TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN UTN-FRC

TUP

UTN

Facultad Regional Córdoba

TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN

PROGRAMACIÓN II

Unidad Temática 1: Colecciones de Datos Simples

Material de Estudio

1er Año - 2er Cuatrimestre

2019

V.0.1

Índice

1. Componentes Que Manejan Colecciones

Listbox y Combobox

Existen muchas ocasiones en donde el usuario del programa tiene que proporcionar datos que provienen de un conjunto finito y muy pequeño de posibles respuestas esto significa que cada vez que se ejecute el programa el usuario estará proporcionando las mismas respuestas.

Ejemplo de esta clase de datos, son por ejemplos Municipio en BC las posibles respuestas solo son (Tecate, Tijuana, Mexicali, Ensenada, Rosarito), otro ejemplo es Sexo (Hombre, Mujer), etc.

Para situaciones como esta, existen componentes wincontrols que permiten programar por adelantado las posibles respuestas y el usuario solo debe seleccionar la respuesta apropiada en lugar de tener que escribirla.

Estos controles nos permiten definir en primera instancia un conjunto de datos o valores o respuestas asociados a una caja de edición cualesquiera, así ahora el usuario tendrá la oportunidad de seleccionar un dato del conjunto de datos o respuestas ya predefinido.

Estos componentes DEBERAN CONSTRUIRSE EN dos partes una parte de encabezado para poner el nombre del grupo de respuestas (por ejemplo, municipios, sexo, etc.) estos los podrán poner en label apropiados y acomodados

La segunda parte es la lista de opciones o respuestas que se debe cargar al tiempo de ejecución de la forma como lo muestra el siguiente programa:

Ejemplo:

```
private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
     label1.Text = listBox1.SelectedItem.ToString();
     label2.Text = comboBox1.SelectedItem.ToString();
}
```

Notas:

- Se está usando un objeto LINKLABEL, en lugar de BUTTON porque como se observa en la corrida, se pueden usar también LIGAS o enlaces tipo WEB.
- Para cargar las opciones de los LISTBOX y COMBOBOX se deberá hacer un click en los **(...)** que están a un lado de la propiedad ITEMS (Collection), esto activará un pequeño editor, donde cada renglón será una opción del combobox o listbox y en propiedad TEXT se puede cargar el encabezado del combobox.
- Ya en código se usa la propiedad SelectedItem que esta apuntando o cargada con el valor o datos seleccionado por el usuario, convertida a string, recuerden que si la ocupan numérica pueden usar el PARSE apropiado para mandarla a una variable numérica.
- La diferencia en pantalla o ejecución entre ambos controles, se ve en la corrida, que esta unos párrafos más abajo.
- Recordar que estos controles tienen muchas propiedades muy útiles y que se seguirán usando a lo largo del curso.

Corrida:

ListBox es un componente DINAMICO (es decir no tiene tamaño definido) que permite procesar visualmente un **conjunto de elementos de tipo string**.

La propiedad Rows que se usa al crearlo, es solo para indicarle cuantos renglones desplegara en pantalla, es decir si se usa rows=5, en listbox se podrá capturar todos los elementos o datos que se quiera, pero solo desplegara los últimos cinco elementos.

Sin embargo, existen ciertas propiedades del listbox que permiten conocer cuántos elementos están cargados en el listbox.

Otro importante aspecto a recordar cuando se procese o programe, es que el primer índice de la lista es el índice numero 0(cero).

Este componente, contiene muchas propiedades y métodos que facilitan el trabajo con datos la más importante es su propiedad ITEMS, que a su vez tiene:

Propiedad Accion o Significado

- Items.Add(dato): Inserta un elemento al final del listbox.
- Items.Clear(): Elimina todos los elementos de la lista.
- Items.Count(): Regresa la cantidad de elementos en lista.
- Items.Sorted=true; Ordena los elementos de la lista usada solo al tiempo de diseño.
- Items.Contains(dato): Regresa true o false, si el dato se encuentra o no se encuentra en la lista.
- Items.IndexOf(dato): Regresa el índice del objeto dentro del listbox.
- Items.Insert(indice,dato): Inserta el dato en la posición indicada.
- Items.Remove(dato): Elimina el dato del listbox.
- Items.RemoveAt(indice): Elimina el dato que está en la posición indicada.
- Items[indice]: get or set el dato en la posición indicada (ver primera nota abajo).

