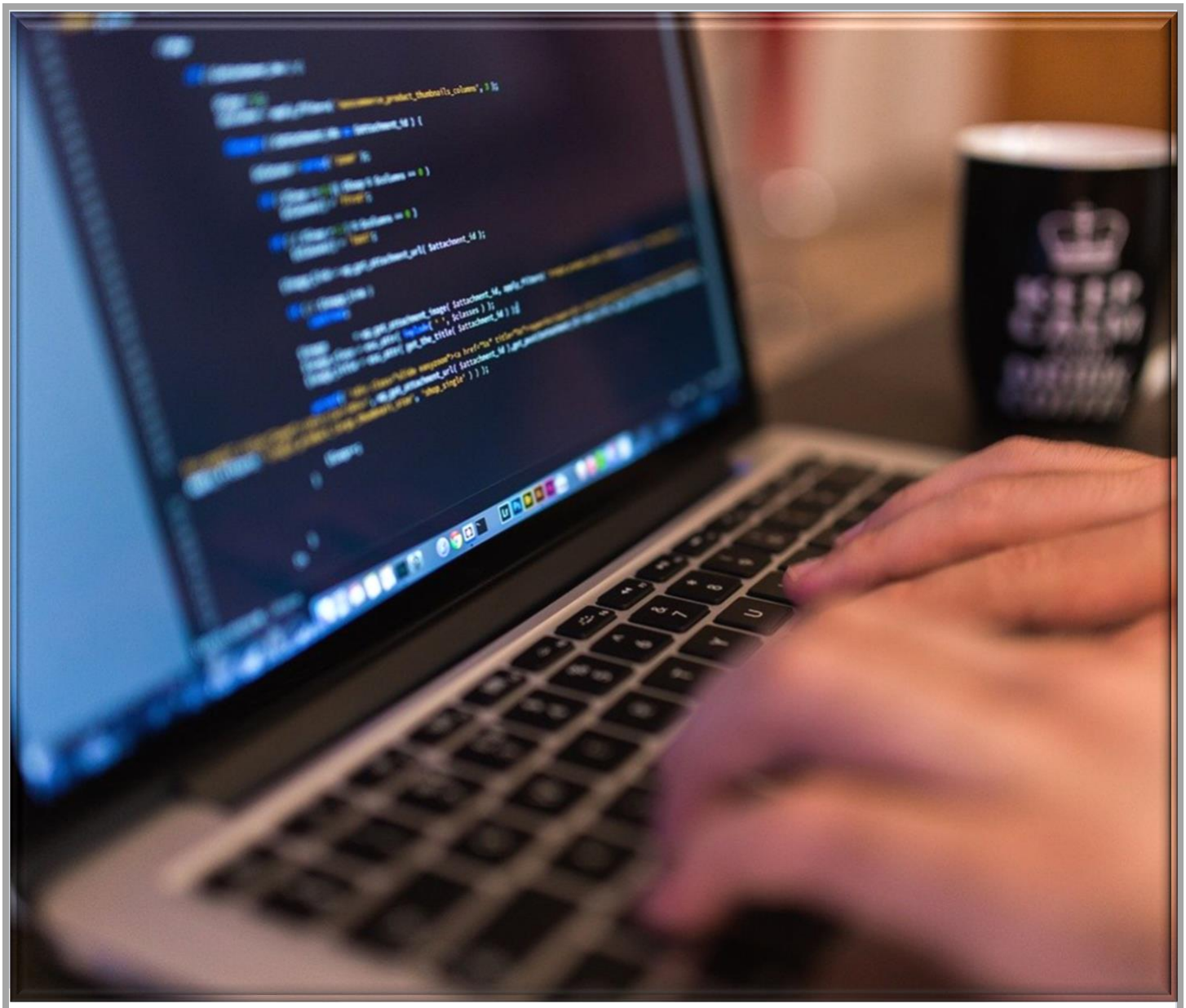


MÓDULO DIDÁCTICO PROGRAMACIÓN VISUAL BASIC

UNDÉCIMO GRADO

DEPARTAMENTO DE EDUCACIÓN
GOBIERNO DE PUERTO RICO

Página web: https://de.pr.gov/ **Twitter: @educacionpr**

CONTENIDO

LISTA DE COLABORADORES 1
CARTA PARA EL ESTUDIANTE, LAS FAMILIAS Y MAESTROS 2
CALENDARIO DE PROGRESO EN EL MÓDULO 4
UNIDAD 1: INTRODUCCIÓN A LA PROGRAMACIÓN *VISUAL BASIC* 2017 5
1.1 - ¿Qué es una Computadora? 5
1.2 - Operaciones Básicas del Programa 5
1.3 - *Visual Basic* 2017 y *Visual Studio* 2017 7
1.4 - *.NET Framework* 4.6.2 9
1.5 - Tipos de Aplicaciones de *Visual Basic* 11
Actividad de Avalúo 1: Unidad 1 12
UNIDAD 2: PROGRAMA Y DISEÑO DE LA INTERFAZ GRÁFICA DE USUARIO 16
2.1 - Uso de *Visual Studio* 2017 16
Actividad de Avalúo 2: Unidad 2 ➡ 2.1 20
2.2 - Agregar Objetos al GUI 22
2.3 - Ciclo de Desarrollo de una Aplicación 28
Actividad de Avalúo 3 : Unidad 2 ➡ 2.2, 2.3 30
UNIDAD 3: DISEÑO DE PROGRAMA Y CODIFICACIÓN 31
3.1 - Ajustar la interfaz gráfica de usuario 31
Actividad de Avalúo 4: Unidad 3 ➡ 3.1 38
3.2 - Códigos de Programación *Visual Basic* 39
Actividad de Avalúo 5: Unidad 3 ➡ 3.2 48
UNIDAD 4: VARIABLES Y OPERACIONES ARITMÉTICAS 51
4.1 - Insertar Variables y Operaciones en el Diseño de la Interfaz 51
4.2 - Introducción a la Entrada de Datos y Tipos de Datos 54
4.3 - Operaciones Aritméticas 63
4.4 - Borrar el Formulario 66
Actividad de Avalúo 6: Unidad 4 68
CLAVES DE RESPUESTA DE ACTIVIDADES DE AVALÚO 77
REFERENCIA 80
GUÍA DE ACOMODOS RAZONABLES PARA LOS ESTUDIANTES QUE TRABAJARÁN BAJO MÓDULOS DIDÁCTICOS 82

HOJA DE DOCUMENTAR LOS ACOMODOS RAZONABLES UTILIZADOS AL TRABAJAR EL MÓDULO DIDÁCTICO.....84

LISTA DE COLABORADORES

Prof. Jean C. Cintrón Colón
Maestro de Educación Comercial
Escuela Superior Dra. María Cadilla de Martínez
Arecibo, Puerto Rico

Prof. José R. Jiménez Hernández
Maestro de Educación Comercial
Escuela Superior Vocacional Nueva de Loíza
Loíza, Puerto Rico

Prof. Laura Álamo Serrano
Maestra de Educación Comercial
Escuela Ramón Power y Giralt
Las Piedras, Puerto Rico

Prof. Madeline Álvarez Ortiz
Maestra de Educación Comercial
Escuela Emilio R. Delgado
Corozal, Puerto Rico

Prof. Mercedes Alverio Rivera
Maestra de Educación Comercial
Escuela Rafael Cordero Molina
San Juan, Puerto Rico

Prof. Janette González Pérez
Maestra de Educación Comercial
Escuela Jaime A. Collazo del Río
Morovis, Puerto Rico

Prof. Mirelys Maceira Ruiz
Maestra de Educación Comercial
Escuela Fernando Callejo
Manatí, Puerto Rico

Prof. Luz Roldán Rohena
Maestra de Educación Comercial
Escuela Voc. Ana Delia Flores Santana
Fajardo, Puerto Rico

Prof. Yesenia Pérez Irizarry
Maestra de Educación Comercial
Escuela Dr. Manuel de la Pila Iglesias
Ponce, Puerto Rico

CARTA PARA EL ESTUDIANTE, LAS FAMILIAS Y MAESTROS

Estimado estudiante:

Este módulo didáctico es un documento que favorece tu proceso de aprendizaje. Además, permite que aprendas en forma más efectiva e independiente, es decir, sin la necesidad de que dependas de la clase presencial o a distancia en todo momento. Del mismo modo, contiene todos los elementos necesarios para el aprendizaje de los conceptos claves y las destrezas de la clase de Programación Visual Basic, sin el apoyo constante de tu maestro. Su contenido ha sido elaborado por maestros, facilitadores docentes y directores de los programas académicos del Departamento de Educación de Puerto Rico (DEPR) para apoyar tu desarrollo académico e integral en estos tiempos extraordinarios en que vivimos.

Te invito a que inicies y completes este módulo didáctico siguiendo el calendario de progreso establecido por semana. En él, podrás repasar conocimientos, refinar habilidades y aprender cosas nuevas sobre la clase de Programación Visual Basic por medio de definiciones, ejemplos, lecturas, ejercicios de práctica y de evaluación. Además, te sugiere recursos disponibles en la internet, para que amplíes tu aprendizaje. Recuerda que esta experiencia de aprendizaje es fundamental en tu desarrollo académico y personal, así que comienza ya.

Estimadas familias:

El Departamento de Educación de Puerto Rico (DEPR) comprometido con la educación de nuestros estudiantes, ha diseñado este módulo didáctico con la colaboración de: maestros, facilitadores docentes y directores de los programas académicos. Su propósito es proveer el contenido académico de la materia de Programación Visual Basic para las primeras diez semanas del nuevo año escolar. Además, para desarrollar, reforzar y evaluar el dominio de conceptos y destrezas claves. Ésta es una de las alternativas que promueve el DEPR para desarrollar los conocimientos de nuestros estudiantes, tus hijos, para así mejorar el aprovechamiento académico de estos.

Está probado que cuando las familias se involucran en la educación de sus hijos mejora los resultados de su aprendizaje. Por esto, te invitamos a que apoyes el desarrollo académico e integral de tus hijos utilizando este módulo para apoyar su aprendizaje. Es fundamental que tu hijo avance en este módulo siguiendo el calendario de progreso establecido por semana.

El personal del DEPR reconoce que estarán realmente ansiosos ante las nuevas modalidades de enseñanza y que desean que sus hijos lo hagan muy bien. Le solicitamos a las familias que brinden una colaboración directa y activa en el proceso de enseñanza y aprendizaje de sus hijos. En estos tiempos extraordinarios en que vivimos, les recordamos que es importante que desarrolles la confianza, el sentido de logro y la independencia de tu hijo al realizar las tareas escolares. No olvides que las necesidades educativas de nuestros niños y jóvenes es responsabilidad de todos.

Estimados maestros:

El Departamento de Educación de Puerto Rico (DEPR) comprometido con la educación de nuestros estudiantes, ha diseñado este módulo didáctico con la colaboración de: maestros, facilitadores docentes y directores de los programas académicos. Este constituye un recurso útil y necesario para promover un proceso de enseñanza y aprendizaje innovador que permita favorecer el desarrollo holístico e integral de nuestros estudiantes al máximo de sus capacidades. Además, es una de las alternativas que se proveen para desarrollar los conocimientos claves en los estudiantes del DEPR; ante las situaciones de emergencia por fuerza mayor que enfrenta nuestro país.

El propósito del módulo es proveer el contenido de la materia de Programación Visual Basic para las primeras diez semanas del nuevo año escolar. Es una herramienta de trabajo que les ayudará a desarrollar conceptos y destrezas en los estudiantes para mejorar su aprovechamiento académico. Al seleccionar esta alternativa de enseñanza, deberás velar que los estudiantes avancen en el módulo siguiendo el calendario de progreso establecido por semana. Es importante promover el desarrollo pleno de estos, proveyéndole herramientas que puedan apoyar su aprendizaje. Por lo que, deben diversificar los ofrecimientos con alternativas creativas de aprendizaje y evaluación de tu propia creación para reducir de manera significativa las brechas en el aprovechamiento académico.

El personal del DEPR espera que este módulo les pueda ayudar a lograr que los estudiantes progresen significativamente en su aprovechamiento académico. Esperamos que esta iniciativa les pueda ayudar a desarrollar al máximo las capacidades de nuestros estudiantes.

CALENDARIO DE PROGRESO EN EL MÓDULO

DÍAS / SEMANAS	LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES
1	Unidad 1	Unidad 1	Unidad 1	Unidad 1	Unidad 1
2	Unidad 1	Unidad 1	Unidad 1	Unidad 1	Unidad 1
3	Unidad 1	Unidad 1	Unidad 1	Unidad 1	Unidad 1
4	Unidad 2	Unidad 2	Unidad 2	Unidad 2	Unidad 2
5	Unidad 2	Unidad 2	Unidad 2	Unidad 2	Unidad 2
6	Unidad 3	Unidad 3	Unidad 3	Unidad 3	Unidad 3
7	Unidad 3	Unidad 3	Unidad 3	Unidad 3	Unidad 3
8	Unidad 4	Unidad 4	Unidad 4	Unidad 4	Unidad 4
9	Unidad 4	Unidad 4	Unidad 4	Unidad 4	Unidad 4
10	Unidad 4	Unidad 4	Unidad 4	Unidad 4	Unidad 4

UNIDAD 1: INTRODUCCIÓN A LA PROGRAMACIÓN *VISUAL BASIC* 2017

Estándar: Programación y desarrollo de aplicaciones

Objetivos:
1. Define terminología de programación orientada a objetos.
2. Identifica estructuras de programación.
3. Elige el lenguaje de programación apropiado para el desarrollo de aplicaciones en tareas específicas.

1.1 - ¿Qué es una Computadora?

La computadora es un dispositivo electrónico que completa tareas bajo la dirección de una secuencia de instrucciones para producir resultados útiles para las personas. Tienen tres componentes vitales los cuales deben interacturar uno con el otro para que se pueda ejecutar cualquier tarea. Estos componentes son: *hardware*, programas o *software* y la data.

- *Hardware* - Son los equipos físicos asociados con una computadora. Estos pueden incluir: teclado, *mouse*, monitor, *CPU*, memorias *RAM* y disco duro.
- Programas de Computradoras o *Software* - Son una serie de instrucciones que dirige a los *hardware* para que puedan realizar las tareas. Un programa de computadora en dispositivos móviles se le conoce como *App*. Los programas de computadoras son diseñados por personas conocidos como **programadores**, o **desarrolladores**.
- Data - Ésta incluye palabras, números, videos, gráficos y audio, los cuales son manipulados, mostrados y procesados por los programas.

Las funciones básicas de cualquier programa de computadora son:

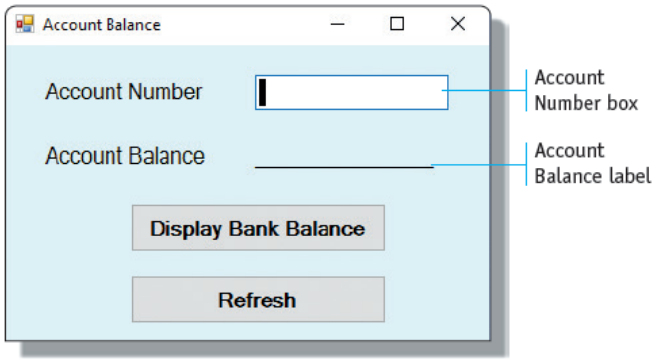
- Aceptar Data (*Input Data*)
- Procesar la Data
- Crear resultados útiles para las personas (*Output Data* o Información)

1.2 - Operaciones Básicas del Programa

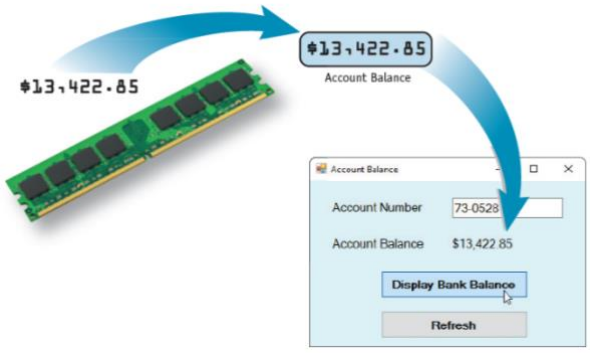
Todos los programas, sin importar su tamaño o complejidad, ejecutan solo unas pocas operaciones fundamentales. Estas operaciones son: Operaciones de Entrada, Operaciones de Salida, Operaciones Básicas de Aritmética y Operaciones de Lógica.

Operaciones de Entrada (*Input Operation*)

Esta actividad involucra el proceso de que un usuario de computadoras entre data a un programa. Un ejemplo de una operación de entrada lo sería colocar el número de cuenta en un programa bancario. Para lograr el proceso de entrada de data se pueden utilizar varíos dispositivos como lo son: teclado, escaner, pantalla táctil, cámara digital, video cámaras, *mouse*, micrófonos. La data entrada se almacenará en la memoria RAM.



Operaciones de Salida (*Output Operation*)



La principal meta de los programas es la creación de *output* o información de utilidad para las personas. Un ejemplo de una operación de salida es cuando obtenemos el balance de una cuenta luego de entrar el número de cuenta en el sistema. Algunos dispositivos utilizados en el proceso de *output* son: monitores, impresoras o bocinas.

Operaciones Básicas de Aritmética

Las operaciones básicas de aritmética incluyen la suma, resta, multipliación y división. Este tipo de operaciones es utilizada para data numérica que se utilizará en algún cálculo matemático.

Operaciones de Lógica

Las computadoras, por medio del uso de programas, pueden hacer comparaciones de números, letras del alfabeto y caracteres especiales. Basados en los resultados los programas pueden realizar una tarea, o proceso, si las condiciones probadas resultan ciertas, o por el contrario realizar otro proceso si las condiciones resultan

falsas. Utilizar los programas para comparar data y ejecutar operaciones alternativas, permite a las computadoras completar tareas sofisticadas tales como: predicciones del clima, edición de fotos digitales, editar videos, entre otros.

Los programas pueden realizar las siguientes operaciones de lógica:

- comparar para determinar si dos valores son iguales.
- comparar para determinar si un valor es mayor que otro.
- comparar para determinar si un valor es menor que otro.

Almacenamiento de Programas y Data

Cuando se desarrolla, o escribe, un programa los códigos escritos y otras funciones deben ser almacenadas en un disco o memoria. Al guardar un programa en un disco, este se puede ejecutar las veces que sea necesario sin la necesidad de volver a escribir los códigos.

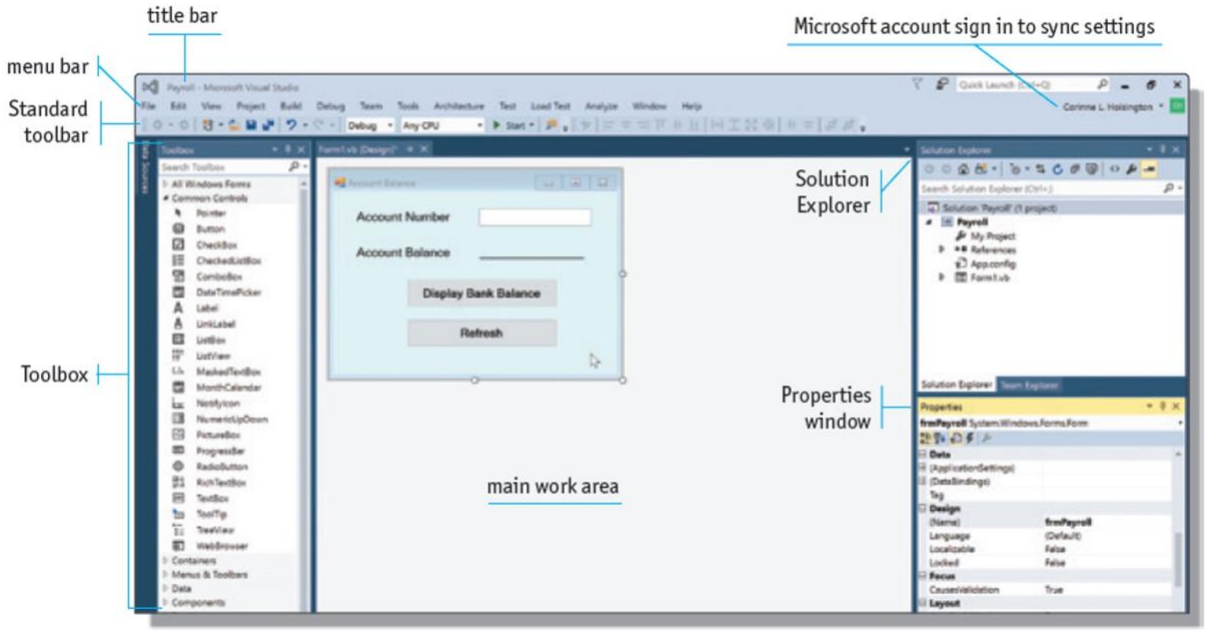
Los programas también pueden almacenar data, la cual se puede generar a partir del procesamiento del programa. En muchos casos se utilizan las bases de datos (*database*) como medio de almacenamiento de datos. Una **base de datos** es una colección de datos organizada de forma que permita el acceso, recuperación y uso de datos.

1.3 - *Visual Basic* 2017 y *Visual Studio* 2017

Cada declaración de programa hace que la computadora realice una o más operaciones. Para escribir programas computarizados, los desarrolladores hacen uso de lenguajes de programación. Un **lenguaje de programación** son una serie de palabras, símbolos y códigos escritos siguiendo un estricto conjunto de reglas de uso llamado **syntaxis**. El desarrollador debe seguir la sintaxis, o las reglas de programación, del lenguaje de programación con precisión. La mayoría de los desarrolladores utilizan una herramienta llamada *Visual Studio* 2017 para diseñar programas con lenguaje de programación *Visual Basic* 2017.

