Seminario de Lenguajes (.NET)

Práctica 2

1. Qué lineas del siguiente código provocan conversiones boxing y unboxing.

**char** c1='A';

string st1="A";

object o1=c1; ---- > boxing

object o2=st1; ---- > boxing

**char** c2=(**char**)o1; ---- > unboxing

string st2=(string)o2; ---- > unboxing

2. El tipo object y los arreglos son tipos referencia. Analice porqué las líneas impresas en la consola por el siguiente código difieren entre ellas.

**public** static void **Main**(String[] args)

{

**int**[] v1= **new int**[1] {1};

**int**[] v2=v1;

v1[0] = 2;

Console.**Write**(v1[0]);

Console.**WriteLine**(" " + v2[0]);

object obj1 = 1;

object obj2 = obj1;

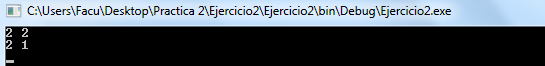
obj1 = 2;

Console.**Write**(obj1);

Console.**WriteLine**(" " + obj2);

Console.**ReadKey**();

}



3. Analice la siguiente porción de código para calcular la sumatoria de 1 a 10. ¿Cuál es el error? ¿Qué hace realmente?

**int** sum=0;

**int** i=1;

**while** (i<=10);

{

sum += i++;

}

Hay un ; de mas luego del while, por lo tanto el programa queda en un loop infinito ya que i siempre es menor a 10.

Si se saca el ; dejando bien el while funciona correctamente

4. ¿Cuál es la salida por consola si no se pasan argumentos por la línea de comandos del siguiente programa?

**using** System;

class Ejercicio4

{

**public** static void **Main**(String[] args)

{

Console.**WriteLine**(args == **null**);

Console.**WriteLine**(args.Length);

Console.**ReadKey**();

}

}



5. ¿Qué hace la instrucción **int**[] vector = **new int**[0];? ¿asigna a vector el valor null?

Creo que no asigna el valor null, crea un vector vacio, sin ninguna posición valida. REVISAR

6. Determine que hace el siguiente programa y explique qué sucede si no se pasan parámetros cuando se invoca desde la línea de comandos.

**using** System;

class HolaMundo

{

**public** static void **Main**(String[] args)

{

Console.**WriteLine**("¡Hola {0}!", args[0]);

Console.**ReadKey**();

}

}

Se produce una excepcion

7. Analice el siguiente código e indique qué líneas producen error de compilación y por qué.

**public** static void **Main**(string[] args){

**char** c;

string st;

c = ""; 🡪No se puede convertir implícitamente el tipo 'string' en 'char'

c = ''; 🡪Literal de carácter vacío (CS1011)

c = **null**; 🡪No se puede convertir null en 'char' porque es un tipo de valor que no acepta valores null (st="";

st=''; 🡪Literal de carácter vacío

st=**null**;

st=(**char**)65; 🡪 No se puede convertir implícitamente el tipo 'char' en 'string'

st=(**string**)65; 🡪No se puede convertir el tipo 'int' en 'string'

st=47.**ToString**();

st=47.89.**ToString**();

}

8. Escriba un programa que reciba una lista de nombres como parámetro e imprima por consola un saludo personalizado para cada uno de ellos.

a) utilizando la sentencia for

b) utilizando la sentencia foreach

9. Escriba un programa que reciba una lista de palabras por parámetro en la línea de comandos, e informe cuáles palabras son palíndromos (se leen igual de izquierda a derecha que de derecha a izquierda)

10. Investigue acerca de la clase StringBuilder del espacio de nombre System.Text ¿En qué circunstancias es preferible utilizar StringBuilder en lugar de utiliar String? Implemente un caso de ejemplo en el que el rendimiento sea claramente superior utilizando StringBuilder y otro en el que no.

