Práctica: Paso de arreglos a funciones

Asignatura:

Metodología y Programación Estructurada (Grupo 7)

Elaborado por:

Adriano Jezrael Almanza Sanchez

Juan Carlos Castellón Rivera

Docente:

Silvia Gigdalia Ticay Lopez

Carrera:

Ingeniería en Sistemas de la Información

Fecha de Entrega:

Domingo 20 de octubre del año 2024

Ejercicio 3 - Guía Didáctica

Registrar Estudiantes, Calcular Promedio y Conocer Promedio Más Alto y Bajo

Primero creamos un objeto el cual nos ayudará a que usemos todos los procesos de la clase Funciones en nuestra rama Main que recibirá el usuario. Luego creamos dos listas, una estará encargada de almacenar todos los nombres de los estudiantes que se registraron en la rama Main y otra la cual almacena las calificaciones de todos los estudiantes. También declaramos tres variables que recibirán los datos que ingresa el usuario al comienzo, como el nombre del estudiante, la cantidad de estudiantes a registrar y la cantidad de calificaciones a registrar.

```
//Creamos un objeto que nos permitira usar los procesos de la clase "Funciones" dentro del "Main"
Funciones funciones = new Funciones();

//Creamos una lista la cual almacenara el nombre de los estudiantes
List<string> estudiantes = new List<string>();
//Creamos otra lista la cual almacenara las calificaciones de cada estudiante
List<List<double>> calificaciones = new List<List<double>>();

//Declaramos las variables que ingresara el usuario en el "Main"
string nombre;
int numestudiantes, numcalificaciones;
```

Le damos la bienvenida al usuario al programa y luego le preguntamos la cantidad de estudiantes que el desea registrar, este proceso estará dentro de un bucle “do-while”, para así verificar que la cantidad de estudiantes a registrar sea de un valor lógico, no negativo, y en caso que lo sea, gracias al bucle y un “if” se le informará al usuario del error y se repetirá la pregunta hasta qué si sea un valor correcto.

```
Console.WriteLine("Bienvenido al sistema de gestión de calificaciones");

//Con un bucle "do-while" para verificar que el usuario ingrese un valor lógico para la cantidad de estudiantes
do
{
    Console.WriteLine("");
    Console.Write("¿Cuántos estudiantes desea registrar? ");
    numestudiantes = int.Parse(Console.ReadLine());

    //Mediante un "if" se le informa al usuario del error, y lo manda de regreso a ingresar un valor correcto
    if (numestudiantes < 0)
    {
        Console.WriteLine("");
        Console.WriteLine("Valor Incorrecto de Estudiantes, Intente Otra Vez...");
    }
}
while (numestudiantes < 0);
```

Si el valor ingresado es correcto, dentro de un ciclo “for” se irá registrando todos los datos necesarios del estudiante, como el nombre y sus calificaciones. Se le solicitará al usuario el nombre del estudiante, y con ayuda de la variable “i” se irá aumentando un contador que indica el número del estudiante que se registra. Ahora creamos una lista la cual se encargará de almacenar todas las notas de cada estudiante, y creamos una variable la cual ayudará a que el usuario ingrese la nota y que luego esta pase a la lista.

```
//Una vez tenga el valor correcto, mediante un ciclo "for" se va ingresando los datos del estudiante
for (int i = 0; i < numestudiantes; i++)
{
    Console.WriteLine("");
    Console.Write("Ingrese el nombre del Estudiante No.{0}: ", i + 1);
    nombre = Console.ReadLine();

    //Creamos una lista la cual almacenara las notas de los estudiantes de los distintos cortes
    List<double> notas = new List<double>();
    //Creamos también una variable que almacenara cada nota para después pasarla a la lista
    double nota;
```

Al igual que la cantidad de estudiantes a registrar, dentro de un bucle “do-while” y la ayuda de un “if” verificaremos que la cantidad de calificaciones que se ingresaran para el estudiante sean correctas, y si no lo son, se le informará al usuario y se reiniciará la pregunta.

```
//Igualmente, mediante "do-while" verificamos que la cantidad de notas a ingresar sea un valor lógico
do
{
    Console.WriteLine("");
    Console.Write("¿Cuántas calificaciones desea ingresar para {0}? ", nombre);
    numcalificaciones = int.Parse(Console.ReadLine());

    //Mediante un "if" informamos al usuario del error, y es regresado a la cantidad de notas
    if (numcalificaciones < 0)
    {
        Console.WriteLine("");
        Console.WriteLine("Valor Incorrecto de Calificaciones, Intente Otra Vez...");
    }
}
while (numcalificaciones < 0);
```