Notas:

- **GET** y **SET** son propiedades asociadas a todos los objetos o controles y sus propiedades de Microsoft net, por ejemplo, para un textbox, si en un programa se dice alfa = text5.text; se está usando get, si se dice text5.text=500; se está usando set.
- Otro ejemplo `alfa=listbox2.Items[2];` se está usando (get)
- `listbox2.Items[4]="mama";` se está usando (set).
- Observar que no se usa propiedad text y recordar que entran y salen strings a listbox.
- Esto de get-set se puede usar para cualquier propiedad, por ejemplo, `alfa = listbox8.background;` se está usando get, pero si se codifica `listbox8.background=amarillol;` se está usando set, como se observa es importante entender y aplicar este GET-SET en todos los programas.
- Capturas: Solo se ocupará un Text, el evento click del button, y el metodo Add del ListBox.
- Proceso: Se ocupara un ciclo for y los metodos count y text de `ListBox.Items[indice].`
- Despliegues: No se ocupa porque todos los cambios son visibles.
- Pero si se quiere pasar de un ListBox a otro ListBox, entonces ciclo for, count, etc.

Checkbox y Checkedboxlist

Estos componentes *CheckBox* y *CheckedBoxList* permiten seleccionar una opción al usuario del programa o tomar una decisión directamente en pantalla.

La diferencia entre ellos aparte de cómo se programa el componente, es que checkboxlist permite agrupar mejor sus elementos internos tal como se muestra en las corridas:

Ejemplos de uso:

Observar que dos o más checkboxs pueden estar seleccionados a la vez.

Ejemplo :

```
private void button1_Click(object sender, EventArgs e)
{
 // CheckBox se tienen que validar uno por uno
 if (GATO.Checked) label1.Text = "miauu";
 if (PERRO.Checked) label1.Text = "wow";
}

private void button2_Click(object sender, EventArgs e)
{
 // checkedlistbox control similar a listbox,
 // tambien usa propiedad selecteditem
 label2.Text = CARRERA.SelectedItem.ToString();
}
```

1. Recordar que es más conveniente asignarles un NAME a todos los componentes que se estén manejando dentro de una FORMA o ventana.

CHECKBOX:

1. La propiedad NAME deberá ser diferente en cada checkbox usado también se puede agregar una propiedad `checked=true` para que aparezca ya palomeado o seleccionado el control.
2. Para **programar** este componente:

Solo recordar usar la propiedad checked en código y un if por cada checkbox.

CHECKEDBOXLIST:

1. Este control nos permite mejorar la apariencia de la salida del checkbox
2. Solo agregar un NAME al control y un ITEMS COLLECTION para sus elementos, ***para programarlo solo usar la propiedad selecteditem.***

Corrida:

Ejercicios:

1. Evaluar la función **y = 3x² - 4x + 2** para *x = 2,-5, 8* (usar un CheckBox por cada valor de x y programar cada if de cada CheckBox con la operación correspondiente y el despliegue del resultado)
2. Construir una ventana con los datos de un automóvil y abajo construir un plan de financiamiento a dos años o muestra un plan de financiamiento a tres años. (Son dos checkbox en la ventana más un montón de botones de texto o labels, para pasar los datos a panels abajo y un botón de ok) (Chekbox).
3. Construir un programa que evalué una función cualquiera on tres valores cualesquiera usando el checkboxlist.

2 - Instrucciones Iterativas

Las ***instrucciones iterativas*** son instrucciones que permiten ejecutar repetidas veces una instrucción o un bloque de instrucciones mientras se cumpla una condición. Es decir, permiten definir bucles donde ciertas instrucciones se ejecuten varias veces. Las instrucciones de ciclos son

- for
- while
- do while
- foreach : ciclo especializado en procesar y manipular arreglos y colecciones.

1. Ciclo FOR

Este ciclo es uno de los más usados para repetir una secuencia de instrucciones sobre todo cuando se conoce la cantidad exacta de veces que se quiere que se ejecute una instrucción simple o compuesta.

Su formato general es:

```
for (inicializacion; condicion; incremento)
{
    instruccion(es);
};
```

En su forma simple la inicialización es una instrucción de asignación que carga una variable de control de ciclo con un valor inicial.

La condición es una expresión relacional que evalúa la variable de control de ciclo contra un valor final o de parada que determina cuando debe acabar el ciclo.

El incremento define la manera en que la variable de control de ciclo debe cambiar cada vez que el computador repite un ciclo.