Visual Studio es un programa de aplicación que permite a los usuarios desarrollar programas utilizando diferentes lenguajes de programación. Este programa cuenta con un tipo de **entorno de desarrollo integrado** (**IDE** por sus siglas en inglés). *IDE* proporciona servicios y herramientas que permiten a un desarrollador codificar, probar e implementar un programa, o una serie de programas, que comprenden una aplicación.

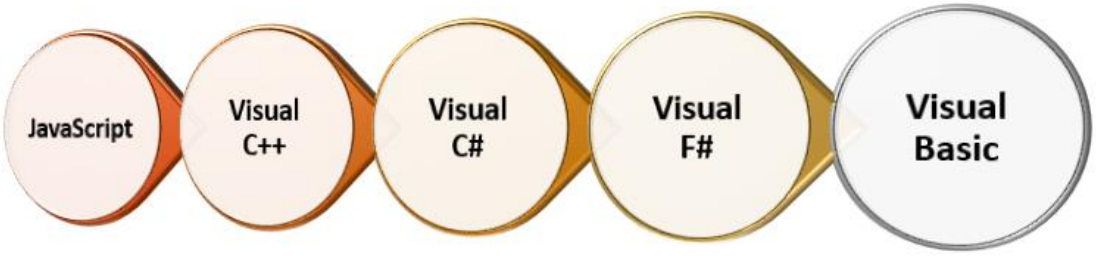
Pantalla Inicial del Programa Visual Studio



- ***Title Bar* (Barra de Título)**: identifica la ventana y la aplicación abierta en la ventana.
- ***Menu Bar* (Barra de Menú)**: muestra los nombres de menú de *Visual Studio*. Los menús contienen listas de comandos que le permiten crear, editar, guardar, imprimir, probar y ejecutar un programa de *Visual Basic*, y realizar otras funciones que son críticas para desarrollar programas de *Visual Basic*.
- ***Standard Toolbar* (Barra de Herramientas Estándar)**: contiene botones que ejecutan comandos de uso frecuente como *Open Project*, *New Project*, *Save*, *Cut*, *Copy*, *Paste* y *Undo*.
- ***Toolbox* (Caja de Herramientas)**: contiene componentes .NET que lo ayudan a desarrollar la GUI (Interfaz Gráfica de Usuario) para su programa. Por ejemplo, puede usarlo para colocar botones, cuadros de fotos, etiquetas y otros.
- ***Main work Area* (Área de trabajo principal)**: contiene el elemento que está desarrollando actualmente.

- ***Solution Explorer*** **(Explorador de Soluciones)**: muestra los elementos de solución de *Visual Basic*, lo cual es el nombre del programa de *Visual Basic* y otros elementos generados por *Visual Studio* para ayudar al programa a ejecutarse correctamente.
- ***Properties Windows*** **(Ventana de Propiedades)**: lugar donde se encuentran y se aplican las características que el objeto va a tener; como lo son el tamaño, color de texto, alineación, etc.

Visual Studio 2017 permite escribir programas en cinco lenguajes de programación:



Visual Basic

Visual Basic fue desarrollado por Microsoft a principios de los 90. Es un lenguaje que permite a los desarrolladores crear programas complejos de Windows y para la *Web*. La mayoría de los programas de *Visual Basic* son controlados por eventos (*event-driven programs*) que se comunica con el usuario a través de una **interfaz gráfica de usuario** (***graphical user interface, GUI***, por sus siglas e inglés). La GUI consiste en una ventana que contiene una variedad de objetos y que pueden mostrarse en varios dispositivos.

En la GUI se colocan objetos que componen la aplicación o programa a crear, Uno de los objetos pueden ser botones. Los botones, por ejemplo, tiene la intención de ser oprimidos por el usuario causando a su vez un evento. Un **evento** es una acción que hace que el programa realice un tipo de procesamiento.

1.4 - *.NET Framework* 4.6.2

Las tecnologías y productos *.NET* se diseñaron para trabajar juntos y permitir a las empresas conectar información, sistemas y dispositivos a travé del software. El ***.NET***

Framework proporciona herramientas y procesos que los desarrolladores pueden usar para producir y ejecutar programas. Su versión más reciente es el *.NET Framework* 4.6.2. Las principales características son: *.NET Class Library*, *ADO.NET*, *ASP.NET* y *Microsoft Intermediate Language and Comon Language Runtine* (CLR).

Una de las características más importantes del *.NET* es conocido como ***.NET Class Library***. Cuando los códigos son comunes en múltiples programas, una buena práctica es escribir los códigos una sola vez y reutilizarlos cuando sea necesario. Estos códigos son conocidos como clase (*class*). Una **clase** es un grupo nombrado de códigos de programas. Por otra parte existe la **Biblioteca de Clase** (***Class Library***) que es el lugar donde se almacenan las clases una vez creadas y las hacen disponibles a todos los desarrolladores que la necesiten usar. Un ejemplo sería añadir un botón al *GUI* desde códigos almacenados en la biblioteca de clases. Este botón sería lo que se conoce como un **objeto de una clase**, y el crearlo desde la Biblioteca de Clase se le conoce como **instanciación**. Este proceso de utilizar clases preconstruidas para desarrollar aplicaciones de una forma más rápida, fácil y confiable se conoce como **Desarrollo Rápido de Aplicaciones** (*Rapid Application Development*, *RAD* por sus siglas en inglés).

Entre las otras características del *.NET* se encuentra *ADO.NET* la cual provee la habilidad de leer y escribir data en una base de datos. El *ASP.NET* es una característica que provee la habilidad de desarrolla aplicaciones web para computaduras de escritorios, tabletas y navegadores móviles. El conocido *CLR* permite a los programas ejecutar en diferentes computadoras que operan bajo sistemas operativos diferentes. En otras palabras permite la compatibilidad entre sistemas, ejemplo, entre sistemas *Windows* y sistemas MAC.

1.5 - Tipos de Aplicaciones de *Visual Basic*

Cuando creamos programas en *Visual Basic*, utilizando *Visual Studio*, debemos elegir el tipo de aplicaciones que deseamos desarrollar. Los cinco tipos principales de aplicaciones son:

Aplicaciones	Características
Aplicaciones de Escritorio clásico de Windows (*Windows Classic Desktop Application*)	Son las aplicaciones de escritorio; el programa se ejecutará en una computadora u otro dispositivo que admita la *GUI* de *Windows*.
Aplicaciones Universales de Window (*Windows Universal apps*)	Diseñadas para ejecutarse en computadoras con *Windows* 8 o *Windows* 10 y dispositivos móviles.
Aplicaciones *Web* o *Cloud*	Utiliza *ASP.NET* y se ejecuta en un servidor *Web*.
Aplicaciones de Base de Datos	Está aplicación esta escrita usando *ADO.NET* para hacer referencia, acceder y actualizar datos almacenados en una base de datos.
Aplicaciones para *HoloLens*	*HoloLens* son un tipo de visores de realidad aumentada que coloca imágenes holográficas en 3D en su alcance de visión.

Actividad de Avalúo 1: Unidad 1

Realizar las contestaciones en la HOJA DE CONTESTACIONES (final del Módulo).

Ejercicio 1: Circule la(s) respuesta(s) correcta (s) para cada pregunta.

1. Una aplicación puede permitir que se ingresen datos utilizando un(a) _____.
   a. impresora
   b. monitor
   c. teclado
   d. bocina

2. ¿En qué orden la mayoría de los programas siguen estos pasos generales?
   a. procesar data, aceptar input, crear output
   b. crear output, aceptar input, procesar data
   c. aceptar input, procesar data, crear output
   d. aceptar input, crear output, procesar data

3. Los programas de Visual Basic 2017 son _____ que se comunican con el usuario a través de una interfaz gráfica de usuario (GUI).
   a. event-driven programs
   b. basados en GUI
   c. persistentes
   d. .NET Framework

4. Un _____ consiste en una ventana que contiene una variedad de objetos y que pueden mostrarse en varios dispositivos.
   a. ROM
   b. GUI
   c. CPU
   d. CLR

5. Tocar o hacer clic en un botón cuando se está ejecutando un programa de Visual Basic activa un____.
   a. índice
   b. GUI
   c. input
   d. evento

6. La suma y la resta se consideran operaciones ____ realizadas por una computadora.
   a. input
   b. comparación
   c. básicas de aritmética
   d. output

7. Una computadora usa operaciones _____ para comparar dos valores y determinar si son iguales entre sí.
   a. básicas de aritmética
   b. lógica
   c. agrupación
   d. clasificación

8. Un(a) _____ es una colección de datos organizados de una manera que permite el acceso, la recuperación y el uso de los datos.
   a. archivo
   b. folder
   c. programa
   d. base de datos

9. El(La) _____ de un lenguaje de programación es el conjunto de reglas de uso para el lenguaje.
   a. lógica
   b. semántica
   c. syntaxis
   d. GUI

10. Un(a) _____ es un conjunto de servicios y herramientas que permiten a un desarrollador codificar, probar e implementar un solo programa o una serie de programas que comprenden una aplicación.
    a. Entorno de Desarrollo Integrado (IDE)
    b. Interfáz Gráfica de Usuario (GUI)
    c. Ambiente de Desarrollo
    d. Microsoft Intermediate Language (MSIL)

11. Un botón o un cuadro de texto son ejemplos de objetos, también llamados _____, que son parte visible de la GUI.
    a. índice
    b. clases
    c. librería
    d. controles

12._____ es un lenguaje que puede ser utilizado en Visual Studio.
   a. Fortran
   b. Visual F#
   c. Lisp
   d. COBOL

13. Uno de los lenguajes de programación más utilizados en el mundo es ___.
   a. Pascal
   b. BASIC
   c. Fortran
   d. Visual Basic

14. El _____ proporciona herramientas y procesos que los desarrolladores pueden usar para producir y ejecutar programas.
   a. SQL Server Set
   b. .NET Framework
   c. Java Toolkit
   d. Oracle Academy

15. Todas las siguientes son características principales de .NET Framework 4.6.2, **EXCEPTO** ___.
   a. RAM
   b. ADO.NET
   c. ASP.NET
   d. .NET Class Library

16. Un conjunto de clases preescritas llamadas _____ le permite, por ejemplo, acceder a datos almacenados en una base de datos.
   a. ADO.NET
   b. ASP.NET
   c. RAD
   d. CLR

17. El proceso de blanco se utiliza para crear un objeto a partir de una clase.
   a. inicialización
   b. instanciación
   c. objetivación
   d. clasificación

18. El proceso de usar clases preconstruidas para hacer que el desarrollo de aplicaciones sea más rápido, más fácil y más confiable se llama _____.
    a. RAD
    b. CAD
    c. ARD
    d. CLR

19. Un _____ son personas expertas en el diseño y creación de programas de computadora utilizando lenguajes de programación.
    a. método
    b. evento
    c. desarrollador
    d. IDE

20. La mayoría de los desarrolladores usan una herramienta llamada _____ para escribir programas Visual Basic 2017.
    a. Visual Studio 2017
    b. RAD
    c. IDE
    d. computador

UNIDAD 2: PROGRAMA Y DISEÑO DE LA INTERFAZ GRÁFICA DE USUARIO

Estándar: Programa de Aplicación

Objetivos:

1. Evalúa las aplicaciones apropiadas para completar tareas productivamente.
2. Identifica recursos para resolver problemas utilizando el software de aplicaciones.
3. Compara y contrastar las características de la aplicación.
4. Analiza el beneficio de costos y el ciclo de vida de las aplicaciones.

2.1 - Uso de *Visual Studio* 2017

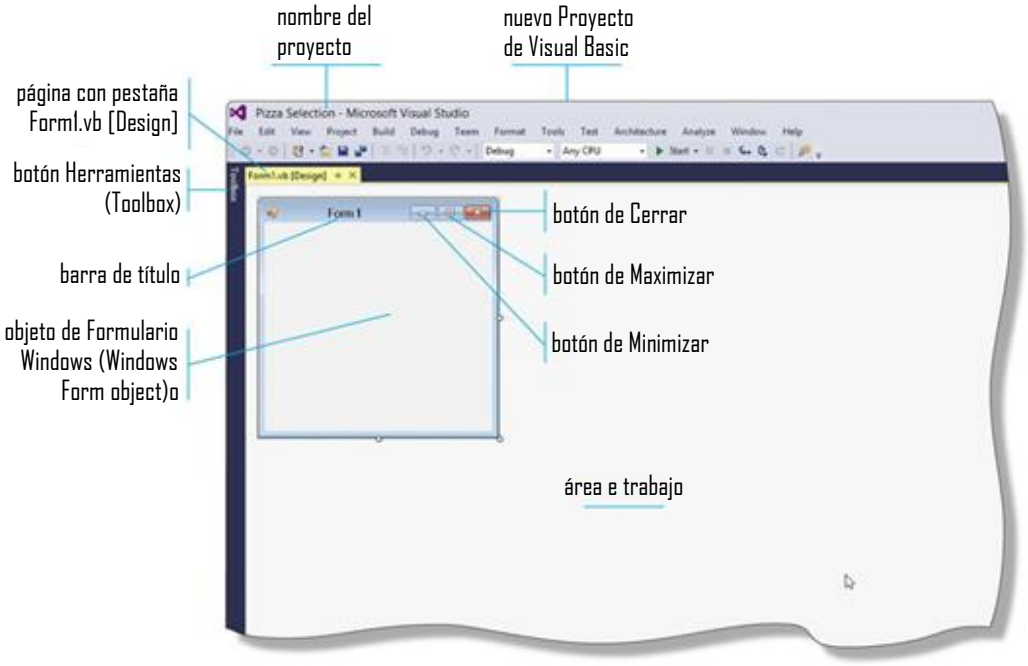
Cuando se diseña un programa controlado por eventos (*event-driven programs*), que utiliza una interfaz gráfica de usuario (*GUI*), uno de los primeros pasos después de definir el propósito del programa es diseñar la interfaz de usuario en sí. Sin embargo, antes de comenzar a diseñar la interfaz de usuario, el desarrollador debe saber cómo usar las herramientas de desarrollo rápido de aplicaciones (*RAD*) de *Visual Studio* y *Visual Basic* en el proceso de diseño.

Un **proyecto** es equivalente a un solo programa creado con *Visual Studio*. Un **proyecto de escritorio clásico de *Windows*** (*Windows Classic Desktop*) es un programa que incluye una interfaz de usuario cuyas ventanas se crean utilizando el sistema operativo *Windows*. Cuando se ejecuta el programa, el usuario interactúa con el programa utilizando sus ventanas y componentes de la interfaz de usuario.

Crear un nuevo proyecto Windows Classic Desktop de Visual Basic:

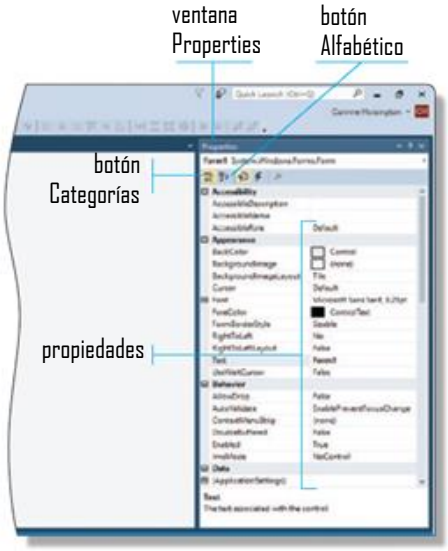
1. Acceda al programa.
2. En la página de inicio, haga clic en el botón *New Project* en la barra de herramientas estándar.
3. Haga clic en *Windows Classic Desktop* en el panel izquierdo, si no está seleccionado.
4. Haga clic en *Windows Forms App* en el panel central, si no está seleccionado.
5. Escriba el nombre del proyecto.
6. Haga clic en el botón Ok.

La ventana de *Visual Studio* contiene varias características importantes que debe conocer.



La página con la pestaña llamada *Form1.vb [Design]* contiene el objeto *Windows Form* llamado *Form1* (nombre por defecto, se puede cambiar el nombre). El objeto *Windows Form* es la ventana que usará para compilar el programa y luego mostrarlo en su pantalla cuando ejecute el programa.

Las propiedades de un objeto



Cada objeto creado en la interfaz de usuario, incluido el objeto *Windows Form*, tiene propiedades. Las **propiedades** pueden describir una multitud de características sobre el objeto, incluido su color, tamaño, nombre y posición en la pantalla, entre otros. Para ver las propiedades de un objeto, se utiliza la ventana *Properties*. Por defecto, la ventana *Properties* se muestra al lado derecho de la ventana de *Visual Studio* y la ventana de *Solution Explorer* se visualiza sobre la ventana *Properties*.

Nombrar el proyecto

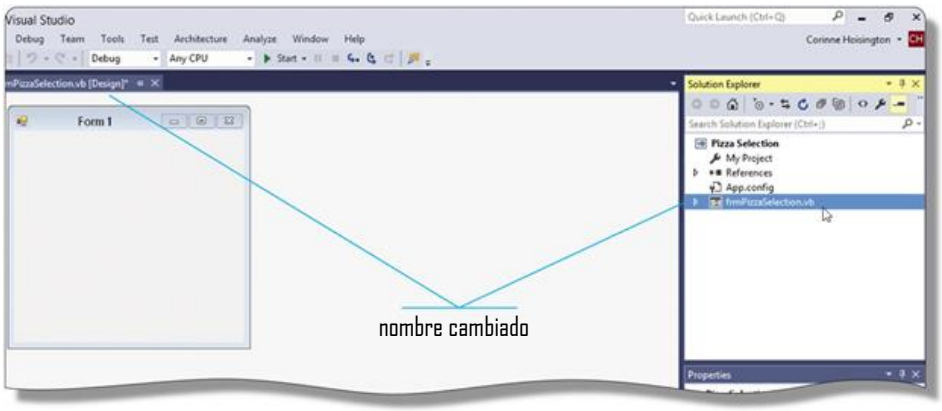
Por defecto *Visual Studio* asigna un nombre al proyecto que está trabajando en el *GUI* de *Visual Basic* (*Form1.vb [Design]*). Un desarrollador debe asignar un nombre significativo a un objeto para que el programa pueda hacer referencia a él si es necesario.

Nota: En la medida que vayamos avanzando en el material, la mayoría de los términos se utilizarán en inglés para que vaya familiarizándose con el programa *Visual Studio*.

Cuando damos nombre al proyecto al proyecto y los objetos no se permiten espacios u otros caracteres especiales en el nombre del objeto. Además, una buena práctica realizada por los desarrolladores es que cada nombre de objeto debe comenzar con un prefijo que identifique el tipo de objeto. Este prefijo se coloca en letras minúsculas. Para objetos *Windows Form*, el prefijo es ***frm*** (viene de formulario). Otros prefijos son: ***btn*** para botones, ***lbl*** para etiquetas, y ***pic*** para cuadro de imágenes. Siempre debe de seguir el método correcto para nombrar el proyecto y los objetos.

Pasos para nombrar el proyecto:

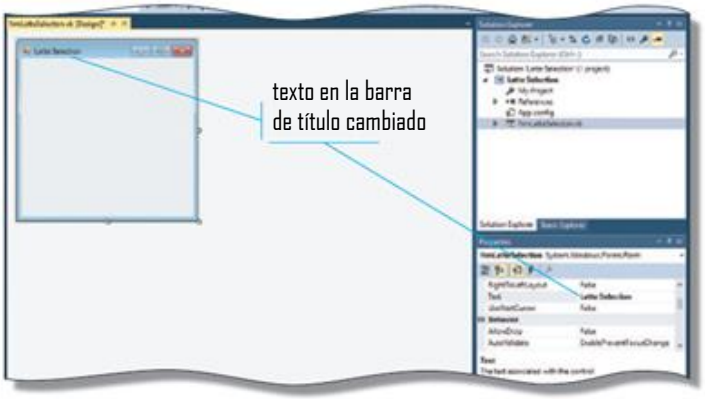
1. Haga clic en cualquier parte del objeto *Windows Form* para seleccionarlo.
2. Si no se muestra la ventana de *Solution Explorer*, en la barra de menú, haga clic en *View.* Haga clic en *Solution Explorer* para mostrar la ventana de *Solution Explorer*.
3. En la ventana de *Solution Explorer*, haga clic con el botón derecho del *mouse* sobre el nombre del archivo *Form1*.vb para mostrar una lista desplegable.
4. Haga clic en *Rename* y escriba el prefijo **frm** seguido por el nombre deseado (ej.: *frmLatteSelection*.vb; la extensión vb representa el programa utilizado).
5. Presiona la tecla *Enter*.



Después de nombrar el proyecto, a menudo el siguiente paso en el diseño de la *GUI* es cambiar el texto de la barra de título para reflejar la función del programa. Se utiliza la propiedad ***Text*** en la ventana *Properties* para cambiar el texto de la barra de título.

Establecer el texto en la barra de título del objeto Windows Form:

1. Selecciona el objeto *Windows Form* (*Form1*).
2. En la ventana *Properties*, busca la propiedad *Text*. La propiedad *Text* contiene la información que se muestra en la barra de título del objeto *Windows Form*.
3. En la columna derecha de la propiedad *Text*, haga doble clic en el espacio.
4. Escriba el nombre deseado.
5. Presiona la tecla *Enter*.



Cambiar tamaño del objeto Windows Form:

1. Selecciona el objeto *Windows Form* (*Form1*).
2. En la ventana *Properties*, busca la propiedad *Size* y en la columna derecha de la propiedad *Size*, haga doble clic en el espacio.
3. Esciba el número exacto de píxeles horizontales y verticales deseados.
4. Presiona la tecla *Enter*.

Actividad de Avalúo 2: Unidad 2 ➡ 2.1

Realizar las contestaciones en la HOJA DE CONTESTACIONES (final del Módulo).

Ejercicio 1: Defina los siguientes términos:

21. Proyecto de Escritorio Clásico de Window
22. Objeto Windows Form
23. Propiedades
24. Propiedad de Texto

Ejercicio 2: Circule la respuesta correcta para cada pregunta.