Otra vez, dentro de otro “for” del mismo proceso anterior se realizará el registro de notas del estudiante, y mediante un “if” y “else” se verificará que la nota ingresada sea correcta, si lo es, esta será guardada en la lista de notas anteriormente creada, en caso que no, se le informará al usuario, y se reinicia el contador para que vuelva a ingresar la nota correcta.

```
//Nuevamente con un "for" se va ingresado las calificaciones del estudiante
for (int j = 0; j < numcalificaciones; j++)
{
    Console.WriteLine("");
    Console.Write("Ingrese la calificación del {0} Corte: ", j + 1);
    nota = double.Parse(Console.ReadLine());

    //Mediante un "if" se verifica que la nota sea lógica, no un valor negativo
    if (nota > 0)
    {
        //Se registra la nota ingresada en la lista de notas en caso que si sea correcto
        notas.Add(nota);
    }
    //En caso que no sea, se reinicia el contador del bucle y se debe ingresar la nota otra vez
    else
    {
        //Le informamos al usuario que es incorrecta la nota y reiniciamios el contador
        Console.WriteLine("");
        Console.WriteLine("Valor Incorrecto de Nota, Intente Otra Vez...");
        j--;
    }
}
```

Dentro de la clase “Funciones”, crearemos una función la cual se encargará de guardar todos los datos de los estudiantes desde la rama “Main” a la clase, y una vez ocurra se agregaran tanto los nombres como las notas.

```
//Creamos una función para agregar los estudiantes la cual recibira los mismos datos del "Main" con sus tipos
1 referencia
public void AgregarEstudiante(List<string> estudiantes, List<List<double>> calificaciones, string nombre, List<double> notas)
{
    //Agrega el nombre del estudiante del "Main" a la lista de estudiantes de la clase "Funciones"
    estudiantes.Add(nombre);
    //Agrega las notas del estudiante del "Main" a la lista de calificaciones de la clase "Funciones"
    calificaciones.Add(notas);
}
```

Ahora simplemente llamamos a la función ya creada, y que se envíen todos los datos ingresados dentro del “Main” para que estos sean usados en las próximas funciones de la clase.

```
//LLamamos a la función "AgregarEstudiante" de la clase "Funciones" y registra todo los datos ingresados a la clase
funciones.AgregarEstudiante(estudiantes, calificaciones, nombre, notas);
```

Creamos otra función la cual se encargará de calcular el promedio de los estudiantes, para esto recibirá la lista de notas. Dentro de la función, declaramos la variable que guardará la suma de las notas, por ello está inicializada en 0. Luego dentro de un bucle “foreach” se recorren todas las notas registradas de los estudiantes registrados, y conforme pase dicho bucle se irán sumando cada nota. Una vez terminado, el valor regresado será el cálculo del promedio, que es la suma de las notas entre la cantidad de notas ingresadas.

```
//Creamos una función para calcular el promedio de los estudiantes y recibira unicamente la lista de notas del "Main"
1 referencia
public double CalcularPromedio(List<double> notas)
{
    //Creamos una variable la cual sumara todas las nota y estara inicializada en 0
    double suma = 0;
    //Usamos un bucle "foreach" para recorrer todas las notas registradas de cada estudiante
    foreach (var nota in notas)
    {
        //Conforme pasa el ciclo, se ira sumando cada nota y sera guardada en la variable de "suma"
        suma += nota;
    }
    //Regresamos al "Main" la suma de todas las notas entre la cantidad de notas que hubieron, que resulta en el promedio
    return suma / notas.Count;
}
```

Ahora crearemos la última función la cual se encargará de determinar cuál es el estudiante con el promedio más alto y el más bajo, y este recibirá la lista de los estudiantes y de las calificaciones. Declaramos dos variables de tipo string para poder almacenar los nombres de los estudiantes que tendrán los promedios más altos y bajos, es por ello que están vacías. Luego declaramos dos variables más de tipo double que guardaran los promedios más altos y bajos y que solo recibirán los valor más altos y bajos.

```
//Creamos una función que determinara quien tiene el promedio más alto y bajo, recibiendo 2 listas
1 referencia
public void DeterminarAltoBajoEstudiante(List<string> estudiantes, List<List<double>> calificaciones)
{
    //Declaramos 2 variables tipo string que estaran vacias y ayudaran a almacenar los nombres de los dos estudiantes
    string estudiantemayor = "", estudiantemenor = "";
    //Declaramos otras dos que guardaran los promedios más alto, bajo y solo recibiran los resultados más altos y bajos
    double promediomayor = double.MinValue, promediomenor = double.MaxValue;
```