Se deben separar esos 3 argumentos con punto y coma (;)

Ejemplo:

```
private void button1_Click(object sender, EventArgs e)
{
    int reng;
    LISTA.Items.Clear();
    for (reng = 1; reng <= 10; reng++)
        LISTA.Items.Add(reng.ToString() + " mama");
}
```

Notas:

- Se está usando un objeto listbox con NAME=LISTA para procesar el conjunto de datos recordar que listbox, comboboxlist son objetos similares y que pueden almacenar conjuntos de datos, cosa que los textbox y label no permiten.
- Se está usando la propiedad add de la colección ítems del componente o control listbox (LISTA).
- Recordar que para encadenar strings en visual c# se usa el signo +
- Como dentro del listbox entran y salen puros datos strings la variable numerica reng de tipo entero se está convirtiendo a string dentro del listbox.
- Y el método `items.clear()` es porque cuando el usuario usa el click más de una vez el control listbox los agrega abajo del listbox por eso en cuanto se activa el onclick lo primero que se realiza es limpiar el listbox.
- El ciclo for es muy sencillo y no ocupa mucha explicación, solo empieza en UNO y se va incrementando de UNO en UNO.

Corrida:

Casos Particulares del ciclo for:

1. El ciclo comienza en uno y se incrementa de uno en uno este es el caso mas general.
2. Pero el valor inicial puede se diferente de uno, ejemplo;
   for(x=5;x⇐15;x=x+1){ etc.};
3. Incluso el valor inicial puede ser negativo, ejemplo;
   for (x = -3 ;x⇐ 8; x=x+1) { etc.};
4. Los incrementos también pueden ser diferentes al de uno en uno, ej.;
   for (x=1; x⇐ 20; x=x+3){ etc. };
5. Incluso pueden ser decrementos, solo que, en este caso, recordar;
   5.1. el valor inicial de la variable debe ser mayor que el valor final.
   5.2. cambiar el sentido de la condicion. ejemplo;
   for (x= 50 ; x >= 10; x= x-4) { etcetera };

6.- Solo para los casos de incrementos y decrementos de una en una unidad substituir en el for;
el x = x + 1 por x++
el x = x - 1 por x–

Ejercicios:

1. Construir un programa que despliegue los numeros del 20 al 30.
2. desplegar los enteros entre 50 y 30 acompanados de su potencia cuadrada y raíz cubica respectiva (revisar tema de operadores aritméticos).
3. desplegar los múltiplos de 5, entre 10 y 50, acompanados de su factorial y logaritmo respectivo (la misma nota de arriba).
4. desplegar la tabla de multiplicar que el usuario indique.
5. evaluar la función **$y = 5x^2 + 3x + 8$** cuando x → -3...10 (rango de -3 hasta 10)

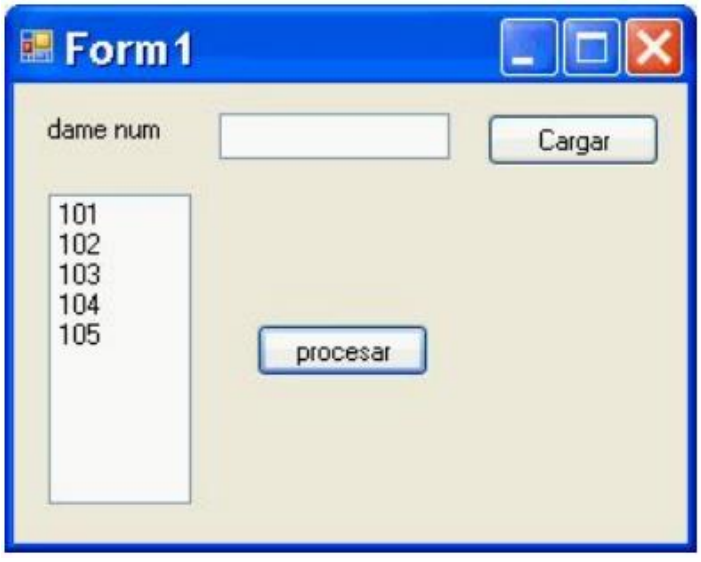
Ejemplo con listBox:

```
private void button1_Click(object sender, EventArgs e)
{
  LISTA1.Items.Add(DATO.Text);
  DATO.Text = " ";
}

private void button2_Click(object sender, EventArgs e)
{
  int r, cantidad, dato;
  cantidad = LISTA1.Items.Count;
  for (r = 0; r <= cantidad - 1; r++)
  {
    dato = Int32.Parse(LISTA1.Items[r].ToString());
    dato = dato + 100;
    LISTA1.Items[r] = dato.ToString();
  };
}
```

Recordar que el primer índice en un ListBox es el cero por eso el ciclo va desde el cero hasta la cantidad de elementos menos uno.