25. No se permiten colocar _____ en el nombre de un objeto.
    a. espacios y carácteres especiales
    b. letras
    c. números
    d. todas las anteriores

26. Un(a) _____ es equivalente a un solo programa creado con Visual Studio.
    a. aplicación
    b. proyecto
    c. comando
    d. matrix

27. Para crear un nuevo proyecto en Visual Studio, debes especificar tanto el tipo de aplicación que se desea crear como también el _____ que desea usar.
    a. esquema de color
    b. lenguaje de programación
    c. tamaño de fuente
    d. tamaño del Windows Form

28. Al crear un programa en Visual Studio, el objeto de Windows Form que está diseñando aparecerá en el _____ del programa *Visual Studio*.
    a. área de tareas
    b. área de diseño
    c. área de formularios
    d. área de trabajo

29. Al crear un programa en Visual Studio, un _____ es la ventana que utiliza para compilar el programa y que se mostrará en la pantalla cuando se ejecute el programa.
    a. objeto de Windows Form
    b. objeto de Pantalla de Windows
    c. objeto de Programa de Windows
    d. objeto de escritorio de Windows

30. Las(Los) _____ se utilizan para controlar las características de un objeto.
    a. atributos
    b. propiedades
    c. símbolos
    d. índices

31. De manera predeterminada, la ventana Properties del Programa Visual Studio se muestran en la sección _____ de la pantalla.
    a. superior
    b. inferior
    c. izquierda
    d. derecha

32. El texto predeterminado para el primer objeto de Windows Form creado en un proyecto es _____.
    a. 1Form
    b. Form
    c. Form1
    d. ThisForm

2.2 - Agregar Objetos al GUI

Después de configurar el objeto *Windows Form* según sea necesario para su proyecto, continue trabajando en el diseño de la interfaz de usuario agregando objetos al *Windows Form*, como por ejemplo: etiquetas, cuadros de imágenes o botones. Modifíque y coloque los objetos para que el formulario sea atractivo y fácil de usar. Para agregar objetos al *Windows Form* tenemos que tener acceso al panel de Caja de Herramientas (*Toolbox*). El *Toolbox* es la herramienta principal para colocar los objetos en la ventana del formulario, ya que contiene los componentes *.NET* que podemos utilizar para crear un programa.

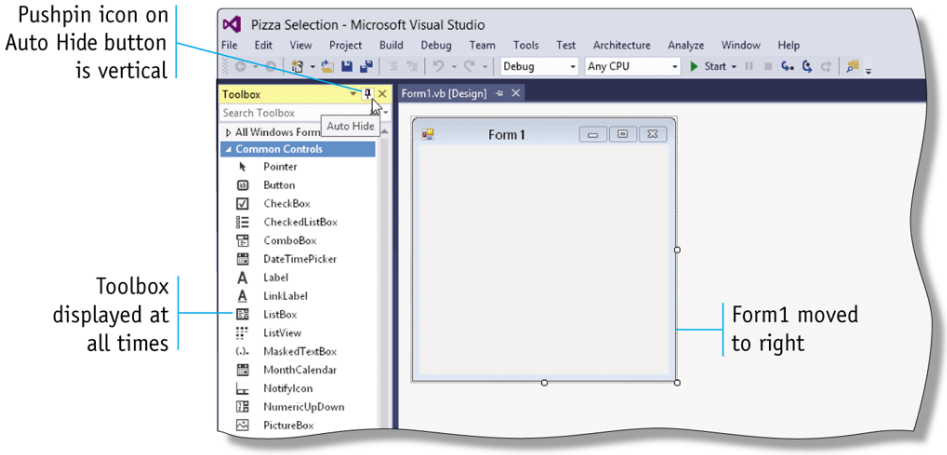
Mostrar el Toolbox:

1. Haga clic en el botón *Toolbox* en el margen izquierdo de la ventana.
2. Si es necesario, haga clic en *Common Controls* para mostrar los controles comunes de los componentes *.NET*.

Mostrar permanentemente el Toolbox:

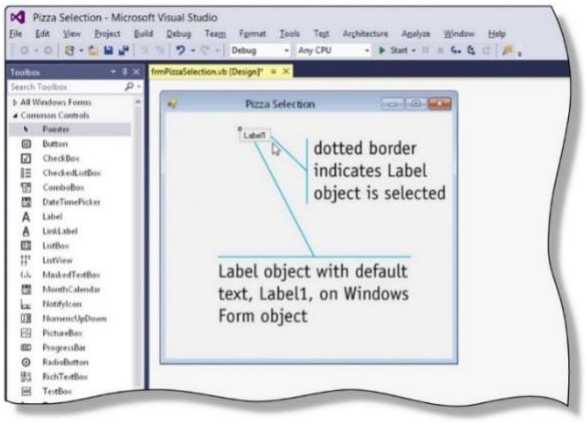
1. Con el *Toolbox* visible, haga clic en el botón *Auto Hide* (icono de la tachuela - *Pushpin icon*) en la barra de título de *Toolbox*.

El icono *Pushpin* va a cambiar a vertical que significa que el *Toolbox* se ha anclado a la ventana y el *Form1* se mueve a la derecha para que se pueda ver ambos. Cuando el icono *Pushpin* está vertical, se dice que el *Toolbox* está en modo acoplable, lo que significa que se puede arrastrar y colocar en cualquier lugar dentro de la ventana de *Visual Studio*.



Añadir un objeto al GUI:

1. En el panel de *Toolbox* localiza el objeto deseado.



2. Haga clic en el objeto y sin soltar arrastre sobre el objeto *Windows Form* a la ubicación donde desea colocar el objeto.
3. Cuando el objeto esté en la ubicación correcta, suelta el botón del mouse.

Pasos eliminar un objeto:

1. Haga clic en el objeto para seleccionarlo.
2. Presiona la tecla *Delete*.

Nombrar un objeto

Como con la mayoría de los objetos que coloca en el objeto *Windows Form*, el primer paso después de crear el objeto debe ser nombrarlo. Se nombra un objeto para identificar cada objeto con un nombre en particular a la hora de codificar. Al nombrar el objeto hay que identificarlo con el prefijo correspondiente del objeto seguido del nombre y se escribe todo unido. Se utiliza la propiedad ***Name*** en la ventana *Properties* para nombrar los objetos.

Algunos de los objetos más utilizados, su utilidad y prefijos son:

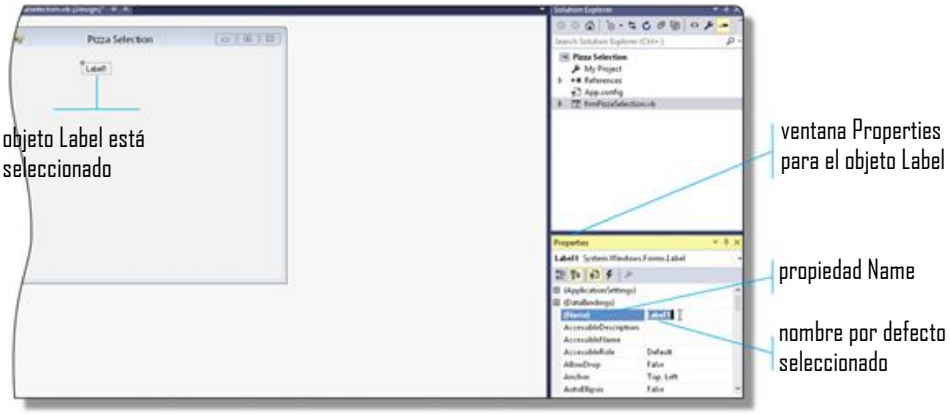
- Etiquetas (*Labels*): se utiliza para mostrar texto, descripciones, información para el programa; el prefijo es **lbl**
- Cuadro de imágenes (*PictureBox*): se utiliza para mostrar y cargar archivos de imagen; el prefijo es **pic**
- Botones (*Button*): generalmente se usa para generar un evento al hacer clic en el butón, el prefijo es **btn**
- Cuadro de texto (*TextBox*): permite al usuario ingresar datos en el cuadro de texto; el prefijo es **txt**

Pasos para nombrar un objeto:

1. Selecciona el objeto deseado en *Windows Form*.
2. En la ventana *Properties*, busca la propiedad *Name* y en la columna derecha de la propiedad *Name*, haga doble clic en el espacio.

3. Escriba el prefijo correspondiente para el objeto seguido por el nombre deseado (ejemplo: lblHeading)
4. Presiona la tecla *Enter*.

Nota: El nombre no se refleja en el objeto, ya que esta acción es para identificar los objetos al codificar.

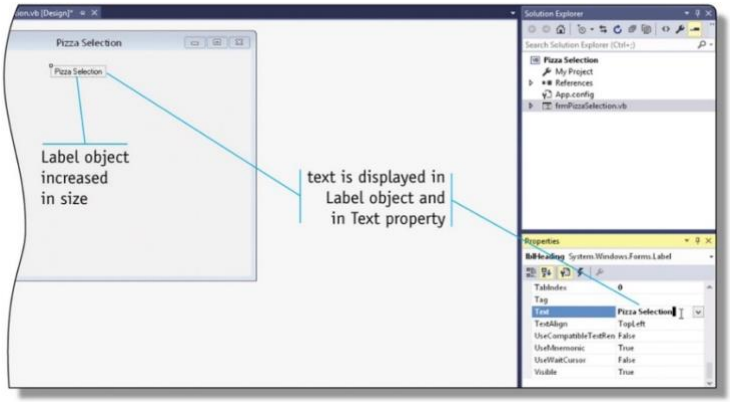


Cambiar el texto que se muestra en un objeto y añadir varias líneas de texto

Normalmente, los objetos que utilizas para mostrar información (ejemplo: objeto *label*) tienen un texto predeterminado. Puedes cambiar el texto en la propiedad ***Text*** en la ventana *Properties* y también escribir varias líneas de texto.

Pasos para cambiar el texto:

1. Selecciona el objeto al que desee cambiar el texto.
2. En la ventana *Properties*, busca la propiedad *Text* y en la columna derecha de la propiedad *Text*, haga doble clic en el espacio.
3. Escriba el nuevo texto deseado.
4. Presiona la tecla *Enter*.



Pasos para escribir varias líneas de texto:

1. Con el objeto seleccionado, busca la propiedad *Text* en la ventana *Properties* y en la columna derecha de la propiedad *Text*, haga doble clic en el espacio.
2. Haga clic en la flecha en la columna derecha de la propiedad *Text*.
3. Escriba el texto deseado.

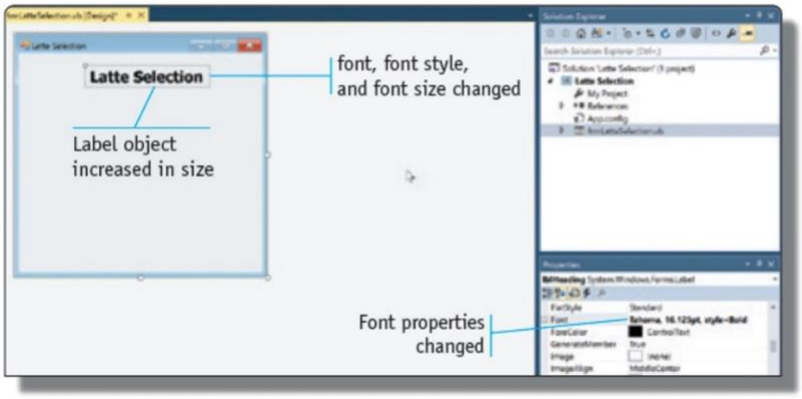
Cambiar la fuente de la etiqueta, el estilo de fuente y el tamaño de fuente

Nota: Cuando utilizamos la palabra fuente nos referimos al término en inglés de *Font*, o sea la letra.

La fuente, el estilo de fuente y el tamaño de fuente predeterminada en un objeto con texto a menudo deben cambiarse para reflejar el propósito de la etiqueta. Por ejemplo, en un *label* utilizado como encabezado, o título principal, el texto debe ser más grande que el tamaño de fuente que viene por defecto (8 puntos) utilizado para otros objetos labels, y debe estar en negrita para destacar como encabezado. Para cambiar la fuente, el estilo y el tamaño de un objeto, se utiliza la propiedad ***Font*** en la ventana *Properties* para realizar los cambios.

Pasos para cambiar tipo, estilo y tamaño de fuente:

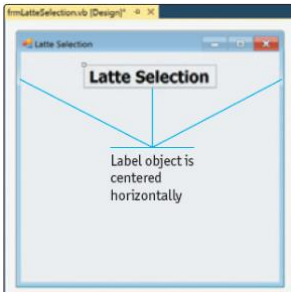
1. Haga clic en el objeto para seleccionarlo.
2. En la ventana *Properties*, busca la propiedad *Font* y haga clic en el nombre (columna izquierda).
3. Haga clic en las elipsis (puntos suspensivos).
4. En la ventana de diálogo de *Font* que aparecerá, selecciona el tipo de letra, estilo y tamaño deseado.
5. Haga clic en el botón OK.



Centralizar objetos en *Windows Form*

Cuando coloca un objeto en el *Windows Form*, es posible que el objeto no se encuentre exactamente en la posición correcta, por lo que se debe alinear el objeto. Los objetos se pueden centralizar horizontal y/o verticalmente en el *Windows Form*.

Pasos para centralizar un objeto:

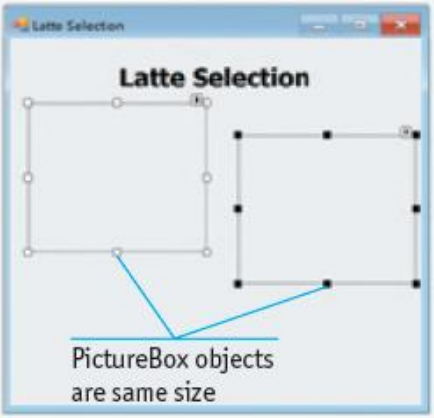


1. Selecciona el objeto que desea centralizar.
2. Haga clic en *Format* en la barra de menú.
3. Haga clic en *Center in Form*
4. Haga clic en la opción deseada en el submenú que aparece: *Horizontally* y/o *Vertically*.

Aplicar el mismo tamaño a otros objetos

Una vez hayas colocado los objetos en el *Windows Form* puedes hacer que sean del mismo tamaño. El primer objeto seleccionado será el que se tomará en cuenta para aplicar ese tamaño a los demás objetos. Se puede aplicar el mismo tamaño de los objetos considerando solo el ancho o el alto o ambos.

Pasos para aplicar mismo tamaño:



1. Selecciona el objeto cuyo tamaño desea duplicar, mantenga presionada la tecla CTRL y toque o haga clic en el objeto que desea cambiar de tamaño
2. Haga clic en *Format* en la barra de menú y luego coloque el puntero en *Make Same Size*.
3. Haga clic en la opción deseada del submenú que aparece: *Width*, *Height* o *Both*.

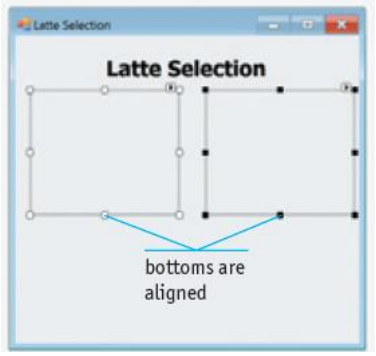
Alinear los objetos

Al diseñar una interfaz gráfica de usuario (GUI), debe considerar alinear los objetos para crear una apariencia limpia y ordenada para el usuario. Los objetos se alinean en forma horizontal (izquierda y derecha) y/o vertical (arriba y abajo). Igual que se hace al aplicar el mismo tamaño a otros objetos, se hace para alinear todos objetos a la misma vez, el primer objeto seleccionado es el que se tomará en cuenta para

alinear a los demás objetos. O sea, que los demás objetos serán alineados con el primer objeto seleccionado.

Pasos para alinear los objetos:

1. Con los objetos seleccionados, haga clic en *Format* en la barra de menú y luego seleccione *Align* en el submenú.
2. Haga clic en la opción deseada: *Left*, *Right*, *Center*, *Top*, *Bottom*, *Middle*.



Grabar proyecto de *Visual Basic*

Mientras trabaja en un proyecto *Visual Basic*, debe guardar su trabajo regularmente. Algunos desarrolladores guardan su proyecto cada 10 a 20 minutos, mientras que otros esperar terminar todo para salvar su trabajo. De todos modos, es importante desarrollar el hábito de salvar regularmente su trabajo.

Para guardar el trabajo que ha completado, puede hacer clic en el botón *Save All* que se encuentra en la barra de herramientas estándar. Seleccione la ubicación donde desea guardar su programa. Puede usar una unidad USB, el disco duro de su computadora o una unidad de red. Después de guardar el programa por primera vez, haga clic en el botón *Save All* en la barra de herramientas de estándar para guardar su programa en la misma ubicación con el mismo nombre.

Abrir un proyecto de *Visual Basic*

Después de guardar un proyecto y cerrar *Visual Studio*, a menudo querrá abrir el proyecto y trabajar en él nuevamente. Para abrir un proyecto guardado, puede usar uno de varios métodos:

Método 1: Haga doble clic en el archivo de solución en la carpeta donde está almacenado. Este método abre el archivo y le permite continuar su trabajo.

Método 2: Con *Visual Studio* abierto, presione el botón *Open File* en la barra de herramientas estándar. Ubique este archivo de solución y ábralo de la misma manera que lo hace para la mayoría de los programas de *Windows*.

Método 3: Con *Visual Studio* abierto, haga clic en *File* en la barra de menú y luego seleccione *Recent Projects and Solutions* en el menú *File*. Se muestra una lista de proyectos recientes. Haga clic en el nombre del proyecto que desea abrir. Es posible que este método no funcione bien si no está utilizando su computadora, ya que puede que su proyectos no aparezcan en la lista.

2.3 - Ciclo de Desarrollo de una Aplicación

El **Ciclo de Desarrollo de una Aplicación** es un conjunto de fases y pasos que los desarrolladores siguen para diseñar, crear y mantener un programa de computadoras. Las fases del ciclo de vidad de desarrollo del programa son las siguientes:

1. **Reunir y analizar los requisitos del programa**. El desarrollador debe obtener información que identifique los requisitos del programa y luego documentar estos requisitos. Reúna los requisitos del proyecto entrevistando a los usuarios, revisando los procedimientos actuales y completando otras tareas de recopilación de datos. Dos tipos de documentación:
   - Documento de requisitos (*Requirements document*)
   - Definición de caso de uso (*Use Case Definition*)

2. **Diseñar la interfaz de usuario**. Después de que el desarrollador comprende los requisitos del programa, el siguiente paso es diseñar la interfaz de usuario. La interfaz de usuario proporciona el marco de referencia para el procesamiento que ocurrirá dentro del programa. Los diseños de interfaz a menudo se denominan maquetas (*mock-ups*).
3. **Diseñar los objetos de procesamiento del programa**. Un programa de computadora esta formado de uno o más objetos de procesamiento que realizan la tarea requerida por el programa. El desarrollador debe determinar qué objetos de procesamiento se requieren y luego determinar los requisitos de cada objeto.
4. **Codificar el programa**. Después de que se haya diseñado un objeto de procesamiento, el objeto debe implementarse en el código del programa. El **código del programa** es el conjunto de instrucciones, escritas usando un lenguaje de programación como *Visual Basic*, que ejecuta una computadora.
5. **Probar el programa**. A medida que se codifica el programa y una vez que se completa la codificación, el desarrollador debe probar el código del programa para asegurarse de que se está ejecutando correctamente. El proceso de prueba es continuo e incluye una variedad de etapas.
6. **Documentar el programa o sistema**. A medida que los programas se diseñan y codifican, y una vez que se completa el proceso, el desarrollador debe documentar el programa. Documentar un programa significa usar un método prescrito para escribir las instrucciones sobre como usar el programa, la forma en que el programa realiza su tarea y otros elementos que los usuarios, desarrolladores y administradores pueden requerir.
7. **Mantener el programa o sistema**. Este es el proceso de cambiar y actualizar un programa. Este proceso se realiza después de que un programa se ponga en uso, y probablemente requerirá de modificaciones en el futuro.

El ciclo de vida de desarrollo del programa rara vez se realiza de manera lineal, con una fase completada antes de que comiencen las siguientes. Por el contrario, los programas se desarrollan de manera interactiva, lo que significa que las fases y los pasos dentro de estas deben repetirse varias veces antes de que se complete el programa.

Actividad de Avalúo 3 : Unidad 2 ➡ 2.2, 2.3

Realizar las contestaciones en la HOJA DE CONTESTACIONES (final del Módulo).

Ejercicio 1: Contesta:

33. Opciones para aplicar el mismo tamaño a varios objetos colocados en el Windows Form.
34. Prefijo para nombrar el objeto PictureBox.
35. Herramienta principal que se utiliza para colocar objetos en el Windows Form.
36. Propiedad para cambiar el texto.
37. Prefijo para nombrar el objeto Label.
38. Propiedad para cambiar el estilo de fuente.
39. Opciones al alinear los objetos colocados en el Windows Form.
40. Modo para identificar que el Toolbox está anclado e icono que lo indica.
41. Prefijo para nombrar el objeto Button.
42. Propiedad para nombrar un objeto.
43. Propiedad para cambiar la fuente.
44. Opciones para centralizar objetos colocados en el Windows Form.

Ejercicio 2: Identifica la fase del Ciclo de Desarrollo de una Aplicación.

45. Desarrollador debe determinar qué objetos de procesamiento se requieren.
46. Debe asegurarse de que se está ejecutando correctamente.
47. Se usa un método prescrito para escribir las instrucciones sobre cómo usar el programa.
48. Tener un marco de referencia para el procesamiento que ocurrirá dentro del programa.
49. Proceso de cambiar y actualizar un programa.
50. Desarrollador obtiene información que identifique los requisitos del programa y luego los documenta.
51. El objeto de procesamiento debe implementarse en el código del programa.