Dentro de un ciclo “for” se recorren todas las calificaciones de todos los estudiantes registrados, y también conforme vaya pasando el ciclo se crea una variable la cual guardará el

promedio de los estudiantes, es por ello que está en forma de arreglo y llama a la función del promedio.

```
//Con un ciclo "for" se ira verificando las calificaciones de cada estudiante registrado
for (int i = 0; i < estudiantes.Count; i++)
{
    //Conforme pase el ciclo, se crea una variable que guardara el promedio de cada estudiante registrado
    double promedio = CalcularPromedio(calificaciones[i]);
```

Usamos dos “if” para poder verificar si el promedio del estudiante es mayor al promedio más alto que se registró, y conforme pase el ciclo, se irá guardando el promedio más alto y el estudiante con dicho promedio, igualmente ocurre lo mismo con el de menor promedio.

```
//Usamos un "if" para verificar si el promedio que se revisa es mayor que el promedio más alto
if (promedio > promediomayor)
{
    //Si lo es, se actualizan las variables antes declaradas para guardar los nuevos valores conforme pase el ciclo
    promediomayor = promedio;
    estudiantemayor = estudiantes[i];
}

//Usamos otro "if" para verificar si el promedio que se revisa es menor que el promedio más bajo
if (promedio < promediomenor)
{
    //Si lo es, se actualizan las variables antes declaradas para guardar los nuevos valores conforme pase el ciclo
    promediomenor = promedio;
    estudiantemenor = estudiantes[i];
}
```

Imprimimos los resultados del proceso anterior y usamos las variables anteriormente declaradas para mostrar los resultados al usuario.

```
//Una vez calculado todo, simplemente imprimimos los resultados de los procesos y son mostrados al usuario en el "Main"
Console.WriteLine("");
Console.WriteLine("Estudiante con Promedio Más Alto");
Console.WriteLine("Nombre: {0} / Promedio: {1}", estudiantemayor, promediomayor);

Console.WriteLine("");
Console.WriteLine("Estudiante con Promedio Más Bajo");
Console.WriteLine("Nombre: {0} / Promedio: {1}", estudiantemenor, promediomenor);
```

Ahora simplemente llamamos a la función en la rama “Main” para que se muestre el resultado del estudiante con mayor y menor promedio al usuario.

```
//Llamamos a la función "DeterminarAltoBajoEstudiante" de la clase para que se realize dicho proceso y sea entregado en el "Main"
funciones.DeterminarAltoBajoEstudiante(estudiantes, calificaciones);

Console.ReadKey();
```

Ejercicio 2

Ventas Diarias de un Negocio, Cálculo del Total Vendido y Día con Más Ventas

Primero declaramos tres variables las cuales 2 serán para almacenar datos y 1 para leer los días que el usuario registrará. Una vez declaradas, le damos la bienvenida al usuario al programa y se le pide que ingrese cuantos días va a registrar, esto dentro de un bucle “do-while”, ya que se verificará que el valor ingresado sea lógico y no sea negativo, si es incorrecto se le informa al usuario mediante un “if” para verificarlo y se regresa la solicitud de días.

```
//Declaramos tres variables, los días a registrar, el día con más ventas, y el total de ventas
int días, maxdía;
double totalventa;

Console.WriteLine("Bienvenido al registro de ventas diaras");
//Usando un bucle "do-while" se verifica que los días ingresado sean de un valor correcto
do
{
    Console.WriteLine("");
    Console.Write("Cuantos días desea registrar? ");
    días = int.Parse(Console.ReadLine());

    //Mediante un "if" se verifica los días
    //Si no son un valor correcto, se le informa al usuario y se reincia la solicitud de días
    if (días < 0)
    {
        Console.WriteLine("");
        Console.WriteLine("Valor Incorrecto de Días, Intete Otra Vez...");
    }
}
while (días < 0);
```

Ahora, se crea una clase la cual guardará las funciones que cumplira el sistema, y se le llama a esta clase mientras se crea una variable de ventas diarias que se usarán en el "Main", y luego mediante un "new" se mandará la variable de los días a registrar a dicha clase, y luego se llama la función se agregar las ventas.

```
//Se llama a la clase "Ventas" y se crea una variable de las ventas diarias para ser usada en el "Main"
//Se manda la variable de los "días" a registrar a la clase de "Ventas"
Ventas ventasdiarias = new Ventas(días);

//Se llama a la función de agregar las ventas
ventasdiarias.AgregarVentas();
```