Corrida:



Ejercicios:

1. Capturar en una lista los sueldos de 6 empleados y desplegarlos en una segunda lista aumentados en un 30%
2. Capturar en una lista los pesos en kilogramos de 6 personas desplegarlos en una segunda lista convertidos a libras y ademas solo los mayores de 100 libras.
3. Capturar en sus 4 listas respectivas matricula, nombre y dos calificaciones de 5 alumnos, despues calcular una lista de promedios de calificaciones.
4. Capturar en sus listas respectivas numempleado, nomempleado, dias trabajados y sueldo diario de 5 empleados, desplegar en otra pantalla o panel la nómina, pero solo de aquellos empleados que ganan mas de $300.00 a la semana.

2. Ciclo WHILE

En este ciclo el cuerpo de instrucciones se ejecuta mientras una condición permanezca como verdadera en el momento en que la condición se convierte en falsa el ciclo termina.

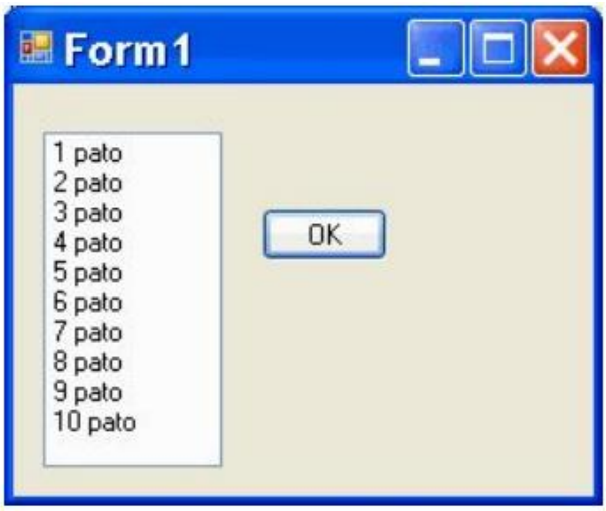
Su formato general es:

```
cargar o inicializar variable de condición;
while(condicion)
{
    grupo cierto de instrucciones;
    instruccion(es) para salir del ciclo;
};
```

Ejemplo:

```
private void button1_Click(object sender, EventArgs e)
{
    int reng = 1;
    LISTA.Items.Clear();
    while (reng <= 10)
    {
         LISTA.Items.Add(reng.ToString() + " pato");
         reng++;
    };
}
```

Corrida:



1. While puede llevar dos condiciones en este caso inicializar 2 variables de condición y cuidar que existan 2 de rompimiento o terminación de ciclo.
2. El grupo cierto de instrucciones puede ser una sola instrucción o todo un grupo de instrucciones.
3. La condición puede ser simple o compuesta.
4. A este ciclo también se le conoce también como ciclo de condición de entrada o prueba por arriba porque este ciclo evalúa primero la condición y posteriormente ejecuta las instrucciones.

Ejercicios:

1. Desplegar enteros entre 50 y 80
2. Desplegar múltiplos de 4 entre 60 y 20 acompañados de sus logaritmos de base 10 y base e respectivos (revisar tema operadores aritméticos)
3. Construir la tabla de dividir que el usuario indique
4. Evaluar una función cualquiera para el rango de valores de x de -3 a +5

3. Ciclo DO WHILE

Su diferencia básica con el ciclo while es que la prueba de condición es hecha al finalizar el ciclo, es decir las instrucciones se ejecutan cuando menos una vez porque primero ejecuta las instrucciones y al final evalúa la condición;

También se le conoce por esta razón como ciclo de condición de salida.