UNIDAD 3: DISEÑO DE PROGRAMA Y CODIFICACIÓN

Estándar: Programación y desarrollo de aplicaciones

Objetivos:
1. Define terminología de programación orientada a objetos.
2. Identifica estructuras de programación.
3. Eligue el lenguaje de programación apropiado para el desarrollo de aplicaciones en tareas específicas.
4. Aplica principios de diseño a tareas de programación.
5. Desarrolla programas tanto procesales como orientados a objetos.
6. Prueba y depura el código de una aplicación.

Nuevos Elementos

En la Unidad 2 se discutió la creación del diseño de una interfaz gráfica de usuario (*GUI*). Si bien los usuarios y otras personas pueden aprobar el diseño del *Windows Form* como funcional, el desarrollador normalmente debe realizar una variedad de cambios en el *GUI* para preparar la versión de producción real del programa. Entre estos cambios, o nuevos elementos de esta unidad, están los siguientes:

- Agregar color a la interfaz para que sea más atractiva visualmente.
- Adquirir e incluir imágenes que se requieren para el programa.
- Configuración de propiedades de interfaz de acuerdo con las necesidades del programa.

3.1 - Ajustar la interfaz gráfica de usuario

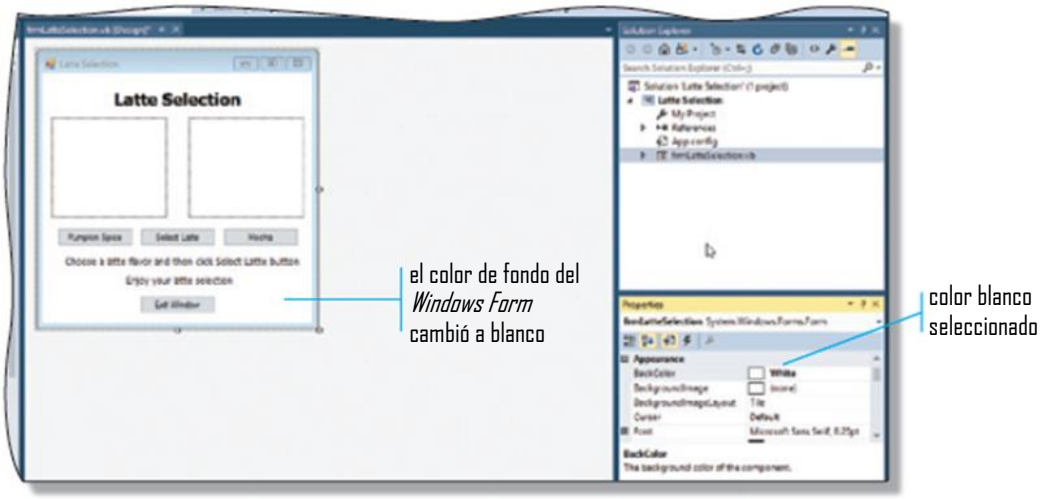
Durante la discusión de la Unidad 2 se discutió sobre algunas propiedades de los objetos en *Visual Studio*, incluidas la propiedad *Name* y la propiedad *Text*. En la ventana *Properties* hay más opciones de propiedades disponibles para cada uno de los objetos en la *GUI*. En muchos casos, se configuran estas propiedades para ajustar la interfaz de usuario y hacerla más útil.

Propiedades de *BackColor* y *ForeColor*

El *BackColor* de un objeto es el color de fondo que se muestra en el objeto. *Visual Studio* usa el color gris por defecto como el *BackColor*. Puede cambiar el color de fondo del objeto utilizando la propiedad ***BackColor*** en la ventana *Properties*. También en la ventana *Properties* encuentras la propiedad *ForeColor*. El *ForeColor* de un objeto es el color que se muestra en el texto del objeto.

Pasos para cambiar el BackColor:

1. Haga clic en el objeto al cual desea cambiarle el color de fondo.
2. En la ventana *Properties*, busca la propiedad *BackColor* y haga clic en la columna derecha de la propiedad *BackColor*.
3. Haga clic en la flecha hacia abajo en la propiedad *BackColor*.
4. Haga clic en la pestaña *Web* para mostrar los colores disponibles en esta opción.
5. Busca el color deseado y haga clic en el color.



Para cambiar el color de texto, realizas los mismos pasos del *BackColor*, pero utilizando la propiedad ***ForeColor***.

Agregar una imagen a un objeto *PictureBox*

Cuando desee mostrar una imagen en *Windows Form*, debe colocar primero el objeto *PictureBox* en el *Windows Form*. Los objetos *PictureBox* se usan para mostrar una imagen gráfica.

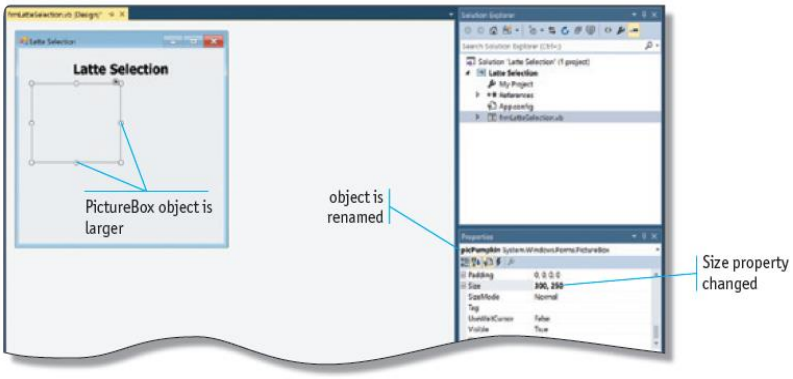
Pasos para agregar un cuadro de imágenes (PictureBox):

1. En el panel de Toolbox localiza el objeto *PictureBox*.
2. Haga clic en el objeto y sin soltar arrastre sobre el *Windows Form* a la ubicación donde desea colocar el objeto.
3. Cuando el puntero esté en la ubicación correcta, suelta el botón del mouse.

Recuerde nombrar el objeto *PictureBox* una vez sea añadido al *Windows Form* con el prefijo correspondiente (pic). *(Ver Pasos para nombrar un objeto).*

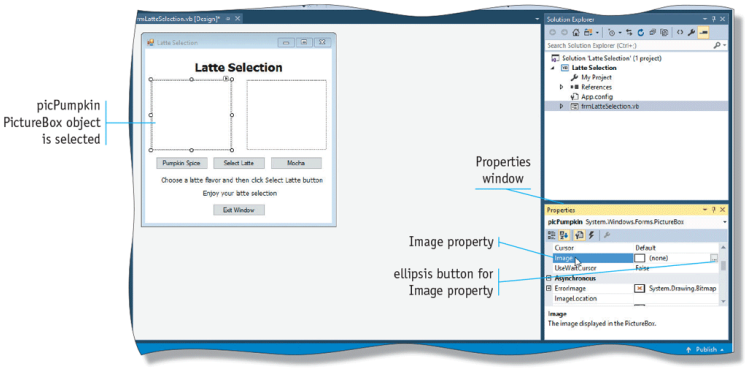
Pasos para modificar tamaño del PictureBox:

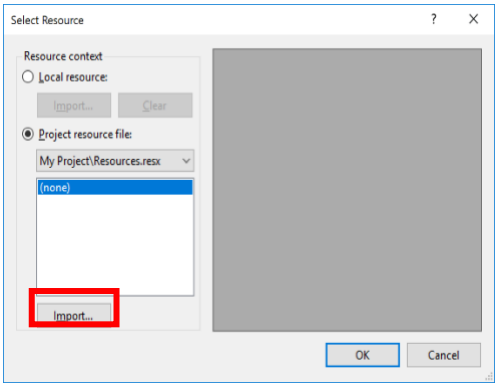
1. Haga clic en el objeto para seleccionarlo.
2. En la ventana *Properties*, busca la propiedad ***Size*** y en la columna derecha de la propiedad *Size*, haga doble clic en el espacio.
3. Escriba el tamaño deseado.
4. Presiona la tecla *Enter*.



Pasos para añadir la imagen al PictureBox:

1. La imagen debe haber sido guardada en su computadora o en un dispositivo externo.
2. Haga clic en el objeto *PictureBox* al que desea añadir una imagen.
3. En la ventana *Properties*, busca la propiedad ***Image***, haga clic en el nombre (columna izquierda).
4. Haga clic en las elipsis (puntos suspensivos) en la columna derecha de la propiedad *Image*.





5. Haga clic en el botón *Import* en el cuadro de diálogo.
6. Busca y selecciona la imagen deseada.
7. Haga clic en el botón *Open* en el cuadro de diálogo que se muestra.
8. Haga clic en el botón *OK*.

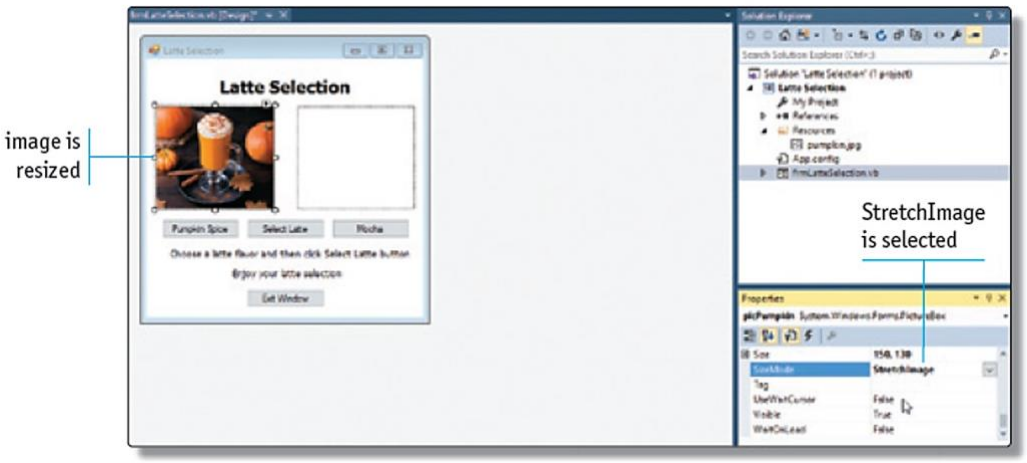
La imagen se mostrará en el objeto PictureBox y en adición se crea por defecto una carpeta con el nombre *Resources* en la ventana de *Solution Explorer*, lo que indica que ese cartapacio es ahora parte del programa y ahí es donde están las imágenes importadas.

Ajustar dimensiones de una imagen

En la mayoría de los casos, cuando importa una imagen a un programa, la imagen no se ajustará perfectamente en el objeto *PictureBox* porque los dos elementos tienen tamaños o dimensiones diferentes. El desarrollador debe ajustar el tamaño de la imagen para que quepa en el objeto *PictureBox* o ajustar el tamaño del objeto *PictureBox* para acomodar la imagen. Si el objeto *PictureBox* debe permanecer en su tamaño actual, la imagen debe ajustarse utilizando la propiedad ***SizeMode***.

Pasos para ajustar el tamaño de la imagen en el PictureBox:

1. Selecciona el objeto *PictureBox*.
2. En la ventana *Properties*, busca la propiedad *SizeMode* y haga clic en el nombre (columna izquierda).
3. Haga clic en la flecha en la columna derecha de la propiedad *SizeMode*.
4. En la lista desplegable, haga clic en la opción deseada:
   - *Normal*: imagen tiene el tamaño en la que fue realizada, es la que viene por defecto
   - *CenterImage*: imagen se muestra al centro del *PictureBox*
   - *StretchImage*: la imagen se ajusta al tamaño del *PictureBox*
   - *AutoSize*: el *PictureBox* se ajusta al tamaño de la imagen
   - *Zoom*: se aumenta o disminuye el tamaño de la imagen



Establecer la propiedad *Visible*

La propiedad ***Visible*** controla si un objeto añadido al GUI en el *Windows Form* se muestra tan pronto aparece el programa en pantalla o va a esperar un código para mostrar el objeto. Por defecto, la propiedad *Visible* se establece en *True* para que cualquier objeto que coloque en el *Windows Form* se muestre cuando se ejecute el programa. Si no desea que se muestre un objeto cuando el programa inicie, debe establecer la propiedad *Visible* con la opción *False* y luego aplicar un código para mostrar el objeto una vez se ejecute una acción.

Pasos para cambiar la propiedad Visible:

1. Selecciona el objeto al que desea cambiar la propiedad de visibilidad.
2. En la ventana *Properties*, busca la propiedad *Visible* y haga clic en el nombre (columna izquierda).
3. Haga clic en la flecha en la columna derecha de la propiedad *Visible*.
4. En la lista desplegable, haga clic en la opción *False*. Cuando el programa se ejecute, el objeto no se mostrará.

Si desea que se muestre el objeto al ejecutarse el programa dejar la opción *True*.

Establecer la propiedad *Enabled*

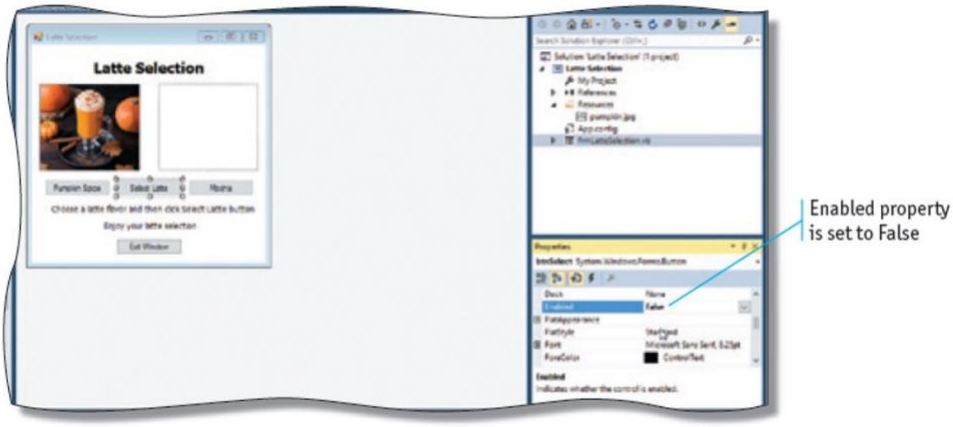
En un programa controlado por eventos, los objetos, ejemplo los objetos *Button*, se pueden utilizar para hacer que ocurran eventos. Por ejemplo, un objeto *Button* que ha sido agregado a un GUI está habilitado (*enabled*), lo que significa que se puede hacer clic para lograr que ocurra un evento.

La propiedad ***Enabled*** controla cuando un objeto está habilitado o activado y también controla cuando un objeto está desactivado o no disponible. Cuando un botón está desactivado significa que hacer clic en el botón no causa ninguna acción y se muestra atenuado (color translúcido). La selección por defecto para la propiedad *Enabled* al añadir este tipo de objetos, es *True*, lo que significa que el objeto está habilitado (activado). Para establecer que el objeto esté desactivado se selecciona la opción *False*.

Pasos para cambiar la propiedad Enabled:

1. Selecciona el objeto al que desea cambiar la propiedad de activado.
2. En la ventana *Properties*, busca la propiedad *Enabled* y haga clic en el nombre (columna izquierda).
3. Haga clic en la flecha en la columna derecha de la propiedad *Visible*.
4. En la lista desplegable, haga clic en la opción *False*. Cuando el programa se ejecute, el objeto estará desactivado.

Si desea que el objeto esté activado al ejecutarse el programa dejar la opción *True*.



Ejecutar programa

Cuando cambias algunas propiedades de un objeto, el efecto de sus cambios puede no ser evidente hasta que ejecute el programa. Por ejemplo, si cambias la propiedad *Visible* en *False* para un objeto *PictureBox* ningún cambio se podrán observar mientras este en el área de trabajo de *Windows Form*. Esta configuración, o cambios realizados a los objetos, se podrá observar solo cuando se ejecuta el programa.

Para garantizar que los cambios aplicados a las propiedades de los objetos se hayan realizado de forma correcta debe ejecutar el programa. La **ejecución del programa** significa que se compila, o traduce, los códigos de programación generado utilizando el lenguaje *Visual Basic* en una forma de instrucciones que la computadora puede ejecutar. Estas instrucciones se guardan y luego se ejecutan como un programa. Para ejecutar el programa que ha creado, puede hacer clic en el botón de *Start* en la barra de herramientas estándar.

Actividad de Avalúo 4: Unidad 3 ➡ 3.1

Realizar las contestaciones en la HOJA DE CONTESTACIONES (final del Módulo).

Ejercicio 1: Cierto o Falso. Si la premisa es cierta, escriba la **C**. Si la premisa es falsa, escriba la **F** y sustituya la(s) palabra(s) subrayada(s) para convertirla en cierta.

Ejemplo: Las propiedades Name y Text se encuentran en la ventana Solutions Explorer.
Respuesta: F, Properties

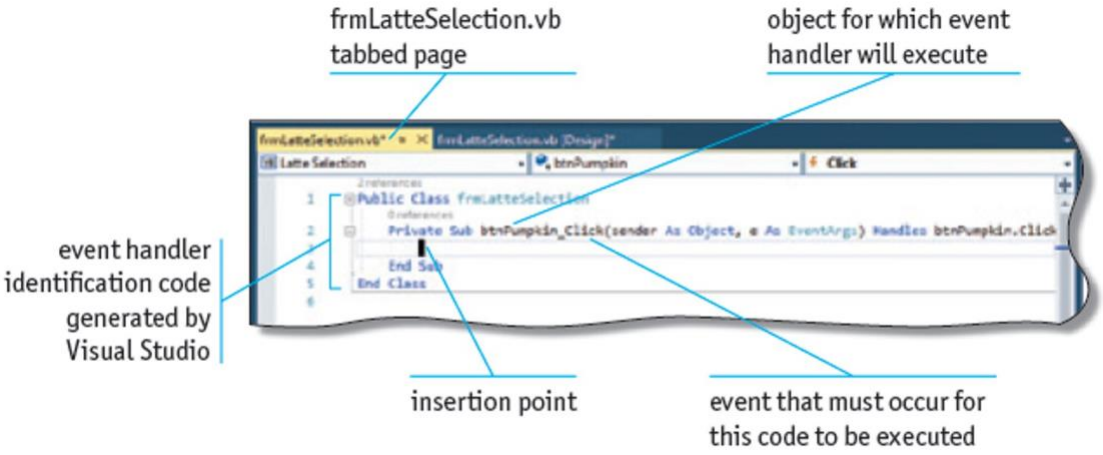
52. El objeto PictureBox se utiliza para mostrar una imagen.
53. Al ejecutar el programa, en la propiedad Visible del objeto picFrutas tiene la opción False, lo que indica es que la imagen se muestra en pantalla.
54. La propiedad que controla el color de texto es FontColor.
55. La opción Align Center es para alinear la imagen al centro en el PictureBox.
56. La propiedad Enabled controla cuando un objeto está activado o desactivado.
57. Al importar una imagen en el PictureBox, se crea el cartapacio Resources en la ventana Solution Explorer.
58. El BackColor de un objeto es el color de fondo que se muestra en el objeto seleccionado.
59. Cuando se importa una imagen para el objeto PictureBox, el tamaño que tiene por defecto es Fit To.
60. La configuración, o cambios realizados a los objetos, se podrá observar solo cuando se ejecuta el programa.
61. Al ejecutarse el programa el botón ya está activado por defecto con la opción False en la propiedad Enabled.
62. La opción StrechImage ajusta la imagen al tamaño del PictureBox.
63. En la propiedad Size se ajusta la imagen en el PictureBox.

3.2 - Códigos de Programación *Visual Basic*

El código de programación (*program code*), es el conjunto de instrucciones escritas por el desarrollador que le indican al programa que lleve a cabo el procesamiento requerido. Como ya se ha discutido, la mayoría del procesamiento en un programa controlado por eventos ocurre cuando el usuario activa el evento. Por ejemplo, cuando un usuario hace clic en un botón en la GUI, esto puede activar una acción que desencadene un evento y el programa realiza el procesamiento requerido. El desarrollador escribe el código de programación para llevar a cabo este procesamiento. Este código se coloca en una sección del programa llamada controlador de eventos **(*event handler*)**. Éste maneja el evento que la acción del usuario desencadena (ejemplo oprimir el botón), ejecutando el código que realiza el procesamiento requerido.

Para escribir el código de programación para un controlador de eventos, el desarrollador primero debe identificar el objeto en la GUI que se utilizará para desencadenar el evento. Una vez identificas el objeto a codificar, haces doble clic en el objeto y aparecerá la ventana de código (*Window Code*).

Ventana de códigos



Pasos para crear el controlador de eventos:

1. Haga doble clic sobre el objeto que desea programar.

   Esto causará que se muestre la ventana de códigos, o ventana de programación, en donde el desarrollador procederá a escribir la codificación.

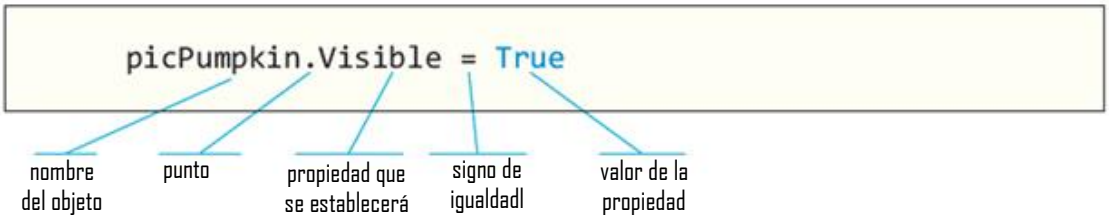
2. Escriba la declaración de codificación. *(Se explicará en el siguiente tema).*

Declaraciones de codificación (*Coding Statements*) de *Visual Basic* 2017

Una declaración de codificación de *Visual Basic* 2017 contiene instrucciones que la computadora eventualmente ejecuta. La sintaxis de *Visual Basic* especifica cómo se debe escribir cada instrucción. Cuando el usuario toca o hace clic en un botón mientras el programa se está ejecutando, se producirá una acción asociada con la instrucción de codificación.