Dentro de la clase "Ventas", declaramos 2 variables, una que funcionará como "arreglo" para guardar las ventas registradas, y otra la cual guardará los días a registrar. Ahora con la variable de "ventas", se igualara a los días que se desean registrar, y con la variable "díasv" se igualara a los días ingresados en el "Main".

```
1 referencia
public Ventas(int dias)
{
    //La variable "ventas" dependera de los días que se desean registrar
    ventas = new double[dias];
    //La variable "díasv" dependera de lo ingresado en el "Main"
    díasv = dias;
}
```

Creamos una función tipo "Void" la cual registrará las ventas, dentro de un bucle "for" el cual acabará hasta que el índice "i" sea menor que los días a registrar. Mediante la "i" se muestra un contador del día en que se registra y luego se manda al usuario que ingrese la venta de dicho día. Igualmente que los días solicitados, se verifica que el valor ingresado sea lógico y no negativo, y en caso que sea negativo se le informará al usuario y se elimina del índice el valor registrado mediante "i--", a su vez, el contador se reinicia.

```
1 referencia
public void AgregarVentas()
{
    //Usamos un bucle "for" para ir registrando cada día ingresado por el usuario
    for (int i = 0; i < díasv; i++)
    {
        //Usando la variable "i", se crea un contador indicado el día
        Console.WriteLine("");
        Console.WriteLine("Día No.{0}", i + 1);
        Console.Write("Ingrese la venta del día: ");
        ventas[i] = double.Parse(Console.ReadLine());

        //Usando un "if", se observa que la venta ingresada es correcta
        //Si no lo es, se le informara al usada, se eliminara del arreglo y se reinicia el contador
        if (ventas[i] < 0)
        {
            Console.WriteLine("");
            Console.WriteLine("Valor Incorrecto de Venta, Intete Otra Vez...");
            i--;
        }
    }
}
```

Creamos una función la cual calculará el total de ventas de todos los días, para esto se declara una variable la cual comenzará desde 0 y dentro de un bucle “foreach” se recorrerá todas las ventas ingresadas por el usuario, y conforme pase esto se irán sumando todas las ventas dentro de la variable del total de ventas. Una vez se recorran todas las ventas, se regresará el total de ventas al “Main”.

```
1 referencia
public double TotalVendido()
{
    //Se declara una variable para el total de ventas y es iniciada en 0
    double totalventas = 0;
    //Con un bucle "foreach" se recorre todas las ventas ingresadas
    foreach (var venta in ventas)
    {
        //Conforme pasa cada recorrido se van sumando todas las ventas en la variable del total de ventas
        totalventas += venta;
    }
    //Se regresa la variable del total de ventas en el "Main"
    return totalventas;
}
```

Dentro del “Main” se llamará la función del Total de Ventas, y con el resultado que se regrese de dicha función, este será guardado en la variable de las ventas totales para ser usado en el “Main”.

```
//Se llama a la función del total que se vendio
//Con el resultado de la función, se guardara en la variable "totalventa" para ser usada en el "Main"
totalventa = ventasdiarias.TotalVendido();
```

Creamos una función la cual determinará cuál fue el día con la mayor venta, para esto se declaran dos variables, una la cual determinara la venta máxima y es inicializada por el “arreglo” de las ventas registradas y su índice comenzará desde 0, y la otra será el índice de cuál fue el día con más ventas, por ello está desde 0. Ahora mediante un ciclo “for” se recorrerán todos los días donde se registraron las ventas, y conforme pase el ciclo, mediante un “if” se verifica si alguna de las ventas del “arreglo” es mayor que la venta mayor que ya se registro. Conforme pase el ciclo se irá actualizando la variable de la mayor venta y el índice

de donde se encontró la venta, para luego mostrarlo y regresarlo al “Main”. El valor del día máximo será sumado por uno ya que el índice del día comienza desde 0.

```
1 referencia
public int DíaVentaMasAlta()
{
    //Se declara la variable para el día máximo y la venta máxima la cual comienza desde la primera venta registrada
    double ventamax = ventas[0];
    int díamax = 0;

    //Mediante un bucle "for" se recorre todas las ventas registradas por el usuario
    //"i" comienza desde 1 por el contador de la primera función
    for (int i = 1; i < díasv; i++)
    {
        //Mediante un "if" se verifica si alguna de las ventas es la mayor
        if (ventas[i] > ventamax)
        {
            //Gracias al "for", constantemente se actualizan las dos variables declaradas
            //Se actualiza la venta máxima conforme pasa el ciclo hasta que se de la que es la mayor
            ventamax = ventas[i];
            //También se registra el día donde esta dicha venta mediante la variable "i"
            díamax = i;
        }
    }
    //Se regresa la del día máximo en el "Main" y se suma 1 para pasar del día 0 a 1
    return díamax + 1;
}
```