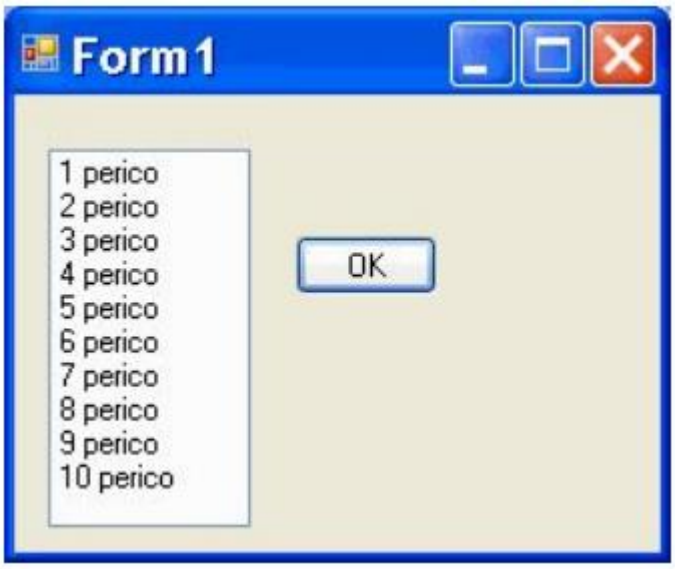
Su formato general es:

```
cargar o inicializar variable de condición;
do {
    grupo cierto de instrucción(es);
    instrucción(es) de rompimiento de ciclo;
} while (condición);
```

Ejemplo:

```
private void button1_Click(object sender, EventArgs e)
{
    int reng = 1;
    LISTA.Items.Clear();
    do
    {
        LISTA.Items.Add(reng.ToString() + " perico");
        reng++;
    } while (reng <= 10);
}
```

Corrida:



Otra diferencia básica con el ciclo while es que, aunque la condición sea falsa desde un principio el cuerpo de instrucciones se ejecutara por lo menos una vez.

Ejercicios:

1. seleccionar y construir tres tareas del for
2. seleccionar y construir dos tareas del while

4. Conclusiones Acerca De Ciclos

El problema de dado un problema O PROGRAMAS cualesquiera en visual c# 2005 cual ciclo se debe usar se resuelve con:

1. Si se conoce la cantidad exacta de veces que se quiere que se ejecute el ciclo o si el programa de alguna manera puede calcularla usar for.
2. Si se desconoce la cantidad de veces a repetir el ciclo o se quiere mayor control sobre la salida o terminación del mismo entonces usar while.
3. Si se quiere que al menos una vez se ejecute el ciclo entonces usar do while.

3. ARREGLOS

1. Introducción

Uno de los problemas más comunes en los diversos sistemas de información es el tratamiento o procesamiento de un gran volumen de datos o de información.

Las variables usadas hasta ahora en visual c# reciben propiamente el nombre de variables escalares porque solo permiten almacenar o procesar un dato a la vez.

Por ejemplo, si se quiere almacenar nombre y edad de 15 personas con el método tradicional se ocuparán 30 variables y solo es nombre y edad de 15 personas, agreguen más datos y más personas y ya es tiempo de empezar a analizar otro tipo de variables.

Es decir, en problemas que exigen manejar mucha información o datos a la vez, variables escalares no son suficientes ya que su principal problema es que solo permiten almacenar y procesar un dato a la vez.

Se ocupan entonces variables que sean capaces de almacenar y manipular conjuntos de datos a la vez.

Variables de tipo arreglo si permiten almacenar y procesar conjuntos de datos del mismo tipo a la vez.

Cada dato dentro del arreglo se le conoce como elemento del arreglo y se simboliza y procesa (captura, operación, despliegue) usando el nombre del arreglo respectivo y un subíndice indicando la posición relativa del elemento con respecto a los demás elementos del arreglo, solo recordar que en VISUAL C# la primera posición, elemento o renglón es el 0 (cero), ej.:

```
NOMBRES
Juan  -> nombres(0)
Pedro → nombres(1)
Rosa  -> nombres(2)
Jose  -> nombres(3)
```

Sin embargo sus problemas son similares a los de variables normales es decir hay que declararlos, capturarlos, hacer operaciones con ellos, desplegarlos, compararlos, etc.

2. Arreglos

En programación tradicional siempre se manejan dos tipos de arreglos los arreglos tipo listas, vectores o unidimensionales y los arreglos tipo tablas, cuadros, concentrados, matrices o bidimensionales en ambos casos son variables que permiten almacenar un conjunto de datos del mismo tipo a la vez, su diferencia es en la cantidad de columnas que cada uno de estos tipos contiene, como en los siguientes ejemplos:

a) LISTAS

EDAD
18
34
22
15

b) TABLAS

INGRESO MENSUAL DE VTAS (MILES DE $)

	ENE	FEB	MAR	ABR
SUC A	10	20	30	40
SUC B	50	60	70	80
SUC D	90	100	110	120

Como se observa la diferencia principal entre un arreglo tipo lista y un arreglo tipo tabla son las cantidades de columnas que contienen.