La figura a continuación, muestra una declaración de codificación de *Visual Basic* que establece la propiedad *Visible* en *True* para un objeto *PictureBox* llamado *picPumpkin*, por lo que la imagen se mostrará en el cuadro de imagen después de que se ejecuta la declaración.

Ejemplo de una declaración de codificación



Explicación de la declaración de codificación en el ejemplo:

1. La primera parte de la declaración de codificación identifica el objeto que contiene la propiedad a establecer. Es el mismo nombre que le diste al objeto en la propiedad *Name* (ventana *Properties*). En el ejemplo, el nombre del objeto es *picPumpkin*, recordamos que el pic es el prefijo utilizado para los objetos *PictureBox*.
2. Al nombre del objeto le sigue un punto [.] sin espacios intermedios. El punto separa el nombre del objeto de la próxima entrada en la declaración de codificación, y es requerido.
3. Luego del punto se encuentra el nombre de la propiedad a establecer (ejemplos de propiedades ya discutidas son *Visible* y *Enabled*). En el ejemplo, el nombre de la propiedad es *Visible*. Recordará que la propiedad *Visible* determina si una imagen se muestra en el objeto *PictureBox* cuando se ejecuta el programa.
4. El nombre de la propiedad es seguido por un espacio, un signo de igualdad y otro espacio. Los espacios no son obligatorios, pero una buena práctica de codificación dicta que los elementos dentro de una declaración deben estar separados por un espacio para que la declaración sea más fácil de leer. Se requiere el signo de igualdad porque indica que sigue el valor que se utilizará

para establecer la propiedad. Este signo identifica la declaración de codificación como un ***assignment statement***, lo que significa que el valor en el lado derecho del signo de igualdad se asigna al elemento en el lado izquierdo del signo de igualdad, lo que en este ejemplo sería la propiedad.

5. Por último está el valor de la propiedad.

Cada entrada dentro de la declaración del programa debe ser correcta o el programa no se ejecutará. Esto significa que el nombre del objeto debe escribirse correctamente, el operador de puntos debe colocarse en la ubicación correcta, el nombre de la propiedad debe escribirse correctamente, el signo igual debe estar presente y una entrada válida para la propiedad debe seguir al signo de igualdad. Debido a que un protocolo correcto de programación dicta que solo debe haber una declaración de códigos por línea, el siguiente paso es presionar la tecla *Enter* para mover el punto de inserción a la siguiente línea en la ventana de código.

Formato General de la Declaración de Codificación:	
NombreDelObjeto.Propiedad = ValorDeLaPropiedad	
Ejemplo	*Resultado*
picPumpkin.Visible = True	Muestra la imagen picPumpkin
btnSelectLatte.Enabled = False	Inactiva el botón btnSelectLatte

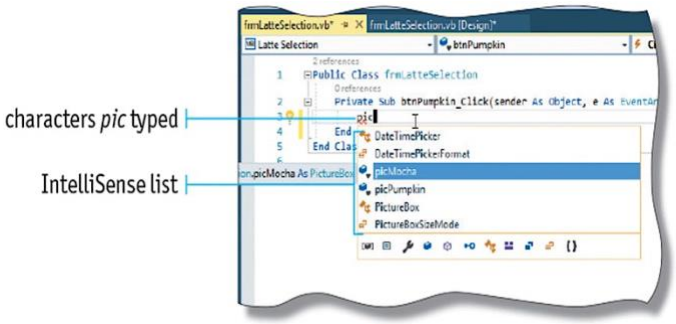
IntelliSense

Al estar en la ventana de código (*window code*), el punto de inserción está localizado donde se escribirá la declaración completa. *Visual Studio* provee ayuda para que sea menos propenso a cometer un error al escribir la declaración. Esta función de ayuda se llama ***IntelliSense***.

IntelliSense muestra todas las entradas permitidas en una declaración de codificación de *Visual Basic* cada vez que se requiere colocar un punto (.), un signo de igualdad (=) u otro carácter especial para la declaración. En otras palabras, esta opción predice lo que se pretende escribir en la declaración de codígos, o la oración de programación. Cuando escribes el prefijo del objeto a codificar, la ventana de *IntelliSense* se muestra con los nombres de todos los objetos que colocaste en el *Windows Form* que comienzan con el prefijo que escribiste al comenzar a codificar.

En lugar de escribir el nombre del objeto, puedes seleccionarlo de la lista *IntelliSense*. Cuando escribes las primeras letras del nombre del objeto, *IntelliSense* muestra una lista de todos los objetos y otras entradas que se puede identificar en la declaración.

Visual Studio y *IntelliSense* automáticamente crean las sangrías en la declaración del programa y así hace que las declaraciones sean más fáciles de leer y comprender.



A medida que los programas se vuelven más complejos, la sangría adecuada de las declaraciones del programa puede ser un factor importante en el desarrollo de programas libres de errores. El uso de esta opción para ingresar declaraciones de códigos de *Visual Basic* proporciona dos ventajas significativas:

- es más rápido ingresar una oración de códigos usando *IntelliSense* que escribir la declaración completa
- el uso de *IntelliSense* reduce drásticamente la cantidad de errores al ingresar la declaración de códigos

Pasos para ingresar las declaraciones de codificación utilizando IntelliSense:

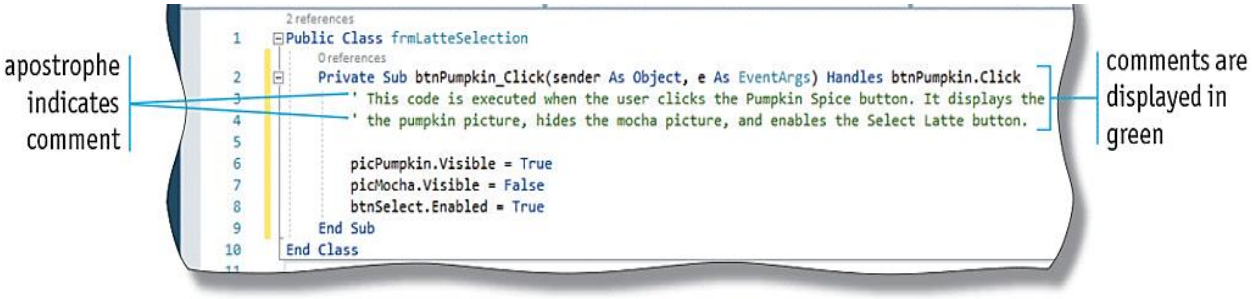
1. Haga doble clic sobre el objeto que desea programar.
2. En la ventana de códigos, escriba las primeras letras del nombre del objeto a programar.
3. En la lista *IntelliSense*, selecciona el nombre del objeto (si no está seleccionado) y escriba el punto.
4. Esciba las primeras letras de la propiedad.
5. En la lista *IntelliSense*, selecciona el nombre de la propiedad (si no está seleccionado) y presione la barra espaciadora.
6. Presiona el signo de igualdad y luego la barra espaciadora.
7. Escriba las primeras letras del valor de la propiedad.
8. En la lista IntelliSense, selecciona el valor (si no está seleccionado) y presione *Enter*.

Declaraciones de Comentarios en el programa

Un programa de *Visual Basic* bien escrito normalmente contiene declaraciones de comentarios (*comment statements*) dentro del propio código para documentar lo que

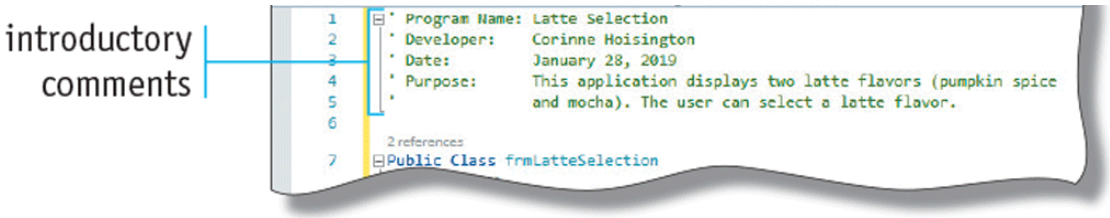
debe hacer la declaración de código. El propósito general de los comentarios es ayudar al lector a comprender cómo los códigos realizan la tarea.

Para ingresar un comentario antes se debe colocar un apóstrofe (') y luego la información deseada. Todos los caracteres (letras, números o símbolos) que siguen al apóstrofe en una línea de código se consideran un comentario. Cada vez que el compilador de *Visual Basic* encuentra un apóstrofe en el código, ignora los caracteres que le siguen en la línea. Para el compilador, es como si los comentarios no existieran. Los comentarios en el código se muestran en texto color verde y describen el procesamiento que ocurrirá en el código que sigue. Debido a que el compilador de *Visual Basic* ignora los comentarios, los desarrolladores no deben incluir la sintaxis del lenguaje de programación en los comentarios. Cualquier caracter está permitido dentro de los comentarios.



Comentarios de Introducción

Cada programa debe comenzar con comentarios que indiquen el nombre del programa, el nombre del desarrollador, la fecha y el propósito del programa. Estos comentarios de introducción deben escribirse antes de todo código de programación, incluso el código generado por *Visual Studio*.



Pasos para escribir los comentarios de introducción:

1. En la venta de códigos, haga clic a la izquierda de la palabra *Public* en la línea 1 del programa para colocar el punto de inserción en esa línea.
2. Presiona la tecla *Enter* y presione la tecla de la flecha de arriba para colocar el punto de inserción antes de lo que está escrito.
3. Escriba lo siguiente, escribiendo un apóstrofe antes de escribir en cada línea:

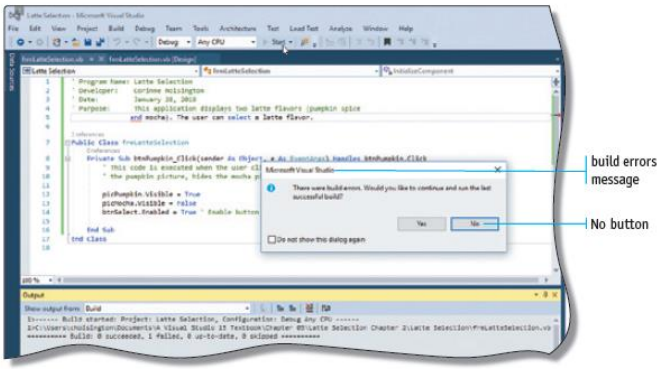
Luego de escribir el apóstrofe se da un espacio.

```
' Nombre del Programa:     Nombre del Programa
' Desarrollador:          Nombre del Desarrollador
' Fecha:                  Fecha de la creación del programa
' Propósito:              Propósito de la aplicación creada
```

Si la información del propósito toma más de una línea, se debe escribir el apóstrofe en cada línea para que sea considerado como un comentario. Los comentarios se recomienda que sean escritos en inglés para que vaya a la par con toda la información que hay en la ventana de códigos.

Corregir errores en el código

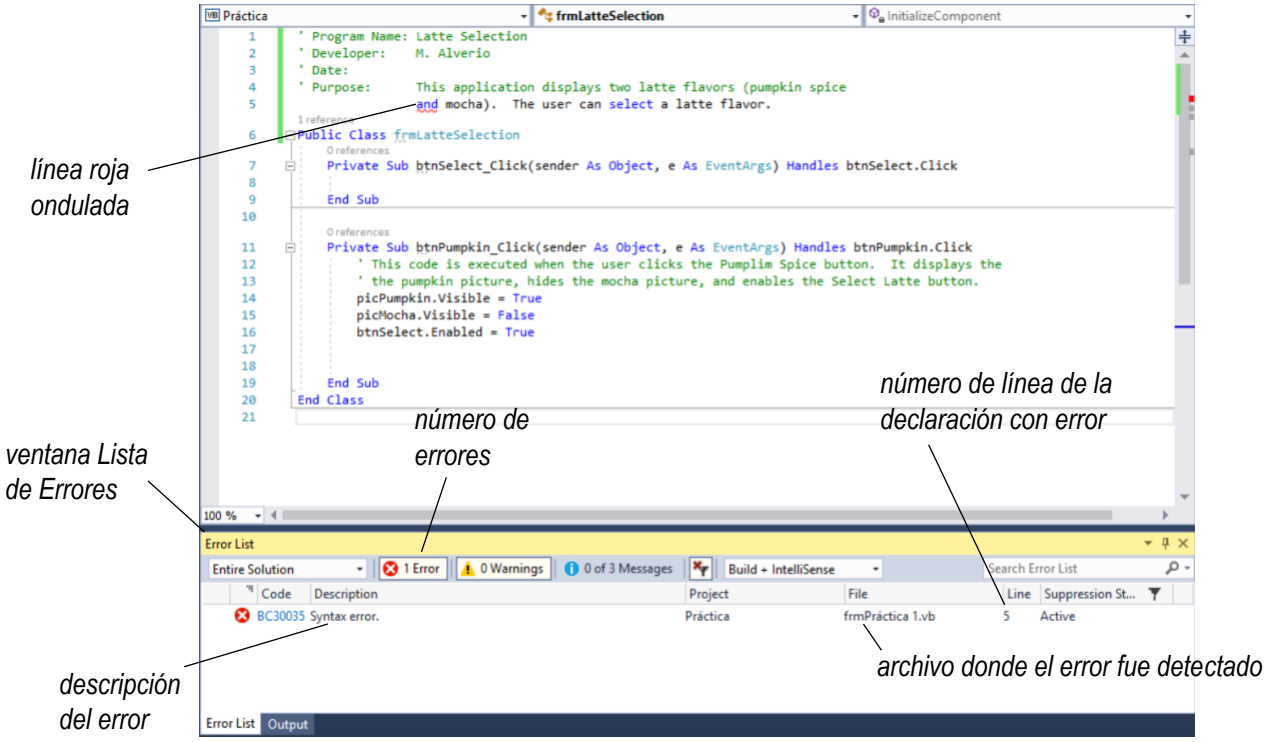
Debido a que se podrían crear errores al ingresar los códigos, debe entender qué hacer cuando se produce un error de codificación. Si escribes un código incorrecto, al ejecutar el programa *Visual Studio* muestra un **mensaje de errores de compilación** (*build errors message*). El *build errors message*, significa que el compilador de *Visual Basic* detectó un error de codificación en el programa. Es muy importante que al crear programas de *Visual Basic*, y surga un mensaje de errores de compilación, siempre haga clic en el botón No y seguir los pasos para hacer las correcciones. Bajo ninguna circunstancia debe continuar la ejecución del programa.



Cuando hace clic en el botón No, *Visual Studio* muestra el código del programa y la **ventana Lista de Errores** (*Error List window*). La ventana Lista de Errores identifica el número de errores que ocurrieron y muestra una descripción de los errores. Por

ejemplo, un error de sintaxis significa que *Visual Studio* esperaba encontrar un tipo diferente de declaración de códigos.

Ventana Lista de Errores



La ventana Lista de Errores identifica la cantidad de errores que ocurrieron, muestra las descripciones de los errores, archivo donde ocurrió el error y el número de la línea de la declaración con error. *Visual Studio* resalta el error con una línea roja ondulada o una línea roja en zigzag. El desarrollador puede escribir y reemplazar el texto resaltado con el código correcto. Con el error resaltado, el desarrollador debe examinar la declaración para determinar el error. Al corregir el error, se elimina la línea de errores. Cerramos la ventana Lista de Errores en el botón *Close* (x).

Procedimiento para cerrar el programa

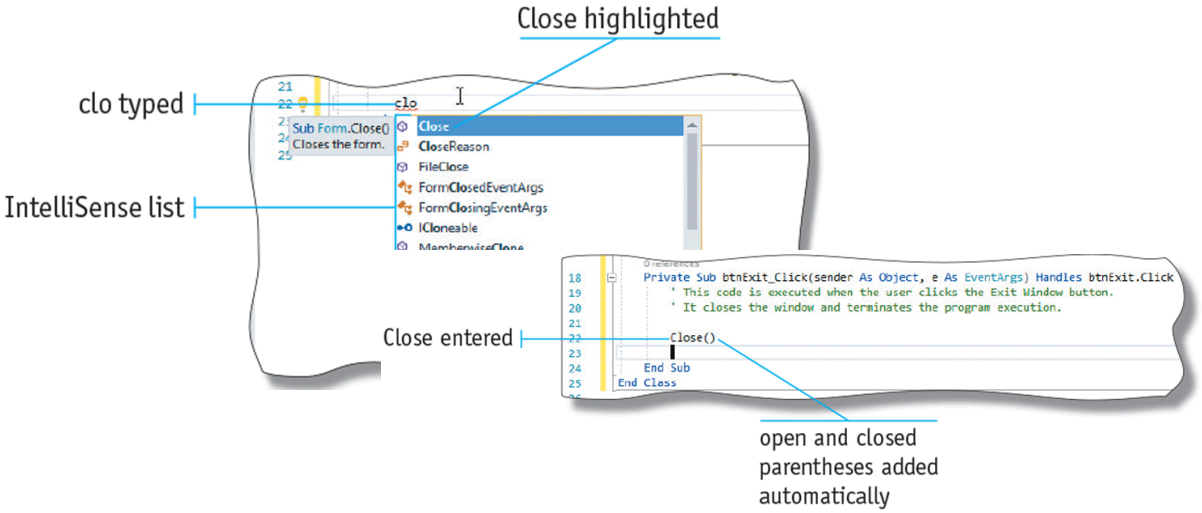
La instrucción de *Visual Basic* para cerrar la ventana y finalizar el programa invoca a un procedimiento para que realice la tarea. Un **procedimiento** es un conjunto de códigos preescritos que pueden ser invocados por una declaración en el programa *Visual Basic*. Cuando se invoca al procedimiento, el programa procesa el código. El procedimiento utilizado para cerrar una ventana y finalizar el programa es el

Procedimiento de Cierre (*Close Procedure*). La declaración para el Procedimiento de Cierre es: **Close()**.

La palabra *Close* especifica el nombre del procedimiento a invocar. Los paréntesis (izquierdo y derecho) inmediatamente después del nombre del procedimiento identifican la instrucción de *Visual Basic* como una **instrucción de llamada al procedimiento**.

Pasos para escribir la declaración para el procedimiento de cierre:

1. Con el punto de inserción colocado donde desea que aparezca la declaración, escriba las primeras letras de la declaración (clo).
2. En la lista desplegable *IntelliSense* selecciona la declaración (si no está seleccionada) y presione la tecla *Enter*.



Documento de Planificación de Eventos

Como has aprendido, los programas escritos usando un GUI normalmente son programas controlados por eventos. Un evento significa que el usuario ha iniciado una acción que hace que el programa realice el tipo de procesamiento apropiado en respuesta. Una vez que se ha creado el diseño propuesto para la interfaz de usuario, el desarrollador debe documentar los eventos que pueden ocurrir en función de la interfaz de usuario. El **Documento de Planificación de Eventos** (E*vent-Planning Document*) es una tabla en donde se especifican los objetos añadidos a la interfaz de

usuario que causarán eventos, las acciones tomadas por el usuario para activar los eventos y el procesamiento de eventos que debe ocurrir.

EVENT PLANNING DOCUMENT

Program Name:	Developer:	Object:	Date:
Latte Selection	Corinne Hoisington	frmLatteSelection	January 28, 2019
Object	**Event Trigger**	**Event Processing**	
btnPumpkin	Click	Display the pumpkin picture Hide the mocha picture Enable the Select Latte button	
btnSelect	Click	Disable the Pumpkin Spice button Disable the Select Latte button Disable the Mocha button Hide the Instructions label Display the Confirmation Message label Enable the Exit Window button	
btnMocha	Click	Display the mocha picture Hide the pumpkin picture Enable the Select Latte button	
btnExit	Click	Close the window and exit the program	

La primera columna (*Object*) identifica el objeto en la GUI que puede desencadenar un evento y está identificado con su nombre. La segunda columna (*Event Tigger*) identifica lo que desencadenará el evento, que es la acción que un usuario realiza para provocar el evento. La última columna (*Event Processing*) especifica el procesamiento de eventos que el programa debe realizar cuando se produce el evento. También se escribe en las primeras dos filas de la tabla el nombre del programa, nombre del desarrollador, nombre del formulario y fecha en qué fue creado el programa.

Esta lista de tareas, para cada evento, es un elemento crítico en el diseño del programa. Debe ser preciso y exacto. Ningún paso del procesamiento que deba ocurrir debe ser omitido de la columna de procesamiento de eventos. Las tareas deben colocarse en la misma secuencia que se realizarán en el programa.

Actividad de Avalúo 5: Unidad 3 ➡ 3.2

Realizar las contestaciones en la HOJA DE CONTESTACIONES (final del Módulo).

Ejercicio 1: Circule la(s) respuesta(s) correcta(s). para cada pregunta.

