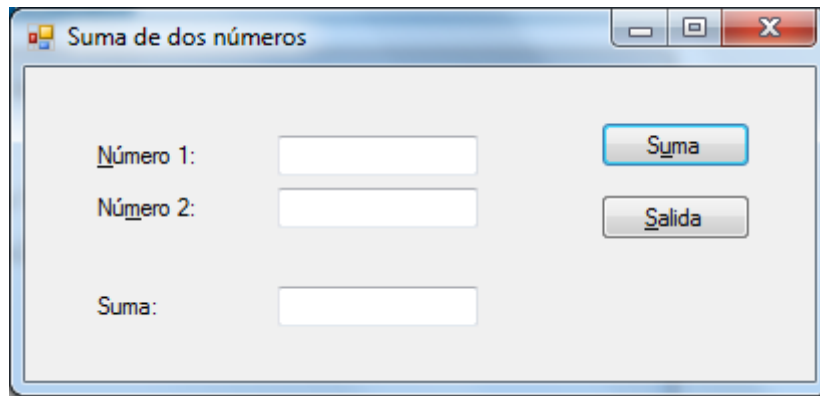
5.- Dentro de la caja que aparece ir a la sección de Especificaciones de guardado del archivo del programa e introducir lo siguiente:

Nombre: **Práctica 19**

Ubicación: (Ir a **Examinar** y elegir ubicación, usar unidad de almacenamiento móvil)

Nombre de la solución: (Automáticamente se copia el **Nombre**, dejarlo con el mismo)

6.- Diseñar la forma, que deberá quedar como se muestra en la siguiente figura:



a) Detalles de la forma.

7.- Dar click en la forma e ir a la **Ventana propiedades**, y localizar la propiedad **Name** y dar un click a su derecha donde dice **Form1** y sustituir por **frmPractica19** (no acepta espacios en blanco) para así asignarle nombre a la forma.

8.- Estando la forma seleccionada ir a la **Ventana propiedades**, y localizar la propiedad **Text** y dar un click a su derecha donde dice **Form1** y sustituir por **Suma de dos números** para así agregarle el título que aparecerá en la forma.

9.- Ir la propiedad **BackColor** y elegir un color para la forma.

10.- El tamaño de la forma se puede cambiar como cualquier ventana de Windows con arrastrar cualquier parte de su contorno.

b) Detalles de las etiquetas.

11.- Crear la etiqueta utilizando la herramienta **Label**  y dar un click donde se desea insertar la etiqueta.

12.- Estando seleccionada la etiqueta ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Label1** y sustituir por **lblNumero1** (no acepta espacios en blanco) para así asignarle nombre a la etiqueta.

13.- Estando la etiqueta seleccionada ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Label1** y sustituir por **&Número 1:** para así agregarle el texto que aparecerá en la etiqueta (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución).

14.- Ir la propiedad **ForeColor** y elegir un color para la etiqueta.

15.- Ir la propiedad **Font** y elegir un Tipo de letra para la etiqueta.

c) Detalles de las cajas de texto.

16.- Para insertar una caja de texto dar click en la **Cuadro de herramientas** en  **Text Box** y dar click en la forma donde se desea insertar la caja de texto.

17.- Estando seleccionada la caja de texto ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **textBox1** y sustituir por **txtNumero1** (no acepta espacios en blanco) para así asignarle nombre a la caja de texto.

18.- Ir la propiedad **BackColor** y elegir un color para la caja de texto.

19.- Ir la propiedad **ForeColor** y elegir un color para el texto que posteriormente aparecerá en la caja de texto.

20.- Ir la propiedad **Font** y elegir un Tipo de letra para el texto que posteriormente aparecerá en la caja de texto.

21.- Repetir los pasos del 11 al 20 para las siguientes etiquetas y caja de texto (crear primero la etiqueta y después su correspondiente caja de texto):

Propiedades		
<i>Name</i>	<i>Text</i>	<i>ReadOnly</i>
lblNumero2	Nú&mero 2:	
lblSuma	La suma es:	
txtNumero1		False
txtSuma		True

d) Detalles de los botones de comando.

22.-Para insertar un botón de comando dar un click en la **Cuadro de herramientas** en  **Button** y dar click en la forma donde se desea insertar el botón de comando.

23.- Estando seleccionado el botón de comando ir a las propiedades y modificar **Name** dando un click a su derecha donde dice **Button1** y sustituir por **cmdSuma** (no acepta espacios en blanco) para así asignarle nombre.