Nota Importante:

- Los conceptos manejados aquí están enfocados a los sistemas de información contables financieros y administrativos.
- En algebra matricial, si son importantes los conceptos de vectores y matrices, pero las operaciones y métodos son precisamente los del algebra matricial.

3. Arreglo Tipo Lista o Vectores

Un arreglo tipo lista se define como una variable que permite almacenar un conjunto de datos del mismo tipo organizados en una sola columna y uno o más renglones.

También reciben el nombre de vectores en algebra o arreglos unidimensionales en programación.

Los procesos normales con una lista o con sus elementos incluyen declarar toda la lista, capturar sus elementos, desplegarlos, realizar operaciones con ellos, desplegarlos, etc.

Para declarar una lista se usa el siguiente formato;

```
Tipodato[] nomlista= new tipodato[cant de elementos o renglones];
```

Como se observa por el formato y como ya se ha indicado anteriormente en visual c# 2005 no existen tipos de datos tradicionales, en su lugar c# usa objetos derivados de las clases numéricas apropiadas, por lo que no debe sorprender que realmente se está creando un objeto arreglo derivado de la clase de los enteros.

Recordar también que, como objeto arreglo, también puede usar una serie de métodos pertenecientes a la clase numérica de la cual heredo.

Ejemplos:

```
int[] edad= new int[12];
float[] sueldos= new float[10];
string[] municipios= new strings[5];
```

Notas:

- Recordar que la primera posición o renglón en una lista es la posición o renglón 0 (cero).
- El dato capturado, proviene de momento de un componente escalar textbox y/o se usan tantos de estos controles como elementos tenga el arreglo o más fácil aún se deberá controlar la captura de elementos usando algún algoritmo sencillo de validación como lo muestra el programa ejemplo.

Ejemplo:

```
//como atributo de la clase
int[] edad = new int[5];
int reng = 0;

private void button1_Click(object sender, EventArgs e)
{
    if (reng <= 4)
    {
        edad[reng] = System.Int32.Parse(EDAD1.Text);
        reng++;   EDAD1.Text = " ";
     };
    if (reng == 5)
    {
        EDAD1.Text = "YA SON CINCO";
    };
}

private void button2_Click(object sender, EventArgs e)
{
    // LIMPIANDO LISTAS
    LISTA1.Items.Clear();
    LISTA2.Items.Clear();
    //CARGANDO LISTA EDAD CAPTURADA
    for (reng = 0; reng <= 4; reng++)
    {
        LISTA1.Items.Add(edad[reng].ToString());
    };
    //CALCULANDO Y DESPLEGANDO
    for (reng = 0; reng <= 4; reng++)
    {
        edad[reng] = edad[reng] * 12;
    };
    //usando ciclo foreach para desplegar
    foreach (int r in edad)
    {
        LISTA2.Items.Add(r.ToString());
    };
    //dejando listo el arreglo para nueva corrida
    reng = 0;
}
```

Corrida :

Notas:

- Para el caso de operaciones y comparaciones con todos los elementos de la lista a la vez se deberá usar un ciclo for con una variable entera llamada renglón, misma que también se usa como índice de la lista.
- Recordar que todos los datos internos de la lista estarán almacenados en la memoria ram del computador, para despliegues se usa un componente visual que permite manipular un conjunto de datos a la vez, el ListBox con sus métodos apropiados, pero se tiene que usar un ciclo for () para ir añadiendo o agregando elemento por elemento como se observa en el problema ejemplo que se ha venido desarrollando, en este caso se quiere desplegar las cinco edades convertidas a meses.
- Se están usando metodos apropiados de conversión de enteros a strings y viceversa.
- Casi al final se usa un ciclo foreach para desplegar el arreglo edad, como se indico este ciclo foreach se especializa en la manipulacion de arreglos y colecciones, el formato de foreach es:

```
Foreach ( tipodato varcontrol in arreglo) intruccion(es);
```

- Observar tambien que en foreach quien se procesa es la variable de control (`r.ToString()`) no el arreglo, no se aconseja usar foreach ni para cargar arreglos ni para actualizarlos, solo para navegar dentro de ellos.
- La ultima intruccion y muy importante es poner en cero las variables de control de ciclos o índice de arreglos, esto es porque el servidor mantiene el programa ejecutándose continuamente en memoria y si se vuelve a pedir la ejecución del programa, en cuento se intente capturar un nuevo dato va a marcar el error arreglo fuera de limite o arrayofbound, están avisados.
- Para inicializar una lista se debe usar el siguiente formato:

```
tipodato[] nomlista={lista de valores};
```