64. ¿Cuál de los siguientes es el formato correcto para asignar un valor a una propiedad?
    a. ValordelaPropiedad = Propiedad.NombredelObjeto
    b. NombredelObjeto = Propiedad.ValordelaPropiedad
    c. Propiedad.NombredelObjeto = ValordelaPropiedad
    d. NombredelObjeto.Propiedad = ValordelaPropiedad

65. Un (a) _____ se usa para comenzar una declaración de comentario.
    a. Punto y coma (;)
    b. Punto (.)
    c. Apostrofe (‘)
    d. Coma (,)

66. Esta opción predice lo que se pretende escribir en la declaración de códigos, o la oración de programación.
    a. Window Code
    b. Coding Statements
    c. IntelliSense
    d. Assignment Statement

67. Al ejecutar el programa y Visual Basic encontrar un error, lo resalta con un(a):
    a. línea ondulada verde
    b. texto azul
    c. texto verde
    d. línea roja ondulada

68. El Procedimiento de Cierre _____ se utiliza para cerrar el programa.
    a. Exit
    b. Close()
    c. Close
    d. Exit()

69. Conjunto de instrucciones escritas por el desarrollador que le indican al programa que lleve a cabo el procesamiento requerido.
    a. código de programación
    b. declaración de codificación
    c. controlador de eventos
    d. procedimiento de cierre

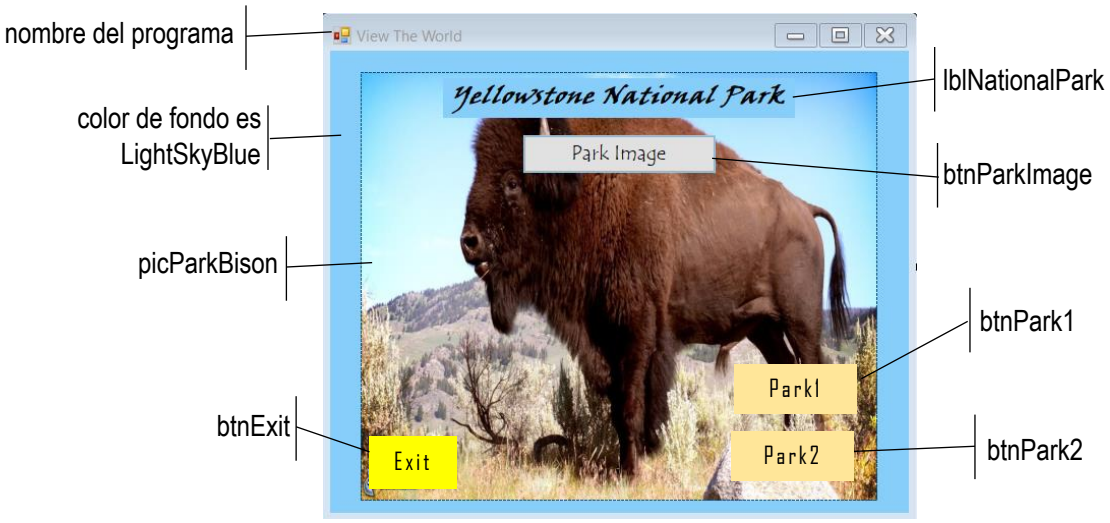
70. En el(la) _____ se especifican los objetos añadidos a la interfaz de usuario que causarán eventos.
    a. Documento de Planificación de Eventos
    b. Declaración de Codificación
    c. Ventana de Códigos
    d. Controlador de Eventos

71. Los comentarios en la ventana de código se muestran en texto color:
    a. gris
    b. rosa
    c. verde
    d. rojo

72. Algunas de las opciones que puedes encontrar en la ventana Lista de Errores son:
    a. número de errores, cómo arreglar el error
    b. descripción del error, número de la llínea donde está el error
    c. cómo arreglar el error, alternativas
    d. dónde ocurrió, alternativas

73. Un(a) _____ contiene instrucciones que la computadora eventualmente ejecuta.
    a. controlador de eventos
    b. procedimiento de cierre
    c. código de programación
    d. declaración de codificación

Ejercicio 2: Analiza la siguiente Interfaz Gráfica de Usuario y contesta las preguntas.



74. ¿Cuál es la declaración de codificación de Visual Basic para ver la imagen que se muestra cuando el usuario haga clic en el objeto Park1?
75. ¿Qué propiedad y opción se establece para que el objeto ParkImage y el objeto Exit estén desactivados al ejecutar el programa?
76. Para cambiar el color de fondo de la ventana a LightSkyBlue como se muestra, ¿qué propiedad debe utilizar?
77. ¿Qué propiedad se usó para modificar el texto del título Yellowstone National Park?
78. ¿Cuál es la declaración de codificación de Visual Basic para que el objeto Exit se active?

Este programa fue desarrollado por usted y debe estar listo para la fecha del *12 de noviembre de* ----. El propósito de la aplicación es que el usuario puede seleccionar entre dos parques. ¿Cómo se deberían escribir los comentarios de introducción de esta aplicación?

79. _____ 81. _____
80. _____ 82. _____

¿Cuáles son las declaraciones de codificación de *Visual Basic* para que cuando el usuario haga clic en el objeto ParkImage los objetos Park1 y Park2 se desactiven y el objeto Exit se active?

83. _____ 85. _____
84. _____

UNIDAD 4: VARIABLES Y OPERACIONES ARITMÉTICAS

Estándar: Programación y desarrollo de aplicaciones

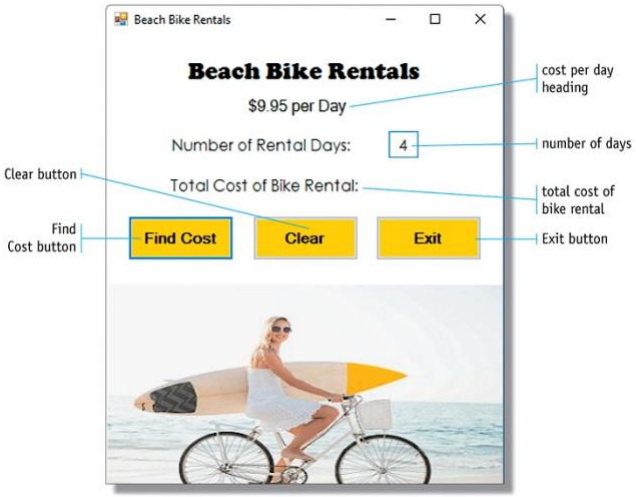
Objetivos:

1. Identifica estructuras de programación.
2. Elige el lenguaje de programación apropiado para el desarrollo de aplicaciones en tareas específicas.
3. Utiliza lenguajes de secuencias de comandos en el desarrollo de aplicaciones.
4. Selecciona el compilador apropiado.
5. Codifica tareas comunes utilizando herramientas de desarrollo de aplicaciones.
6. Codifica una solución de programa en más de un lenguaje de programación.
7. Prueba y depura el código de una aplicación.

4.1 - Insertar Variables y Operaciones en el Diseño de la Interfaz

En las demostraciones anteriores, se crearon botones para activar eventos al hacer clic en ellos, pero no ingresó datos. En muchas aplicaciones, los usuarios deben entrar datos que el programa usa luego en su procesamiento. Cuando se procesan datos insertados por un usuario, un requisito común es realizar operaciones aritméticas en los datos para generar resultados útiles. Las **operaciones aritméticas** incluyen sumar, restar, multiplicar y dividir datos numéricos.

Este capítulo describe los procesos necesarios para crear la aplicación *Beach Bike Rental* (una aplicación en la que el usuario especifica el número de bicicletas de playa a rentar y el programa entonces calcula el costo total de las bicicletas de playa).



El usuario entra el núm. 4 como el número de días a rentar la bicicleta de playa mientras está de vacaciones. El tiempo mínimo de alquiler es 1 día. Cuando el usuario hace clic en el botón Find Cost, el programa multiplica los 4 días por el costo de bicicleta por día ($9.95) y muestra el resultado del costo total. Cuando el usuario hace clic en el botón Clear, el número de días y el costo total del alquiler de la bicicleta se borran para que el siguiente usuario pueda ingresar su opción. Al hacer clic en el botón Exit, se cierra la ventana y finaliza el programa.

Para crear este tipo de aplicación, el desarrollador debe comprender cómo realizar los siguientes procesos, entre ellos:

1. Definir el objeto *TextBox* para la entrada de datos;
2. Definir una etiqueta (*label*) para almacenar los resultados de las operaciones aritméticas;
3. Convertir los datos del *TextBox* para que puedan usarse para operaciones aritméticas; y
4. Realizar operaciones aritméticas en los datos ingresados por un usuario.

En esta unidad, se presentan tres elementos nuevos:

- Objetos de cuadro de texto (***TextBox object***) → permiten a los usuarios ingresar datos en un programa
- Etiquetas destinadas a valores de propiedad de texto variable (***Labels***) → destinadas a valores de propiedad de texto variable
- Botones de aceptación (***Accept buttons***) → hacen que el programa lleve a cabo el procesamiento del controlador de eventos asociado con el botón que usuario hace clic o presiona la tecla *Enter*

Objeto *TextBox*

El objeto *TextBox* (cuadro de texto) permite a los usuarios ingresar datos en un programa. Recuerde nombrar el objeto *TextBox* con el prefijo **txt** antes del nombre y se escribe todo unido. Para colocar correctamente un objeto *TextBox* en el *Windows Form*, necesita saber el tamaño mínimo y máximo del cuadro de texto. El tamaño mínimo normalmente está determinado por la cantidad máxima de caracteres que el usuario ingresará en el cuadro de texto. Los programadores a menudo usan el dígito 8 para determinar el ancho del *TextBox* porque el número es más ancho que otros dígitos. Si el número es de dos dígitos se escribe 88, y se sigue añadiendo 8 para aumentar los dígitos y así determinar el ancho.

Alinear texto en el *TextBox*

Al entrar los datos numéricos en el cuadro de texto, éstos van a estar a la izquierda.

Pasos para alinear el texto:

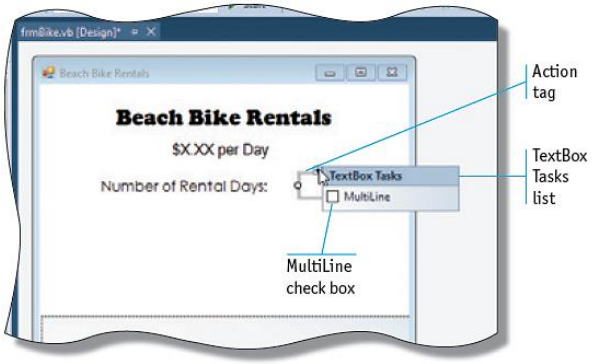
1. Selecciona el objeto *TextBox*.
2. En la ventana *Properties*, busca la propiedad *TextAlign* y haga clic en el nombre (columna izquierda).

3. Haga clic en la flecha en la columna derecha de la propiedad *TextAlign*.
4. En la lista desplegable, selecciona la alineación deseada.

MultiLine TextBox

Cuando el programa es ejecutado, el usuario puede entrar datos en el cuadro de texto. Los usuarios pueden entrar letras, números y otros caracteres. Si el usuario entra más de los caracteres que el cuadro de texto puede mostrar, los caracteres que ya se han ingresado se desplazan hacia la izquierda y ya no son visibles. El cuadro de texto no contiene una barra de desplazamiento, por lo tanto, el usuario debe mover el cursor a la izquierda o derecha con las flechas del teclado para ver los datos.

Podemos seleccionar una opción especial para que el cuadro de texto permita entrar múltiples líneas de texto. Un cuadro de texto *MultiLine* permite al usuario ingresar varias líneas en el cuadro de texto. El objeto *TextBox* se ajusta verticalmente para mostrar las múltiples líneas.



Pasos para crear el Multiline TextBox:

1. Selecciona el objeto *TextBox*.
2. H aga clic en la etiqueta *Action*.
3. Selecciona la casilla de verificación *Multiline*.

Accept Button

Los usuarios de computadoras a menudo presionan la tecla *Enter* para ingresar datos en un objeto *TextBox* y hacer que ocurra el procesamiento. Por ejemplo, en el programa de ejemplo para esta unidad, los usuarios pueden preferir escribir el número de días de alquiler y presionar la tecla *Enter* en lugar de escribir el número y hacer clic en el botón *Find Cost*. Al asignar un botón *Accept* (Aceptar) en la interfaz de usuario, significa que el programa llevará a cabo el procesamiento de eventos asociado con el botón si el usuario hace clic o presiona la tecla *Enter*.

Puede asignar un botón para que sea un botón *Accept* en la interfaz de usuario, lo que significa que el programa llevará a cabo el procesamiento del controlador de eventos asociado con el botón si el usuario hace clic en él o presiona *Enter*.

Cancel Button

Cuando el usuario presiona la tecla *ESC*, el procesamiento del controlador de eventos se ejecutará para el botón identificado como el botón *Cancel*. En el programa de ejemplo, el botón *Cancel* se utilizará para borrar el texto en el cuadro de texto (número de días), el costo total de las bicicletas alquiladas y para colocar el punto de inserción en el cuadro de texto. Por lo tanto, realiza la misma actividad que ocurre cuando el usuario hace clic en el botón *Clear*.

4.2 - Introducción a la Entrada de Datos y Tipos de Datos

Como ha visto, el usuario puede insertar datos en el programa utilizando el objeto *TextBox*. Cuando el usuario ingresa datos, se convierten en el valor almacenado en la propiedad del texto. Por ejemplo, si el usuario inserta 7 como la cantidad de días, la propiedad del texto tendrá el valor 7.

Tipos de datos

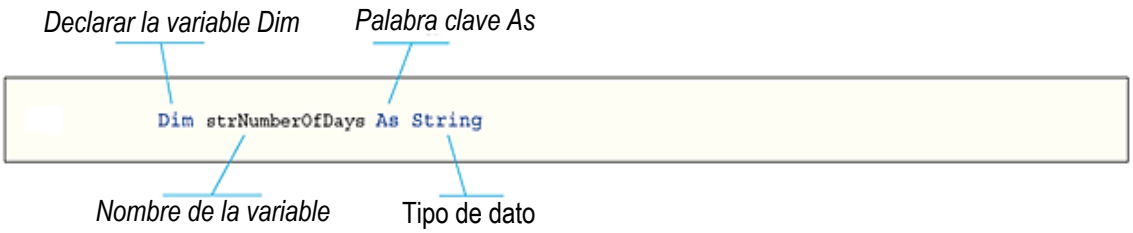
Datos de Cadena (*String data type*)

Cuando los datos se almacenan en la memoria RAM, se almacenan como un tipo de datos particular. Cada tipo de datos permite que los datos se utilicen de manera específica. Por ejemplo, para agregar dos valores juntos, los valores deben almacenarse en uno de los tipos de datos numéricos. El tipo de datos ***String*** se usa para valores que el usuario ingresa en un objeto *TextBox*. Un tipo de datos *String* puede almacenar cualquier carácter (letras, números y símbolos) disponible en la computadora.

Una **variable** es una ubicación con nombre en la memoria RAM donde se almacenan los datos. Una **variable *String*** (variable de cadena) es una ubicación con nombre en la memoria RAM que puede almacenar un valor de cadena. Por lo tanto, el nombre de una persona, una cantidad en dólares, un número de teléfono o el número de bicicletas alquiladas se pueden almacenar en una variable *String*. El programador define una variable durante la codificación del programa.

Declaración de una variable

Para declarar una variable, se requiere la palabra clave ***Dim***. Esta palabra clave significa dimensión variable. Indica al compilador de *Visual Basic* que las entradas después de *Dim* están definiendo una variable. Para nombrar a una variable *String* se comienza con el prefijo **str** seguido de un nombre descriptivo.



Pasos para declarar una variable:

1. Haga doble clic al objeto a declarar e ir a la ventana de códigos.
2. Escriba la palabra *Dim*.
3. Presiona la barra espaciadora y escriba el nombre de la variable. Cada variable debe tener un nombre para que pueda ser diferenciada en otras declaraciones dentro del programa.
4. Presiona la barra espaciadora y escriba la palabra clave ***As***. Si no está incluido, se producirá un error de compilación.
5. Presiona la barra espaciadora y escriba el tipo de datos de la variable.

Como resultado de la instrucción, cuando se compila el programa, el compilador de *Visual Basic* asignará un área en la memoria RAM que está reservado para almacenar el valor en la cadena.

El formato general para definir una variable es el siguiente:

General Format: Define a Variable

```
Dim VariableName As DataType
```

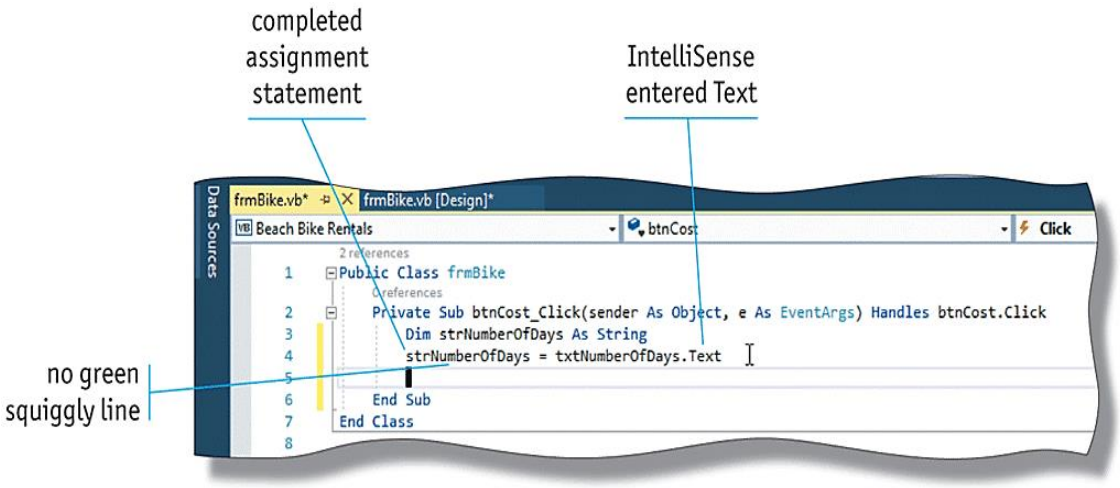
EXAMPLE	*RESULT*
`Dim strNumberOfDays`	`String variable`
`Dim intNumberOfDays`	`Integer variable`
`decTotalCost As Decimal`	`Decimal variable`

Asignar declaraciones

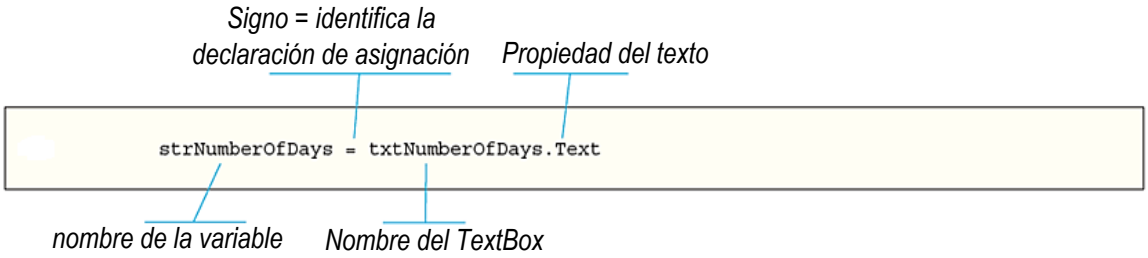
Cuando una variable se define, la variable no contiene ningún dato. Un método utilizado para colocar datos en la variable es una **declaración de asignación** (*assignment statements*). La declaración de asignación copiará los datos del texto de propiedad del objeto *TextBox*.

Pasos para realizar la declaración de asignación:

1. Haga doble clic al objeto para la declaración e ir a la ventana de códigos.
2. Para comenzar la declaración de asignación, escriba *strn*. *IntelliSense* muestra la lista con nombres de las variables que comienza con las letras strn.
3. Selecciona el nombre de la variable deseada (si no está seleccionada) y presiona la barra espaciadora.
4. Presiona la barra espaciadora, presione la tecla igual (=) y luego presione la barra espaciadora.
5. Escriba *txt* para mostrar la lista *IntelliSense* de objetos *Form*.
6. Escriba *n* para identificar el objeto *TextBox* en la lista *IntelliSense*.
7. Presiona la tecla del punto (.) y luego, si es necesario, escriba *te* para resaltar la entrada de Texto en la lista IntelliSense.
8. Presiona la tecla *Enter*.



Declaración de asignación



El nombre de variable a la izquierda de la instrucción de asignación identifica la variable a la que se asignará un valor copiado. El signo igual indica al compilador de *Visual Basic* que la instrucción es una instrucción de asignación. Se requiere el signo igual. El valor a la derecha del signo igual se copiará a la variable a la izquierda del signo igual.

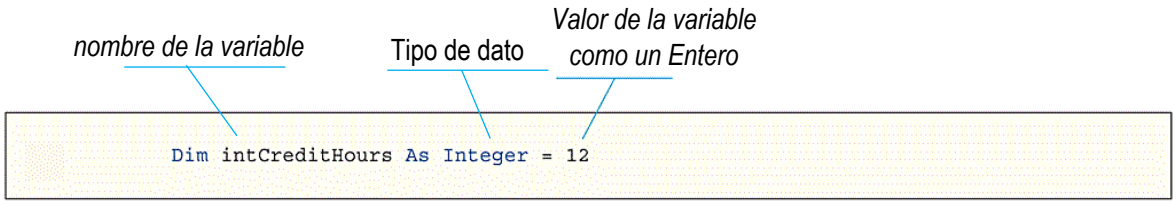
Tipo de datos numéricos

Como recordará, el tipo de datos ***String*** puede contener cualquier carácter que pueda ingresarse o almacenarse en una computadora. Sin embargo, los tipos de datos de cadena no se pueden usar en operaciones aritméticas. Por consiguiente, tenemos que identificar esos datos como tipos de datos numéricos. Para multiplicar dos valores, por ejemplo, los valores deben almacenarse en uno de los tipos de datos numéricos. *Visual Basic* permite una variedad de tipos de datos numéricos dependiendo de la necesidad de la aplicación. Cada tipo de datos numéricos requiere una cantidad diferente de memoria RAM para almacenar el valor numérico, y cada tipo de datos puede contener una forma diferente de datos numéricos y un rango máximo de valores diferentes. En la tabla se enumera tres tipos de datos numéricos utilizados con frecuencia.