Dentro del “Main” se llamará la función de la Venta Más Alta, y con el resultado que se regrese de dicha función, este será guardado en la variable del día con más ventas para ser usado en el “Main”.

```
//Se llama la función de día con la venta más alta
//Con el resultado de la función, se guardara en la variable "maxdía" para ser usada en el "Main"
maxdía = ventasdiarias.DíaVentaMasAlta();
```

Ahora simplemente con las variables con los resultados, son mostradas al usuario. Primero mostrar el total de ventas, donde se usa los “días” a registrar para mostrar cuántos días fueron y el “totalventa” para mostrar la venta total. Por último, mostramos el día con más ventas donde con la variable “max día” se mostrará cual fue dicho día.

```
//Se muestra el total de ventas con la variable de "días" y el "totalventa"
Console.WriteLine("");
Console.WriteLine("Total Vendido");
Console.WriteLine("El total de ventas en {0} días fue de: {1}", días, totalventa);

//Se muestra el día con más ventas mediante la variable de "maxdía"
Console.WriteLine("");
Console.WriteLine("Día con más Ventas");
Console.WriteLine("El Día con más Ventas fue el Día No.{0}", maxdía);

Console.ReadKey();
```

Ejercicio 4

Calcular el Factorial de Cada Elemento Dentro de un Arreglo

Primeramente, dentro del main, se le solicita el tamaño del arreglo. Luego se crean dos arreglos de tamaño “n”. El primero almacena los elementos (el arreglo original). El segundo almacenará los factoriales de cada elemento del arreglo original.

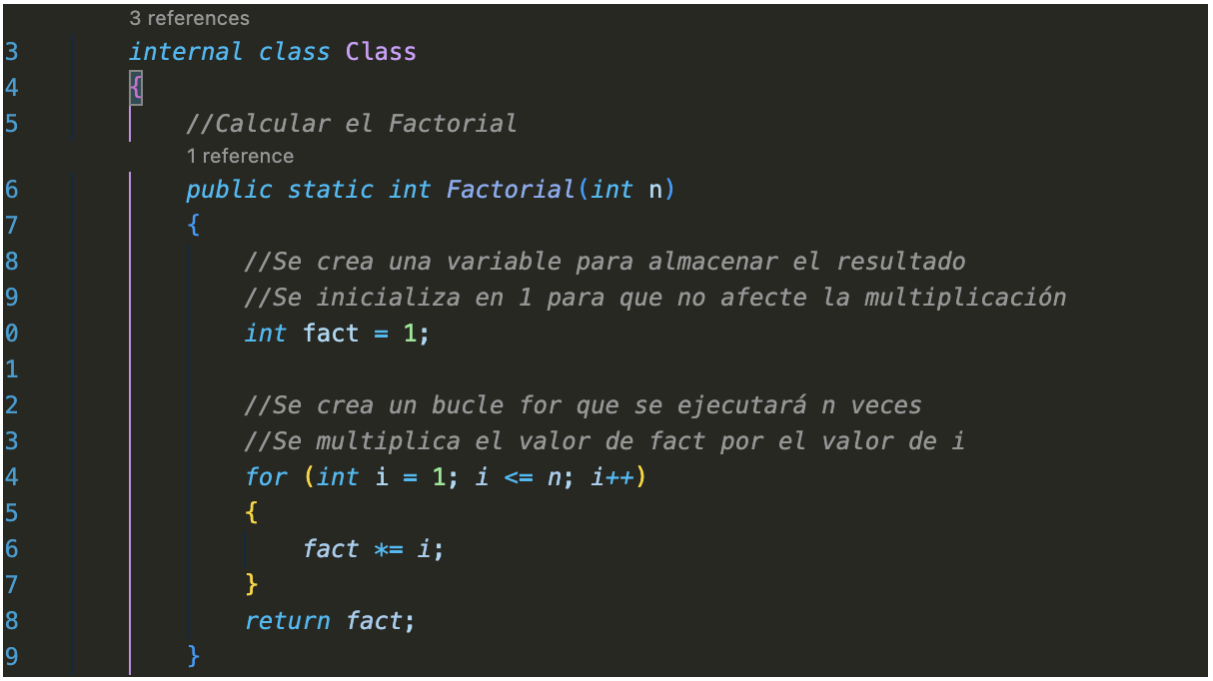
```
0 references
static void Main(string[] args)
{
    Console.WriteLine("----PROGRAMA QUE CALCULA EL FACTORIAL DE UN ARREGLO DE ENTEROS----");

    //Se solicita el tamaño del arreglo
    Console.WriteLine("\nIngrese el tamaño del arreglo: ");
    int n = Convert.ToInt32(Console.ReadLine());
    //Se crean los arreglos
    int[] array = new int[n];
    int[] factArray = new int[n];
```