Ejemplo:

```
int[] edad={15,16,17,18};
float[] sueldo={40.85, 65.30, 33.33};
string[] ciudad={"tecate", "tijuana", "mexicali", "rosarito", "ensenada"};
```

Ejercicios:

1. Capturar y desplegar 5 precios de productos cualesquiera usando dos panel uno para capturar y uno para desplegar
2. Capturar 4 sueldos en un panel desplegarlos aumentados en un 25% en otro panel.
3. Capturar los datos de 5 productos comprados en una tienda, incluyendo nombre, precio y cantidad en sus 3 listas respectivas, después calcular una cuarta lista con el gasto total por cada producto desplegarlo todo en un segundo panel e incluir también el gran total.
4. Capturar en una lista solamente 6 números múltiplos de 5, se debe de estar capture y capture números hasta que se completen los 6 múltiplos de 5.

4. Arreglos Tipo Tabla o Matriz

Un arreglo tipo tabla se define como un conjunto de datos del mismo tipo organizados en dos o más columnas y uno o más renglones.

Para procesar (recordar solo operaciones y comparaciones) internamente todos los elementos de la tabla **se ocupan dos ciclos for()** uno externo para controlar renglón y uno interno para controlar columna.

Los elementos de la tabla se deberán simbolizar con el nombre de la tabla y 2 subíndices, el primer subíndice referencia al renglón y el siguiente subíndice referencia la columna los dos dentro del mismo corchete.

La declaración de una tabla será de acuerdo al siguiente formato:

```
Public static tipodato[,] nomtabla=new tipodato[cantreng, cantcol];

Ej: public static float[,] sueldos=new float[5,8];
```

Para capturar sus elementos, usaremos un textbox y un boton de captura, solo tener cuidado o mucho control sobre los índices ren y col como lo muestra el ejemplo.

Para efectuar otros procesos tales como operaciones, despliegues con todos los elementos de la tabla se deberán usar 2 ciclos un for externo para controlar renglón y un for interno para controlar columna.

Ejemplo:

```
int[,] calif = new int[2, 3];
int r=0, c=0;

private void button1_Click(object sender, EventArgs e)
{
    calif[r, c] = System.Int32.Parse(CALIF1.Text);
    c++;
    CALIF1.Text = " ";
    if (c == 3)
    {
        r++; c = 0;
    };
    if (r == 2)
    {
        CALIF1.Text = "TABLA LLENA";
        r = 0;
        c = 0;
    };
}

private void button2_Click(object sender, EventArgs e)
{
    // procesando y regalando 10 puntos a la calificacion
    for(int reng=0; reng <= 1; reng++)
        for(int col=0; col <=2; col++)
        {
            calif[reng,col]=calif[reng,col] +10;
        };
    // desplegando
    for(int reng=0; reng<=1; reng++)
    {
        // creando un renglon para despliegue
        string temp = calif[reng,0].ToString()+ " " +calif[reng,1].ToString()+ " "   + calif[reng,2].ToString();
        LISTA1.Items.Add(temp);
        // limpiando temporal antes de otro renglon
        temp = " ";
    };
}
```

Notas:

- Observar el formato de declaración y como se controlan los índices de captura r, c
- Para procesar los elementos se usan dos ciclos for y el formato tabla [reng, col].
- En este problema se usa el objeto LISTBOX para presentar el resultado.

Corrida:

- Para inicializar tablas, se usa el siguiente formato:

```
tipodato[,] nomtabla={ {val reng 0}, {val reng 1}, {val reng n} };
```

ejemplo una matriz de 3 x 4 calificaciones:

```
int[,] calif={ { 10,20,30,40}, { 50,60,70,80}, {90,10,20,30} };
```

Ejercicios:

1. Construir un cuadro que contenga los costos fijos de cuatro productos cualesquiera, que se producen en tres plantas diferentes de una empresa maquiladora (2 prog uno capturado y otro inicializado).
2. Construir un cuadro que contenga los ingresos mensuales por ventas durante los tres primeros meses del año de cuatro sucursales de una cadena de auto refacciones, agregar al final una lista que muestre los ingresos mensuales totales por meses y una segunda lista que muestre los ingresos mensuales totales por sucursal (2 progs uno capturado y otro inicializado).
3. Construir un cuadro que contenga las comisiones ganadas por tres vendedores, de los 5 tipos de línea blanca de conocida mueblería, además listas de comisiones totales y promedios ganadas por los vendedores, así como listas de comisiones totales y promedios por tipo de línea blanca.