Tipo de dato	Valor	Asignación en la memoria
Entero (*Integer*)	48	4 bytes
Decimal	3.15419	16 bytes
Doble (*Double*)	5.3452307	8 bytes

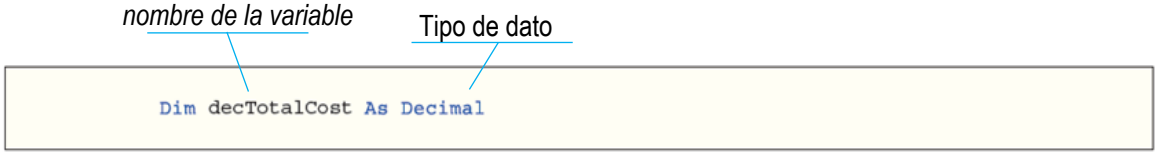
- **Datos Enteros (*Integer data type*)**

  Un **tipo de datos Enteros** contiene un número entero no decimal en *Visual Basic*. Puede almacenar un valor mayor o menor que 2 mil millones. Ejemplos de tipos de datos enteros incluyen la cantidad de bicicletas que se alquilan, la cantidad de horas de crédito que está tomando en un semestre y la cantidad de puntos que anotó su equipo de baloncesto favorito. Cada uno de estos ejemplos es un número entero. Normalmente, un tipo de datos entero se almacena en una variable entero. Una **variable *Integer*** (variable de entero) identifica una ubicación en la memoria RAM donde se almacena un valor entero. Por ejemplo, para definir una variable *Integer* con la función de almacenar el número de horas de crédito que está tomando y colocar el valor 12 en esa variable, debe escribir la declaración *Dim*. Para nombrar a una variable *Integer* se comienza con el prefijo **int** seguido de un nombre descriptivo.



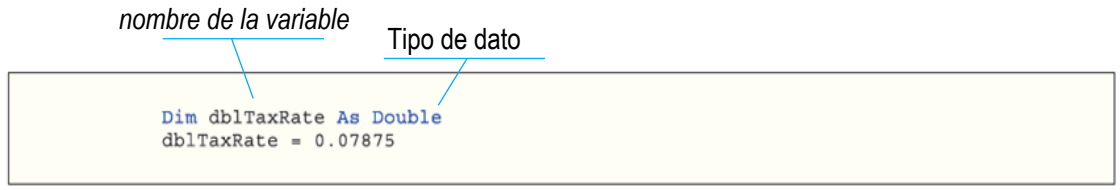
- **Datos Decimales (*Decimal data type*)**

  Un **tipo de datos Decimales** representa con precisión números decimales grandes o muy precisos. Es ideal para su uso en los campos contables y científicos para garantizar que los números mantengan su precisión y no estén sujetos a errores de redondeo. El tipo de datos *Decimal* puede tener una precisión de 28 dígitos significativos. Los dígitos significativos son aquellos que contribuyen a la precisión de un número. A menudo, los tipos de datos decimales se utilizan para almacenar cantidades en dólares. Para nombrar a una variable *Decimal* se comienza con el prefijo **dec** seguido de un nombre descriptivo.



- **Datos Dobles (D*ouble data type*)**

  Un **tipo de datos Dobles** puede representar números positivos enormes y números negativos muy pequeños que pueden incluir valores a la derecha del punto decimal. A veces, representa números de punto flotante, lo que significa que el punto decimal puede estar en cualquier lugar dentro del número. Para nombrar a una variable *Double* se comienza con el prefijo **dbl** seguido de un nombre descriptivo.




```
Dim dblTaxRate As Double
dblTaxRate = 0.07875
```

En la imagen anterior, se declara la variable dblTaxRate como tipo de datos *Double* y luego la instrucción de asignación se coloca el valor 0.07875 en la ubicación de memoria identificada por el nombre de la variable.

Otros tipos de datos

Visual Basic acepta otros tipos de datos que se utilizan para situaciones más especializadas. Dos tipos de datos ampliamente utilizados son ***Char*** y ***Boolean***. El tipo de datos *Char* representa una solo clic de tecla, como una letra del alfabeto, un signo de puntuación o un símbolo. Para nombrar a una variable *Char* se comienza con el prefijo **chr** seguido de un nombre descriptivo. Cuando asigna un valor a una variable *Char*, debe colocar comillas alrededor del valor.

En la imagen anterior, el valor A en la declaración de asignación tiene comillas alrededor. Además, *Visual Studio* muestra la letra y las comillas en texto rojo, lo que indica que no son palabras clave de *Visual Basic*, nombres de variables o nombres de objetos. De hecho, el valor se llama literal.

Una variable de datos *Boolean* (booleanos) puede contener un valor que *Visual Basic* interpreta como verdadero o falso. Si una variable en su programa debe representar si una condición es verdadera o no, entonces la variable debe ser booleana. Para nombrar a una variable *Boolean* se comienza con el prefijo **bln** seguido de un nombre descriptivo.

```
Dim blnFullTimeStudent As Boolean
blnFullTimeStudent = True
```

Constantes

Una variable identifica una ubicación en la memoria donde se puede almacenar un valor. Por su naturaleza, el valor en una variable se puede cambiar mediante declaraciones dentro del programa. Por ejemplo, en el programa de ejemplo de esta unidad, un usuario puede solicitar 4 alquileres de bicicletas y otro usuario puede solicitar 12 alquileres de bicicletas para un grupo grande. El valor en la variable *strNumberOfDays* puede cambiar según las necesidades del usuario. En algunos casos, sin embargo, es posible que no desee que se cambie el valor. Por ejemplo, el costo de alquilar una bicicleta en el programa de ejemplo es de $9.95 por día. Este valor no cambiará, independientemente de cuántas bicicletas se hayan alquilado. Cuando un valor seguirá siendo el mismo durante la ejecución del programa, debe asignar un nombre significativo a un valor. Una **constante** contiene un valor permanente durante la ejecución del programa. No puede ser modificado por ninguna declaración dentro del programa.

```
Const _cdecPricePerDay As Decimal = 9.95D
```

Las siguientes reglas se aplican a una constante:

1. La declaración de una constante comienza con las letras **Const**, **no** con las letras Dim.
2. Debe asignar el valor que debe contener la constante en la misma línea que su definición.
3. No puede intentar cambiar el valor de la constante en ninguna parte del programa. Si lo hace, generará un error de compilación.

4. La letra **c** a menudo se coloca antes del nombre constante para identificarlo en todo el programa como una variable que no se puede cambiar.
5. Además de la letra c, los nombres constantes se forman usando las mismas reglas y técnicas que otros nombres de variables.

El programa se vuelve más fácil de leer porque el valor se identifica por el nombre. Si descubre que se debe cambiar un valor en el código, es mucho más fácil y más confiable cambiar el valor una vez en la constante en lugar de cambiar cada aparición del valor en un literal.

Referencia a una variable

Cuando se declara una variable, se subraya con una línea ondulada verde hasta que se haga referencia en una declaración. Esta característica de *Visual Basic* está diseñada para garantizar que no declare una variable y luego se olvide de usarla. También ayuda a garantizar que no desperdicie memoria al declarar una variable innecesaria. Al usar una variable en un programa, es obligatorio que defina la variable antes de usar el nombre de la variable en una declaración.

Alcance de las variables

Cuando se declara una variable en *Visual Basic*, no solo declara el tipo de datos de la variable, sino que también define implícitamente su alcance. El **alcance de una variable** especifica dónde se puede hacer referencia a la variable en una instrucción de *Visual Basic* dentro del programa. Por ejemplo, en el programa de ejemplo de este unidad, puede declarar una variable en el controlador de eventos de clic para el botón *Find Cost.* Puede declarar otra variable en el controlador de eventos de clic para el botón *Clear*. El alcance determina dónde se puede hacer referencia y utilizar cada una de estas variables en el programa *Visual Basic*. La regla es: **una variable solo puede ser referenciada dentro de la región del programa donde está definida.** Una **región** en los programas que ha visto hasta ahora es el código entre la instrucción **Sub** y la instrucción **End Sub** en los controladores de eventos (en la ventana de códigos). El código entre la instrucción Sub y la instrucción End Sub es un procedimiento. Por lo tanto, si declara una variable dentro del controlador de eventos de clic para el botón *Find Cos*t, no se puede hacer referencia a esa variable en el controlador de eventos de clic para el botón *Clear*, y viceversa.

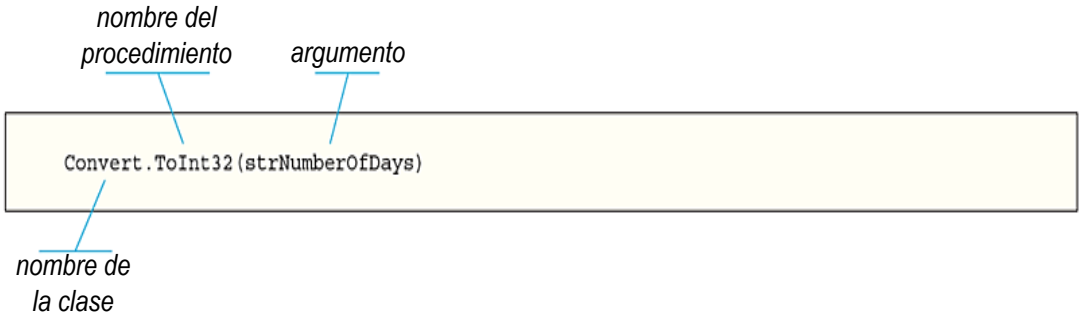
Una variable a la que solo se puede hacer referencia dentro de la región del programa donde se define se denomina **variable local**. Esta variable se define en una región del programa y no se puede cambiar mediante una declaración en otra región del programa. Además, cuando una variable se define en un procedimiento y el procedimiento finaliza, los valores de las variables locales dentro del procedimiento son destruidos. Por lo tanto, las variables locales tienen una cierta duración en el programa. Solo están vivos desde el momento en que comienza el procedimiento hasta que termina. Si el procedimiento se ejecuta nuevamente, el valor que la variable una vez contenía ya no está presente. Una **ejecución del procedimiento** es la vida útil de una variable.

Por lo tanto, si un usuario hace clic en el botón *Find Cost*, los valores en las variables son válidos hasta que se complete el evento de clic. Cuando el usuario vuelve a hacer clic en el botón *Find Cost*, todos los valores del primer clic desaparecen. También puede definir variables que se pueden usar en varias regiones de un programa de *Visual Basic*. Estas variables se llaman **variables globales**. En la mayoría de los programas, las variables locales deben usarse porque causan menos errores que las variables globales.

Convertir datos variables

Las variables utilizadas en las declaraciones aritméticas en un programa de *Visual Basic* deben ser numéricas. Las variables de cadena (*string*) no se pueden usar en una declaración aritmética. Si intenta usarlos, creará un error de compilación. Un usuario a menudo ingresa datos en un *TextBox*. Los datos en la propiedad de texto de un objeto *TextBox* se tratan como datos de cadena. Debido a que los datos de cadena no pueden usarse en una declaración aritmética, los datos de cadena ingresados por un usuario deben convertirse a datos numéricos antes de que puedan usarse en una declaración aritmética. Por ejemplo, en el programa de ejemplo de esta unidad, el usuario ingresa el número de días para alquilar bicicletas. Antes de que este número pueda usarse en una declaración aritmética para determinar el costo total del alquiler de bicicletas, el valor debe convertirse a un tipo de datos entero (*integer*).

Visual Basic incluye varios procedimientos que le permiten convertir un tipo de datos en otro tipo de datos. Recordará que un **procedimiento** es un conjunto de código preescrito que puede ser llamado por una instrucción en el programa *Visual Basic*. Cuando se llama al procedimiento, realiza una tarea particular. En este caso, la tarea es convertir el valor de cadena que ingresó el usuario en un tipo de datos *Entero* que se puede usar en una operación aritmética. Un procedimiento que convierte un tipo de datos de cadena (*string*) en un tipo de datos entero (*integer*) se llama **ToInt32**. El número 32 en el nombre del procedimiento identifica que la representación del número entero requerirá 32 bits o 4 bytes, que es la cantidad de memoria requerida para el tipo de datos entero.



4.3 - Operaciones Aritméticas

La capacidad de realizar operaciones aritméticas en datos numéricos es fundamental para los programas de computadora. Muchos programas requieren operaciones aritméticas para sumar, restar, multiplicar y dividir datos numéricos. Por ejemplo, en el programa *Beach Bike Rentals* en esta unidad, el costo por día debe multiplicarse por la cantidad de días de alquiler para calcular el costo total. Se utiliza una declaración de asignación para realizar la operación aritmética.

Operadores aritméticos

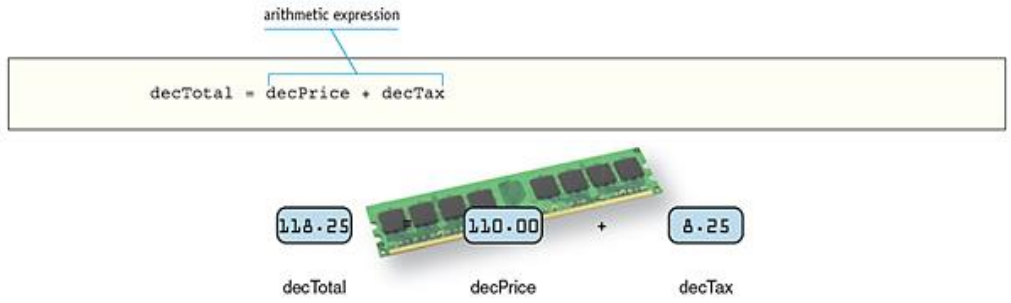
Los operadores aritméticos indican la operación aritmética que se llevará a cabo en un compilador. *Visual Basic* utiliza los siguientes operadores:

Operador aritmético	Uso	Ejemplo
+	Suma	decTotal = decPrice+decTax
-	Resta	decCost = decRegularprice - decDiscount
*	Multiplicación	decTax = decItemprice * decTaxRate
/	División	decClassAverage = decTotalScores / intNumberOfStudents

Operador aritmético	Uso	Ejemplo
^	Exponentes	intSquareArea = intSquareSide^2
\	División de *Enteros*	intResult = 13\5
Mod	Módulo artimético	intRemainder = 13 Mod 5

Adición

El **operador aritmético de suma (+)** agrega los valores numéricos inmediatamente a la izquierda y derecha del operador y reemplaza la expresión aritmética en la instrucción de asignación.



Una expresión aritmética que utiliza el operador de suma puede contener más de dos valores numéricos para agregar. Además de las variables, las expresiones aritméticas pueden contener literales.

```
decTotalPay = decRegularPay + decOvertimePay + decBonusPay
```

```
decTicketCost = decInternetTicketCost + 10.25
```

Resta

Para restar un valor de otro en una declaración de asignación, *Visual Basic* 2017 usa el **operador aritmético de resta (-)**.

```
decNetProfit = decRevenue - decCosts
```

Multiplicación

La multiplicación se logra mediante el uso de una declaración de asignación y el **operador de multiplicación (*)**.

```
intLandPlotArea = intLandPlotLength * intLandPlotWidth
```

División

Visual Basic 2017 proporciona tres operadores aritméticos para la división y los cálculos relacionados: la **barra diagonal** (/), la **barra diagonal inversa** (\) y el operador **MOD**.

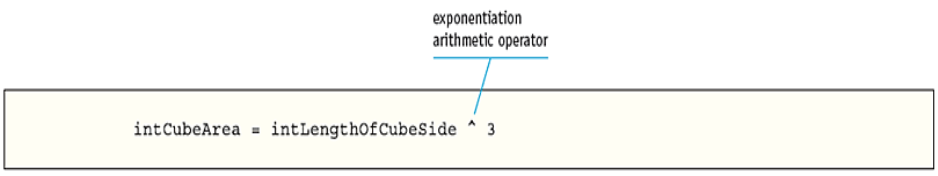
```
decAverageTestScore = decTestScores / 3
```

La **barra invertida** (\) se utiliza para la división de enteros. Con la división de enteros, el cociente por la operación de división es un entero. Si la operación de división produce un cociente con un residuo, el residuo se descarta o se redondea automáticamente. El operador **MOD** divide el número a la izquierda del operador por el número a la derecha del operador y devuelve un valor entero que es el resto de la operación de división. La división de enteros y el operador MOD a menudo se usan juntos, como se muestra en a continuación.

```
intHours = intTotalNumberOfMinutes \ 60
intMinutes = intTotalNumberOfMinutes Mod 60
```

Exponentes

Los exponentes significan elevar un número a un poder. Se realiza en *Visual Basic* 2017 utilizando el operador aritmético de exponenciación (^).



Jerarquía de operaciones

Cuando se incluyen varias operaciones en una sola declaración de asignación, la secuencia para realizar los cálculos se determina mediante las siguientes reglas:

1. El exponente (^) se realiza primero.
2. La multiplicación (*) y la división (/) se realizan a continuación.
3. La división *Entera* (\) es la siguiente.
4. Entonces se produce MOD.
5. La suma (+) y la resta (-) se realizan en último lugar.

Dentro de cada uno de los cinco pasos anteriores, los cálculos se realizan de izquierda a derecha.

4.4 - Borrar el Formulario

Anteriormente en esta unidad, aprendió que cuando el usuario hace clic en el botón *Clear* en el programa *Beach Bike Rentals*, el controlador de eventos para el botón *Clear* debe borrar los resultados de la ventana y permitir que el usuario escriba el siguiente número de días de alquiler. Para borrar los resultados, el controlador de eventos del botón *Clear* debe completar las siguientes tareas:

1. Borrar la propiedad *Text* del objecto *TextBox*.
2. Borrar la propiedad *Text* del objeto *Label* que muestra el costo total.
3. Establecer el foco en el objeto *TextBox*, lo que significa colocar el punto de inserción en el cuadro de texto para comenzar nuevamente.

Procedimiento Clear

El procedimiento *Clear* borra cualquier dato en la propiedad *Text* de un objeto *TextBox*. La declaración es:

General Format: Clear Procedure

```
txtTextboxName.Clear()
```

EXAMPLE:

```
txtNumberOfDays.Clear()
```

Borrar la propiedad de texto en una etiqueta (*Label*)

El procedimiento *Clear* no se puede usar con un objeto *Label*. Debe escribir una declaración de asignación que asigne una cadena de longitud nula a la propiedad *Text*.

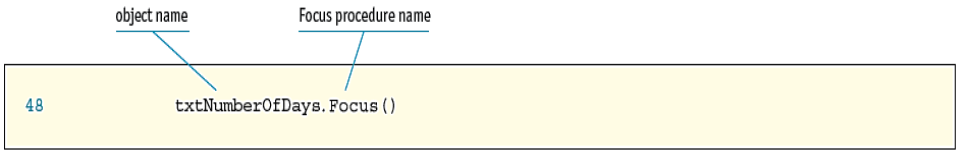
Para asignar una longitud nula, se utiliza la siguiente declaración:

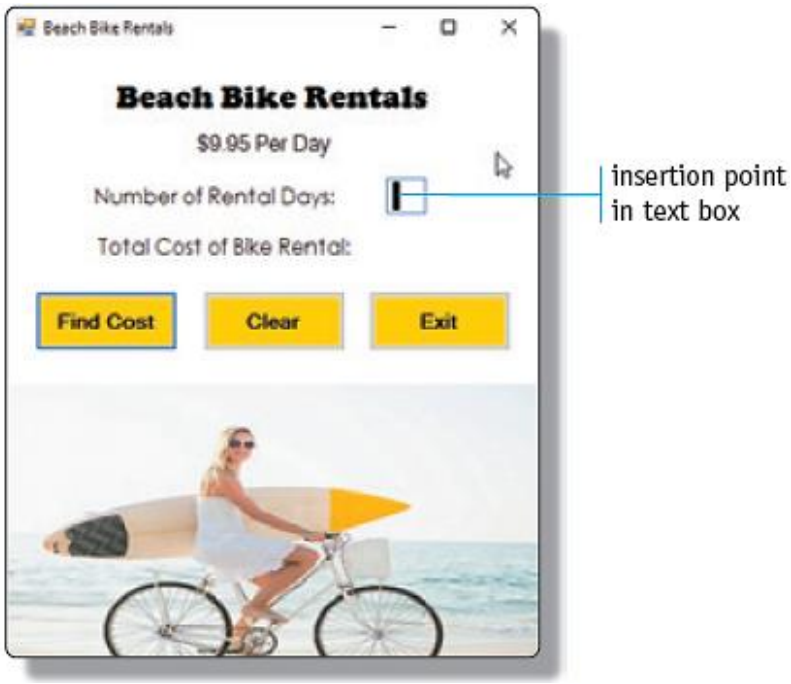
lblLabelName.Text = “”

```
lblTotalCost.Text = ""
```

Establecer el foco

Cuando el foco está en un objeto *TextBox*, el punto de inserción se encuentra en el cuadro de texto. El programador puede usar el procedimiento Focus (enfoque) para colocar el foco en un cuadro de texto automáticamente sin requerir que el usuario haga clic primero, lo que facilita que el usuario ingrese datos en el cuadro de texto. Se utiliza la siguiente declaración: **txtTextBoxName.Focus()**





Actividad de Avalúo 6: Unidad 4

Realizar las contestaciones en la HOJA DE CONTESTACIONES (final del Módulo).

Ejercicio 1: Completa la oración con la contestación correcta..

86. El botón _____ se utiliza para borrar el texto en el cuadro de texto.
87. Al nombrar la variable para los datos enteros se escribe el prefijo _____ seguido por el nombre descriptivo.
88. Al declarar una variable, la palabra clave _____ se requiere al comienzo de la declaración.
89. Una variable _____ debe usarse para valores que consistirán en un solo carácter.
90. El código entre la instrucción _____ y _____ se le conoce como una región del programa.
91. La opción especial _____ permite entrar varias líneas de texto en el TextBox.
92. El procedimiento _____ convierte un tipo de datos de _____ en un tipo de _____.
93. El procedimiento _____ borra datos en la propiedad Text del objeto TextBox.
94. Se utiliza un prefijo de _____ para las variables definidas como un tipo de datos de cadena.
95. Un tipo de datos _____ a veces, representa números de punto flotante.
96. El _____ permite a los usuarios ingresar datos en un programa.
97. Se utiliza el procedimiento _____ para colocar el foco en un cuadro de texto automáticamente.
98. Al nombrar la variable para los datos decimales se escribe el prefijo _____ seguido por el nombre descriptivo.
99. Un tipo de datos _____ puede almacenar cualquier carácter disponible en la computadora.