Aunque StringBuilder y String ambos representan secuencias de caracteres, se implementan de manera diferente. String es un tipo inmutable. Es decir, cada operación que aparece para modificar un objeto de String verdaderamente crea una nueva cadena.

Por ejemplo, la llamada al método de String.Concat devuelve un objeto que tiene un valor y dirección del objeto diferentes.

Con StringBuilder ya no sólo se mejora el rendimiento considerablemente, sino que tenemos una manera más elegante y más "humana" para trabajar con largos strings.

En conclusión, cuando se trabaja con una misma variable string varias veces (concatenándola, por ejemplo), es preferible utilizar la clase StringBuilder.

11. Investigue qué miembros útiles ofrece la estructura DateTime y utilícelos para medir el tiempo de ejecución de los algoritmos implementados en el ejercicio anterior.

El ejercicio implementado esta junto el 10

12. Investigue a cerca de los métodos y propiedades que ofrece la clase Array y ArrayList. Proponga e implemente un ejercicio que utilice el método BinarySearch. Compárelo con una búsqueda secuencial.

ArrayList es una clase que representa una lista de datos. Es bastante parecida a la colección Array, con la diferencia que el ArrayList puede aumentar o disminuir su tamaño dinámicamente de una manera eficiente.

Con un array de datos no era posible aumentar la capacidad del vector ya que dicho parámetro es especificado en el momento de crear la instancia del objeto. El ArrayList a diferencia, brinda la posibilidad de aumentar o disminuir su tamaño dinámicamente según sea necesario.

Para crear una instancia de este objeto, se debe utilizar la clase ArrayList incluida en el espacio de nombre System.Collections:

ArrayList arrayList=new ArrayList();

El constructor de la clase ArrayList acepta también un parámetro tipo entero que indica la capacidad inicial del objeto que se esta creando.

Si es necesario agregar un objeto a la colección, se debe utilizar el método Add, el cual inserta el nuevo elemento en la última posición, o el método Insert el cual lo inserta en la posición indicada.

Todos los objetos almacenados en un Arraylist son tratados como objetos, por lo tanto es posible agregar todo tipo de datos, es decir, se puede agregar enteros, cadenas de texto, objetos de clases propias, etc. Y a diferencia de los array , no todos los elementos deben ser del mismo tipo de dato. Esto en algunas ocasiones puede ser una ventaja ya que permite almacenar gran variedad de información en una sola colección, sin embargo, por razones de rendimiento (cast, boxing, unboxing), hay ocasiones en las que es preferible utilizar las colecciones genéricas que serán tratadas mas adelante.

Si es necesario quitar elementos de la colección, se debe usar el método remove, removeAt o RemoveRange, los cuales eliminan el objeto pasado como parámetro, o un elemento en una posición especifica, o un grupo de elementos respectivamente.

Las propiedades mas utilizadas de esta colección son : Count y Capacity. La primera sirve para conocer la cantidad actual de elementos que contiene la colección. La segunda indica la capacidad máxima actual de la colección para almacenar elementos. Es necesario tener presente que la capacidad de la colección, aumenta en caso de ser necesario al insertar un elemento, con lo que se garantiza el redimensionamiento automático.

La capacidad de una colección nunca podrá ser menor a la cantidad total de elementos contenidos, por lo que si se modifica manualmente la propiedad Capacity y se le asigna un valor menor que el valor devuelto por la propiedad Count, obtendremos una excepción de tipo ArgumentOutOfRangeExcepetion .

13. Escriba un algoritmo que permita al usuario ingresar en una consola una serie de datos de la forma "Apellido<TAB>Documento<ENTER>". El proceso de entrada finaliza con un string vacío. Una vez finalizada la entrada de datos, el programa debe imprimir todos los números de documento ordenados numéricamente. Debe resolverlo de las siguientes maneras:

a) recorriendo cada string como si fuese un arreglo por medio de un índice

b) utilizando el método IndexOf de los objetos strings para encontrar el carácter <TAB>

c) utilizando el método split de los objetos de tipo string.