5. Arreglos Como Parámetros

Ocasionalmente se necesita que un método reciba como parámetro un arreglo completo para recorrerlo y procesarlo. Esto es totalmente factible pero requiere tener en cuenta dos consideraciones:

En el método el parametro formal debe indicarse con el tipo del arreglo requerido y los corchetes, pero sin indicar el tamaño del mismo.

```
private void método(tipo[] arreglo) ...
```

Cuando dentro del método se intente recorrlo se desconoce el tamaño del mismo, por lo tanto debe utilizarse la instrucción foreach o la propiedad .Count del arreglo.

Por otro lado, al invocar al método debe indicarse únicamente el nombre del arreglo, sin agregar el operador []. Esto tiene sentido porque este operador extrae un elemento del arreglo y lo que se requiere es enviar el arreglo completo.

```
tipo [] arreglo  = new arreglo[tamaño];
método(arreglo);
// enviar arreglo[] no es válido, y
// enviar arreglo[i] intentaria pasar sólo el valor de esa posición i.
```

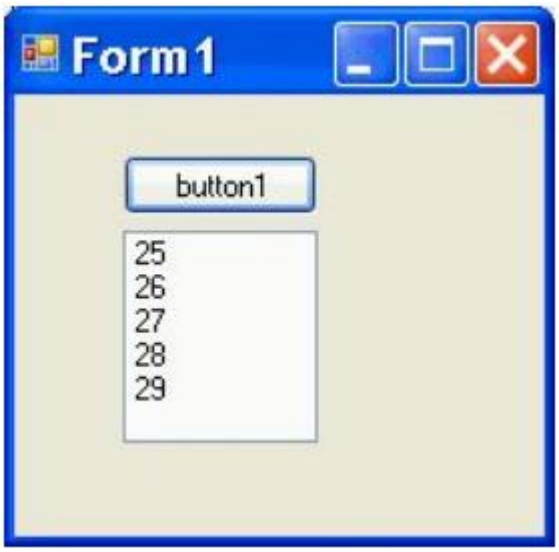
Sin embargo, es conveniente aclarar que a diferencia de variables escalares normales c# no genera una nueva variable en memoria ni tampoco copia los datos al arreglo que recibe, en su lugar se sigue usando los datos que están en el arreglo original, es por esta razón que cambios que se le hagan a los datos del arreglo que recibe realmente se estarán haciendo al arreglo original como lo muestra el siguiente ejemplo:

Ejemplo:

```
private void button1_Click(object sender, EventArgs e)
{
    // creando una lista local en memoria
    int[] lista = new int[5];
    // cargando la lista local con 10,11,12,13,14
    for (int x = 0; x <= 4; x++)
        lista[x] = x + 10;
    // Pasándola a procedimiento observar que va sin corchetes
    proc1(lista);
    for (int x = 0; x <= 4; x++)
    LISTA.Items.Add(lista[x].ToString());
}

void proc1(int[] vector) // se recibió con otro nombre y se creó sin tamaño fijo
{  // procesando la lista recibida sumándole 15
   for (int x = 0; x <= 4; x++)
   {
       vector[x] = vector[x] + 15;
   }
}  // observar que no se regresa la lista o vector recibido
```

Corrida :



Es de recordar que los cambios que le hagan al arreglo dentro del procedimiento se reflejaran en el arreglo original, es por esto que si se quiere modificar un arreglo en un procedimiento función no hay necesidad de regresar ningún valor y por tanto no se ocupan funciones.

Solo para los casos que se quiera regresar algún dato especial del arreglo, por ejemplo, regresar el primer dato par, o la suma de todos los elementos del arreglo o el promedio de todos sus elementos, etc., sólo en casos como estos se mandara un arreglo a una función.

Ejercicios:

1. Inicializar 10 edades en el principal (buttonclick) mandar la lista a un procedimiento que la convierte a meses, desplegar en principal.
2. Capturar un arreglo de 7 ciudades en un primer procedimiento, sortear en un segundo y desplegar en un tercero la lista original y la lista ordenada.