100. La _____ es el método utilizado para colocar datos en la variable.
101. Un tipo de datos _____ contiene un número entero no decimal.
102. Un valor permanente durante la ejecución del programa es un (a) _____ y al escribir la declaración se comienza con _____.
103. El prefijo ____ se usa para denotar que una variable se ha definido como un tipo de datos booleanos.
104. Al declarar una variable y no escribir la palabra _____ se producirá un error de compilación.
105. Una variable _____ solo se puede hacer referencia de la variable dentro de la región del programa donde se define.

Ejercicio 2: Escribe declaración para:

106. Declaración para que el punto de inserción esté en un cuadro de texto automáticamente.
107. Declaración de la variable *Miles* con un valor entero de 25.
108. Declaración para asignar una longitud nula.
109. Declararación de la constante Price per Unit con un valor de $12.75.
110. Declararación del procedimiento Clear.
111. Declaración con el formato general para definir una variable.

CLAVES DE RESPUESTA DE ACTIVIDADES DE AVALÚO

Clave Actividad de Avalúo 1: Unidad 1

Selección Múltiple

1. **c**	6. **c**	11. **d**	16. **a**
2. **c**	7. **b**	12. **b**	17. **b**
3. **a**	8. **d**	13. **d**	18. **a**
4. **b**	9. **c**	14. **b**	19. **c**
5. **d**	10. **a**	15. **a**	20. **a**

Clave Actividad de Avalúo 2: Unidad 2 ➡ 2.1

Ejercicio 1: Define

21. Proyecto de Escritorio Clásico de Window - **Es un programa que incluye una interfaz de usuario cuyas ventanas se crean utiliznado el sistema operativo Windows.**
22. Objeto *Windows Form* - **Es la ventana que se utilizará para compliar el programa y luego mostrarlo en una pantalla cuando se ejecute el programa.**
23. Propiedades - **Pueden ajustar varias de las características hacerca de un objeto, ejemplo su color, tamaño, nombre y posición en la pantalla**.
24. Propiedad de texto – **Contienen el valor, o texto, el cual se mostrará en el Form dentro de este objeto**.

Ejercicio 2: Selección Múltiple

25. **a**	27. **b**	29. **a**	31. **d**
26. **b**	28. **a**	30. **b**	32. **c**

Clave Actividad de Avalúo 3: Unidad 2 ➡ 2.2, 2.3

Ejercicio 1: Contesta

33. **width, height, both**	39. **left, right, c*Enter*, top, bottom, middle**
34. **pic**	40. **vertical, pushpin (tachuela)**
35. **Toolbox**	41. **btn**
36. **Text**	42. **Name**
37. **lbl**	43. **Font**
38. **Font**	44. **horizontally, vertically**

Ejercicio 2: Identifica

45. **Diseñar los objetos de procesamiento del programa**
46. **Probar el programa**
47. **Documentar el programa o el sistema**
48. **Diseñar la interfaz de usuario**
49. **Mantener el programa o sistema**
50. **Reunir y analizar los requisitos del programa**
51. **Codificar el programa**

Clave Actividad de Avalúo 4: Unidad 3 ➡ 3.1

Ejercicio 1: Cierto o Falso

52. **C**
53. **F, True**
54. **F, ForeColor**
55. **F, C*Enter* Image**
56. **C**
57. **C**
58. **C**
59. **F, Normal**
60. **C**
61. **F, True**
62. **C**
63. **F, SizeMode**

Clave Actividad de Avalúo 5: Unidad 3 ➡ 3.2

Ejercicio 1: Selección Múltiple

64. **d**
65. **c**
66. **c**
67. **d**
68. **b**
69. **a**
70. **a**
71. **c**
72. **b**
73. **d**

Ejercicio 2: Analiza

74. **picParkBison.Visible = True**
75. **Enabled, False**
76. **BackColor**
77. **Text**
78. **btnExit.Enabled = True**

79. **‘ Nombre del Programa: Yellowstone National Park**
80. **‘ Desarrollador: Nombre del Estudiante**
81. **‘ Fecha: 12 de noviembre de -----**
82. **‘ Propósito: En esta aplicación el usuario puede seleccionar entre dos parques.**
    ‘
83. **btnPark1.Enabled = False**
84. **btnPark2.Enabled = False**
85. **btnExit.Enabled = True**

Clave Actividad de Avalúo 6: Unidad 4

Ejercicio 1: Llena blanco

86. **Cancel**
87. **int**
88. **Dim**
89. **Char**
90. **Sub**, **End Sub**
91. **MultiLine TextBox**
92. **ToInt32**, **string** (cadenas), **integer** (*Enter*os)
93. **Clear**
94. **str**
95. **Double** (Dobles)
96. **objeto TextBox** (cuadro de texto)
97. **Focus**
98. **dec**
99. **string** (cadena)
100. **declaración de asignación**
101. **Integer** (*Enter*os)
102. **constante**, **Const**
103. **bln**
104. **As**
105. **local**

Ejercicio 2: Declaraciones:

106. **txtTextBoxName.Focus()**
107. **Dim intMiles As Integer = 25**
108. **lblLabelName.Text = “”**
109. **Const _cPricePerUnit As Decimal = 12.75D**
110. **txtTextBoxName.Clear**
111. **Dim VariableName As Data Type**

REFERENCIA

Hoisington, C. (2018). *Microsoft Visual Basic 2017. Comprehensive: For Windows, Web, and Database Applications*. Cengage Learning. ISBN: 9781337102117

(2013). *National Standards for Business Education: What America's Studentes Should Know and Be Able to Do in Business*. NBEA. ISBN 0-933964-77-3

Estimada familia:

El Departamento de Educación de Puerto Rico (DEPR) tiene como prioridad el garantizar que a sus hijos se les provea una educación pública, gratuita y apropiada. Para lograr este cometido, es imperativo tener presente que los seres humanos son diversos. Por eso, al educar es necesario reconocer las habilidades de cada individuo y buscar estrategias para minimizar todas aquellas barreras que pudieran limitar el acceso a su educación.

La otorgación de acomodos razonables es una de las estrategias que se utilizan para minimizar las necesidades que pudiera presentar un estudiante. Estos permiten adaptar la forma en que se presenta el material, la forma en que el estudiante responde, la adaptación del ambiente y lugar de estudio y el tiempo e itinerario que se utiliza. Su función principal es proveerle al estudiante acceso equitativo durante la enseñanza y la evaluación. Estos tienen la intención de reducir los efectos de la discapacidad, excepcionalidad o limitación del idioma y no, de reducir las expectativas para el aprendizaje. Durante el proceso de enseñanza y aprendizaje, se debe tener altas expectativas con nuestros niños y jóvenes.

Esta guía tiene el objetivo de apoyar a las familias en la selección y administración de los acomodos razonables durante el proceso de enseñanza y evaluación para los estudiantes que utilizarán este módulo didáctico. Los acomodos razonables le permiten a su hijo realizar la tarea y la evaluación, no de una forma más fácil, sino de una forma que sea posible de realizar, según las capacidades que muestre. El ofrecimiento de acomodos razonables está atado a la forma en que su hijo aprende. Los estudios en neurociencia establecen que los seres humanos aprenden de forma visual, de forma auditiva o de forma kinestésica o multisensorial, y aunque puede inclinarse por algún estilo, la mayoría utilizan los tres.

Por ello, a continuación, se presentan algunos ejemplos de acomodos razonables que podrían utilizar con su hijo mientras trabaja este módulo didáctico en el hogar. Es importante que como madre, padre o persona encargada en dirigir al estudiante en esta tarea los tenga presente y pueda documentar cuales se utilizaron. Si necesita más información, puede hacer referencia a la ***Guía para la provisión de acomodos razonables*** (2018) disponible por medio de la página www.de.pr.gov, en educación especial, bajo Manuales y Reglamentos.

GUÍA DE ACOMODOS RAZONABLES PARA LOS ESTUDIANTES QUE TRABAJARÁN BAJO MÓDULOS DIDÁCTICOS

Acomodos de presentación	Acomodos en la forma de responder	Acomodos de ambiente y lugar	Acomodos de tiempo e itinerario
Cambian la manera en que se presenta la información al estudiante. Esto le permite tener acceso a la información de diferentes maneras. El material puede ser presentado de forma auditiva, táctil, visual o multisensorial.	Cambian la manera en que el estudiante responde o demuestra su conocimiento. Permite a los estudiantes presentar las contestaciones de las tareas de diferentes maneras. Por ejemplo, de forma verbal, por medio de manipulativos, entre otros.	Cambia el lugar, el entorno o el ambiente donde el estudiante completará el módulo didáctico. Los acomodos de ambiente y lugar requieren de organizar el espacio donde el estudiante trabajará.	Cambian la cantidad de tiempo permitido para completar una evaluación o asignación; cambia la manera, orden u hora en que se organiza el tiempo, las materias o las tareas.
Aprendiz visual: ▪ Usar letra agrandada o equipos para agrandar como lupas, televisores y computadoras ▪ Uso de láminas, videos pictogramas. ▪ Utilizar claves visuales tales como uso de colores en las instrucciones, resaltadores (highlighters), subrayar palabras importantes. ▪ Demostrar lo que se espera que realice el estudiante y utilizar modelos o demostraciones. ▪ Hablar con claridad, pausado ▪ Identificar compañeros que puedan servir de apoyo para el estudiante ▪ Añadir al material información complementaria **Aprendiz auditivo**: ▪ Leerle el material o utilizar aplicaciones que convierten el	**Aprendiz visual:** ▪ Utilizar la computadora para que pueda escribir. ▪ Utilizar organizadores gráficos. ▪ Hacer dibujos que expliquen su contestación. ▪ Permitir el uso de láminas o dibujos para explicar sus contestaciones ▪ Permitir que el estudiante escriba lo que aprendió por medio de tarjetas, franjas, láminas, la computadora o un comunicador visual. ▪ Contestar en el folleto. **Aprendiz auditivo**: ▪ Grabar sus contestaciones ▪ Ofrecer sus contestaciones a un adulto que documentará por escrito lo mencionado.	**Aprendiz visual:** ▪ Ambiente silencioso, estructurado, sin muchos distractores. ▪ Lugar ventilado, con buena iluminación. ▪ Utilizar escritorio o mesa cerca del adulto para que lo dirija. **Aprendiz auditivo**: ▪ Ambiente donde pueda leer en voz alta o donde pueda escuchar el material sin interrumpir a otras personas. ▪ Lugar ventilado, con buena iluminación y donde se les permita el movimiento mientras repite en voz alta el material. **Aprendiz multisensorial:** ▪ Ambiente se le permita moverse, hablar, escuchar música mientras trabaja, cantar. ▪ Permitir que realice las actividades en	**Aprendiz visual y auditivo:** ▪ Preparar una agenda detalladas y con códigos de colores con lo que tienen que realizar. ▪ Reforzar el que termine las tareas asignadas en la agenda. ▪ Utilizar agendas de papel donde pueda marcar, escribir, colorear. ▪ Utilizar “post-it” para organizar su día. ▪ Comenzar con las clases más complejas y luego moverse a las sencillas. ▪ Brindar tiempo extendido para completar sus tareas. **Aprendiz multisensorial:** ▪ Asistir al estudiante a organizar su trabajo con agendas escritas o electrónicas. ▪ Establecer mecanismos para

Acomodos de presentación	Acomodos en la forma de responder	Acomodos de ambiente y lugar	Acomodos de tiempo e itinerario
texto en formato audible. ▪ Leer en voz alta las instrucciones. ▪ Permitir que el estudiante se grabe mientras lee el material. ▪ Audiolibros ▪ Repetición de instrucciones ▪ Pedirle al estudiante que explique en sus propias palabras lo que tiene que hacer ▪ Utilizar el material grabado ▪ Identificar compañeros que puedan servir de apoyo para el estudiante **Aprendiz multisensorial:** ▪ Presentar el material segmentado (en pedazos) ▪ Dividir la tarea en partes cortas ▪ Utilizar manipulativos ▪ Utilizar canciones ▪ Utilizar videos ▪ Presentar el material de forma activa, con materiales comunes. ▪ Permitirle al estudiante investigar sobre el tema que se trabajará ▪ Identificar compañeros que puedan servir de apoyo para el estudiante	▪ Hacer presentaciones orales. ▪ Hacer videos explicativos. ▪ Hacer exposiciones **Aprendiz multisensorial:** ▪ Señalar la contestación a una computadora o a una persona. ▪ Utilizar manipulativos para representar su contestación. ▪ Hacer presentaciones orales y escritas. ▪ Hacer dramas donde represente lo aprendido. ▪ Crear videos, canciones, carteles, infografías para explicar el material. ▪ Utilizar un comunicador electrónico o manual.	diferentes escenarios controlados por el adulto. Ejemplo el piso, la mesa del comedor y luego, un escritorio.	recordatorios que le sean efectivos. ▪ Utilizar las recompensas al terminar sus tareas asignadas en el tiempo establecido. ▪ Establecer horarios flexibles para completar las tareas. ▪ Proveer recesos entre tareas. ▪ Tener flexibilidad en cuando al mejor horario para completar las tareas. ▪ Comenzar con las tareas más fáciles y luego, pasar a las más complejas. ▪ Brindar tiempo extendido para completar sus tareas.

HOJA DE DOCUMENTAR LOS ACOMODOS RAZONABLES UTILIZADOS AL TRABAJAR EL MÓDULO DIDÁCTICO

Nombre del estudiante: _____ **Número de SIE:**_____
Materia del módulo: _____ **Grado:**_____

Estimada familia:

1. Utiliza la siguiente hoja para documentar los acomodos razonables que utiliza con tu hijo en el proceso de apoyo y seguimiento al estudio de este módulo. Favor de colocar una marca de cotejo [✓] en aquellos acomodos razonables que utilizó con su hijo para completar el módulo didáctico. Puede marcar todos los que aplique y añadir adicionales en la parte asignada para ello.

Acomodos de presentación	Acomodos de tiempo e itinerario
Aprendiz visual: ☐ Usar letra agrandada o equipos para agrandar como lupas, televisores y computadoras ☐ Uso de láminas, videos pictogramas. ☐ Utilizar claves visuales tales como uso de colores en las instrucciones, resaltadores (*highlighters*), subrayar palabras importantes. ☐ Demostrar lo que se espera que realice el estudiante y utilizar modelos o demostraciones. ☐ Hablar con claridad, pausado ☐ Identificar compañeros que puedan servir de apoyo para el estudiante ☐ Añadir al material información complementaria **Aprendiz auditivo**: ☐ Leerle el material o utilizar aplicaciones que convierten el texto en formato audible. ☐ Leer en voz alta las instrucciones. ☐ Permitir que el estudiante se grabe mientras lee el material. ☐ Audiolibros ☐ Repetición de instrucciones ☐ Pedirle al estudiante que explique en sus propias palabras lo que tiene que hacer ☐ Utilizar el material grabado ☐ Identificar compañeros que puedan servir de apoyo para el estudiante **Aprendiz multisensorial:** ☐ Presentar el material segmentado (en pedazos) ☐ Dividir la tarea en partes cortas ☐ Utilizar manipulativos ☐ Utilizar canciones	**Aprendiz visual:** ☐ Utilizar la computadora para que pueda escribir. ☐ Utilizar organizadores gráficos. ☐ Hacer dibujos que expliquen su contestación. ☐ Permitir el uso de láminas o dibujos para explicar sus contestaciones ☐ Permitir que el estudiante escriba lo que aprendió por medio de tarjetas, franjas, láminas, la computadora o un comunicador visual. ☐ Contestar en el folleto. **Aprendiz auditivo**: ☐ Grabar sus contestaciones ☐ Ofrecer sus contestaciones a un adulto que documentará por escrito lo mencionado. ☐ Hacer presentaciones orales. ☐ Hacer videos explicativos. ☐ Hacer exposiciones **Aprendiz multisensorial:** ☐ Señalar la contestación a una computadora o a una persona. ☐ Utilizar manipulativos para representar su contestación. ☐ Hacer presentaciones orales y escritas. ☐ Hacer dramas donde represente lo aprendido. ☐ Crear videos, canciones, carteles, infografías para explicar el material. ☐ Utilizar un comunicador electrónico o manual.

Acomodos de presentación	Acomodos de tiempo e itinerario
☐ Utilizar videos ☐ Presentar el material de forma activa, con materiales comunes. ☐ Permitirle al estudiante investigar sobre el tema que se trabajará ☐ Identificar compañeros que puedan servir de apoyo para el estudiante	

Acomodos de respuesta	Acomodos de ambiente y lugar
Aprendiz visual: ☐ Ambiente silencioso, estructurado, sin muchos distractores. ☐ Lugar ventilado, con buena iluminación. ☐ Utilizar escritorio o mesa cerca del adulto para que lo dirija. **Aprendiz auditivo**: ☐ Ambiente donde pueda leer en voz alta o donde pueda escuchar el material sin interrumpir a otras personas. ☐ Lugar ventilado, con buena iluminación y donde se les permita el movimiento mientras repite en voz alta el material. **Aprendiz multisensorial:** ☐ Ambiente se le permita moverse, hablar, escuchar música mientras trabaja, cantar. ☐ Permitir que realice las actividades en diferentes escenarios controlados por el adulto. Ejemplo el piso, la mesa del comedor y luego, un escritorio.	**Aprendiz visual y auditivo:** ☐ Preparar una agenda detalladas y con códigos de colores con lo que tienen que realizar. ☐ Reforzar el que termine las tareas asignadas en la agenda. ☐ Utilizar agendas de papel donde pueda marcar, escribir, colorear. ☐ Utilizar "post-it" para organizar su día. ☐ Comenzar con las clases más complejas y luego moverse a las sencillas. ☐ Brindar tiempo extendido para completar sus tareas. **Aprendiz multisensorial:** ☐ Asistir al estudiante a organizar su trabajo con agendas escritas o electrónicas. ☐ Establecer mecanismos para recordatorios que le sean efectivos. ☐ Utilizar las recompensas al terminar sus tareas asignadas en el tiempo establecido. ☐ Establecer horarios flexibles para completar las tareas. ☐ Proveer recesos entre tareas. ☐ Tener flexibilidad en cuando al mejor horario para completar las tareas. ☐ Comenzar con las tareas más fáciles y luego, pasar a las más complejas. ☐ Brindar tiempo extendido para completar sus tareas.

Otros:

_____
_____
_____
_____
_____
_____
_____
_____

2. Si tu hijo es un candidato o un participante de los servicios para estudiantes aprendices del español como segundo idioma e inmigrantes considera las siguientes sugerencias de enseñanza:

- Proporcionar un modelo o demostraciones de respuestas escritas u orales requeridas o esperadas.
- Comprobar si hay comprensión: use preguntas que requieran respuestas de una sola palabra, apoyos y gestos.
- Hablar con claridad, de manera pausada.
- Evitar el uso de las expresiones coloquiales, complejas.
- Asegurar que los estudiantes tengan todos los materiales necesarios.
- Leer las instrucciones oralmente.
- Corroborar que los estudiantes entiendan las instrucciones.
- Incorporar visuales: gestos, accesorios, gráficos organizadores y tablas.
- Sentarse cerca o junto al estudiante durante el tiempo de estudio.
- Seguir rutinas predecibles para crear un ambiente de seguridad y estabilidad para el aprendizaje.
- Permitir el aprendizaje por descubrimiento, pero estar disponible para ofrecer instrucciones directas sobre cómo completar una tarea.
- Utilizar los organizadores gráficos para la relación de ideas, conceptos y textos.
- Permitir el uso del diccionario regular o ilustrado.
- Crear un glosario pictórico.
- Simplificar las instrucciones.
- Ofrecer apoyo en la realización de trabajos de investigación.
- Ofrecer los pasos a seguir en el desarrollo de párrafos y ensayos.
- Proveer libros o lecturas con conceptos similares, pero en un nivel más sencillo.
- Proveer un lector.
- Proveer ejemplos.
- Agrupar problemas similares (todas las sumas juntas), utilizar dibujos, láminas, o gráficas para apoyar la explicación de los conceptos, reducir la complejidad lingüística del problema, leer y explicar el problema o teoría verbalmente o descomponerlo en pasos cortos.
- Proveer objetos para el aprendizaje (concretizar el vocabulario o conceptos).
- Reducir la longitud y permitir más tiempo para las tareas escritas.
- Leer al estudiante los textos que tiene dificultad para entender.
- Aceptar todos los intentos de producción de voz sin corrección de errores.
- Permitir que los estudiantes sustituyan dibujos, imágenes o diagramas, gráficos, gráficos para una asignación escrita.
- Esbozar el material de lectura para el estudiante en su nivel de lectura, enfatizando las ideas principales.
- Reducir el número de problemas en una página.
- Proporcionar objetos manipulativos para que el estudiante utilice cuando resuelva problemas de matemáticas.

3. Si tu hijo es un estudiante dotado, es decir, que obtuvo 130 o más de cociente intelectual (CI) en una prueba psicométrica, su educación debe ser dirigida y desafiante. Deberán considerar las siguientes recomendaciones:

- Conocer las capacidades especiales del estudiante, sus intereses y estilos de aprendizaje.
- Realizar actividades motivadoras que les exijan pensar a niveles más sofisticados y explorar nuevos temas.
- Adaptar el currículo y profundizar.
- Evitar las repeticiones y las rutinas.
- Realizar tareas de escritura para desarrollar empatía y sensibilidad.
- Utilizar la investigación como estrategia de enseñanza.
- Promover la producción de ideas creativas.
- Permitirle que aprenda a su ritmo.
- Proveer mayor tiempo para completar las tareas, cuando lo requiera.
- Cuidar la alineación entre su educación y sus necesidades académicas y socioemocionales.