24.- Ir a las propiedades y modificar **Text** dando un click a su derecha donde dice **Button1** y sustituir por **S&uma** para así agregarle el texto que aparecerá en el botón (El & se utiliza para que la letra que le sigue se subraye y puede ser activada con la combinación de Alt+letra al momento de la ejecución). Puedes cambiar tipo de letra color, etc.

25.- Agregar botón de comando repitiendo los pasos 22 al 24 para la salida del programa dejando que también se active con la tecla **Esc**, así como establecer el botón que se ejecutará al presionar

la tecla **Enter**, para lo que se modificarán dos propiedades de la forma **frmPractica19**. Utilizar las siguientes propiedades:

Propiedades			
<i>Name</i>	<i>Text</i>	<i>AcceptButton</i>	<i>CancelButton</i>
cmdSalida	&Salida	-	-
FrmPractica19	-	cmdSuma	cmdSalida

e) Introducir código.

26.- Dar doble click sobre el botón **cmdSuma** y escribir el código correspondiente quedando finalmente como a continuación:

```
// Declaración de variables globales
int N1, N2;

// Creado automáticamente para el botón cmdSuma
private void cmdSuma_Click(object sender, EventArgs e)
{
    // Programa que suma dos número utilizando una función y
    // variables globales
    // Práctica 19

    // Declaración de variable local
    int Res;

    // Asignación de valores a las variables
    N1 = System.Int32.Parse(txtNumero1.Text);
    N2 = System.Int32.Parse(txtNumero2.Text);

    // Llamada a la función Suma ()
    Res = Suma();

    //Despliegue del resultado de la suma
    txtSuma.Text = Res.ToString();
}
```

27.- Posteriormente escribir fuera del subprograma para el botón cmdSuma el código de la función definida por el programador que hará la suma.

```
//Subprograma de la función definida por el Programador
int Suma()
```

```
{  
    //Calculo de la suma y retorno del total  
    return N1 + N2;  
}
```

28.- En la caja **Solution Explorer** dar un click para regresar al diseño de la forma.

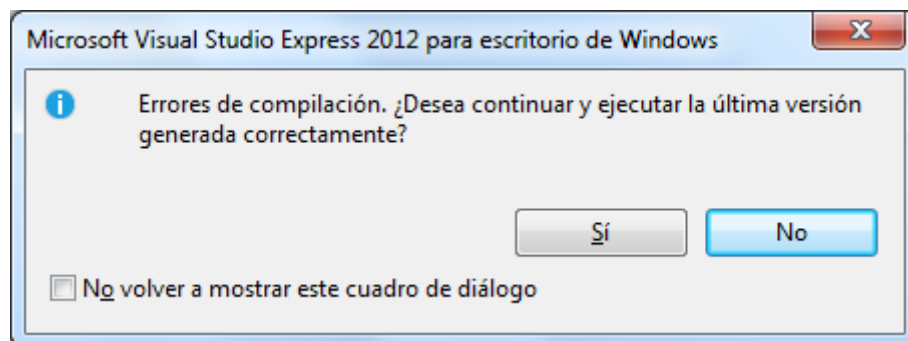
29.- Dar doble click al botón cmdSalida y sin mover el cursor introducir el código correspondiente quedando así finalmente:

```
//Creado automáticamente para el botón cmdSalida  
private void cmdSalida_Click(object sender, EventArgs e)  
{  
    Close();  
}
```

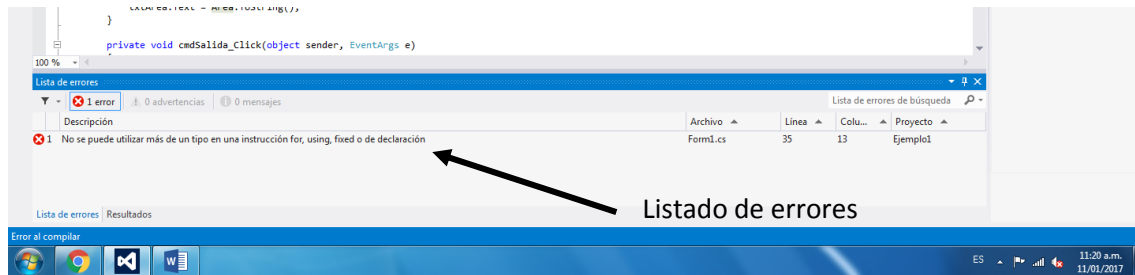
30.- Guardar el programa **Práctica 19**. En caso de que no hayas especificado desde el inicio el nombre y ubicación del programa ir a la opción del menú **ARCHIVO** y posteriormente ir a **Guardar WindowsFormsApplication1 como**. Asegurarse que en el **Explorador de soluciones** se encuentre elegido **WindowsFormsApplication1** para guardar correctamente.