14. ¿Cuál es la salida por consola que se produce al ejecutar el siguiente método?

**private** static void **Main**(string[] args)

{

**int**[] v1 = **new int**[] {10,20,30};

**int**[] v2 = v1;

Console.**WriteLine**(v1==v2);

v2[0]=15;

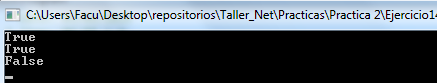
Console.**WriteLine**(v1==v2);

v2= **new int**[] {15,20,30};

Console.**WriteLine**(v1==v2);

Console.**ReadLine**();

}



15. Compruebe el funcionamiento del siguiente programa y dibuje el estado de la pila y la memoria heap cuando la ejecución alcanza los puntos indicados (comentarios en el código)

**using** System;

**using** System.Text;

class Ejemplo

{

**private** static void **Main**(string[] args)

{

object[] v = **new** object[10];

v[0]=**new StringBuilder**("Net");

**for**(**int** i=1;i<10;i++){

v[i]=v[i-1];

}

(v[5] **as** StringBuilder).**Insert**(0,"Framework .");

**foreach**(StringBuilder s **in** v)

Console.**WriteLine**(s);

//dibuje el estado de la pila y la mem. heap

//en este punto de la ejecución

v[5]=**new StringBuilder**("CSharp");

**foreach**(StringBuilder s **in** v)

Console.**WriteLine**(s);

//dibuje el estado de la pila y la mem. heap

//en este punto de la ejecución

Console.**ReadLine**();

}

}

16. Defina el tipo de datos enumerativo llamado Meses y utilícelo para:

a) Imprimir en la consola el nombre de cada uno de los meses en orden inverso (diciembre, noviembre, octubre …)

b) Imprimir en la consola el nombre de los meses pares

c) Realice un programa que pida al usuario ingresar un texto y le responda si el texto tipeado corresponde al nombre de un mes

**Nota**: en todos los casos utilice un *for* iterando sobre una variable de tipo Meses

17. Dado el siguiente tipo enumerativo:

**enum** Dias{lunes,martes,miercoles,jueves,viernes}

¿Qué instrucción elegiría si pretende imprimir en la consola el nombre de todos los días desde el lunes hasta el viernes?

**a) foreach**(Dias d **in** Dias) Console.**WriteLine**(d);

**b) for**(**int** i=0;i< Dias.Length;i++) Console.**WriteLine**((Dias)i);

**c) for**(Dias d=Dias.lunes;d<=Dias.viernes;d++) Console.**WriteLine**(d);



La respuesta correcta es C. Las opciones A y B ni siquiera compilan. En A no existe .Length y en B se usa Dias como tipo y también como variable

18. Pirámide. Escriba una función que imprima por pantalla una pirámide como la de la figura:

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

El argumento de la función es la altura de la pirámide.

19. Escriba una función (método static **int fac**(**int** n)) que calcule el factorial de un número *n* pasado al programa como parámetro por la línea de comando

a) Definiendo una función no recursiva

b) Definiendo una función recursiva

20. Idem. al ejercicio anterior pero ahora codificando un método

static void **fac**(**int** n, **out int** f)

21. Escriba una función recursiva que calcule el termino n (número entero positivo) de la serie de Fibonacci.

F(n) = 1 , si n <=2

F(n) = F(n-1) + F(n-2) , si n > 2

22. Utilizando la función factorial escriba una función que calcule el número combinatorio (*n,k*):

*n*!

(*n,k*) = 

(*n-k*)! \* *k*!

23. Escriba un programa que muestre todos los números primos entre 1 y un número dado. Para ello diseñe una función bool EsPrimo (int n) que indique si *n* es primo o no. Esta función comprobará si *n* es divisible por algún número entero entre 2 y la raíz cuadrada de *n*. (Nota: para calcular la raiz cuadrada utilice Math.sqrt() )