31.- Para compilar y ejecutar el programa dar click en , presionar F5 o ir a la opción del menú **DEPURAR** y elegir **Iniciar depuración**.

32.- En caso que no se ejecute quiere decir que se presentaron errores durante la compilación o traducción y mostrará la siguiente caja:



Dar click en **No** para que permita revisar e identificar los errores de sintaxis y así poder corregirlos. Mostrará una división en la ventana donde se enlistan los errores (los errores varían):



33.- Probar el programa y realizar modificaciones si es necesario.

Notas práctica 19:

- Dentro de la práctica 18 se vieron las clases Form, Label, Text, Button y las instancias de dichas clases a través de los objetos frmPractica18, lblNumero1, lblNumero2, lblSuma, txtNumero1, txtNumero2, txtSuma, cmdSuma, cmdSalida, respectivamente, que como ya se dijo, un objeto es una unidad dentro de un programa que consta de un estado (propiedades) y de un comportamiento (métodos), que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.
- Se modificaron las propiedades de Name para los objetos usando prefijos de acuerdo a la clase: Form (prefijo frm), Label (prefijo lbl), Text (prefijo txt), Button (prefijo cmd).
- Se ligaron métodos a los objetos de la clase Button (cmdSuma, cmdSalida).
- En este programa se introduce el concepto de módulo definido por el usuario y particularmente es aquel módulo que si permite retornar un valor explícito.
- Para indicar el valor que retornará se utiliza la palabra reservada **return** y enseguida el valor de retorno.

```
return Num1+Num2;
```

- Aquí no hay necesidad de pase de parámetros ya que las variables N1 y N2 se declararon globales.

```
// Declaración de variables globales
int N1, N2;

// Creado automáticamente para el botón cmdSuma
private void cmdSuma_Click(object sender, EventArgs e)
{
    .
    .
    .
}
```

IV. CONCLUSIONES

Con la elaboración de este manual de prácticas se busca desarrollar en los estudiantes de la Ingeniería Informática, habilidades, metodologías, así como capacidades de análisis y síntesis, para plantear la solución de problemas susceptibles de ser computarizados, a través de diagramas de flujo, pseudocódigo, algoritmos y el paradigma de programación orientada a objetos así como las habilidades necesarias en la solución de problemas de base tecnológica en el área de Tecnologías de Información, a través de éstas prácticas se impulsa el logro de competencias específicas y genéricas dirigidas a la aplicación de los conceptos básicos de la programación.

Este manual de prácticas sienta una base firme en la preparación del estudiante desde el primer semestre de la Ingeniería Informática en el área de desarrollo de software, garantizando la adquisición gradual y sólida de habilidades y competencias específicas que podrán cubrir, en el futuro, las necesidades de la industria de Tecnologías de Información de la región que requiere de profesionales que tengan los conocimientos para desarrollar aplicaciones robustas que den soporte a la demanda constante de información confiable.

De igual manera este manual de prácticas estimula en el estudiante el desarrollo de competencias genéricas necesarias para un eficiente desarrollo profesional, tales como la capacidad de abstracción y análisis a través de los problemas del contexto a resolver, capacidad de aplicar los conocimientos en la práctica, habilidades en el uso de las tecnologías de la información y de la comunicación siendo su principal herramienta de trabajo, también la habilidad para buscar, procesar y analizar información procedente de fuentes diversas, la capacidad de trabajar en equipo requiriendo el desarrollo de habilidades interpersonales, la capacidad creativa siendo esencial en el desarrollo de software, habilidad para trabajar en forma autónoma durante el análisis de los problemas manteniendo siempre un compromiso ético y de calidad.

El Tecnológico Nacional de México marca el camino hacia el cambio constante en aras de formar profesionistas que puedan resolver problemas de base tecnológica que la sociedad presenta, por lo que este manual de prácticas contribuye al logro de este objetivo en el área de la Ingeniería Informática.