Luego, se crea un bucle para leer los elementos que brinda el usuario. Sin embargo, dentro de este bucle se incluye otro bucle; un do-while. Dentro de este otro bucle, se evalúa una condición, en este caso, el número ingresado debe ser positivo. En caso de que no lo sea, le vuelve a pedir el número al usuario. En caso de que si sea positivo, llamamos una función dentro de otra Clase para calcular el factorial de n número, y el bucle do-while se rompe y pasa a la siguiente iteración del bucle principal.

```
    for (int i = 0; i < n; i++)
    {
        //Se crea un bucle do-while en caso de que el usuario ingrese un número negativo
        do
        {
            //Se solicita un número al usuario
            Console.Write("\nIngrese un número: ");
            array[i] = Convert.ToInt32(Console.ReadLine());
            //Se verifica que el número sea positivo
            if (array[i] <= 0)
            {
                Console.WriteLine("Error. Ingrese un número positivo.");
            }
            else
            {
                factArray[i] = Class.Factorial(array[i]);
                break;
            }
        }while(true);
    }
```

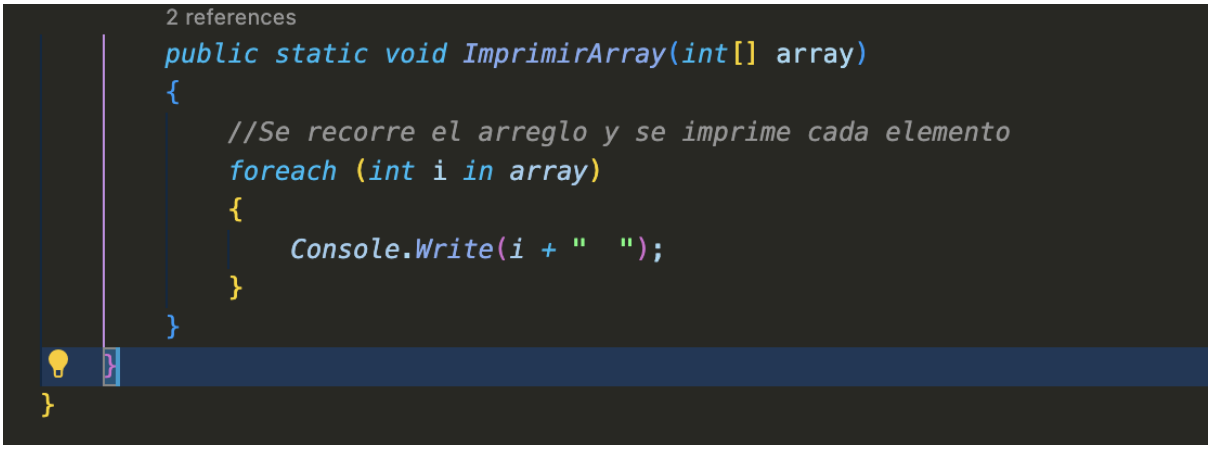
Creamos una clase externa para almacenar nuestras funciones. La primera función es para calcular el factorial de un número. Definimos una variable “fact” para almacenar el resultado.

Luego, dentro de un bucles se hace la operación necesaria para ir obteniendo el catorial. Es decir, ir acumulando dentro de fact, el producto de fact * i. Finalmente, retornamos fact.



```
3 references
internal class Class
{
    //Calcular el Factorial
    1 reference
    public static int Factorial(int n)
    {
        //Se crea una variable para almacenar el resultado
        //Se inicializa en 1 para que no afecte la multiplicación
        int fact = 1;

        //Se crea un bucle for que se ejecutará n veces
        //Se multiplica el valor de fact por el valor de i
        for (int i = 1; i <= n; i++)
        {
            fact *= i;
        }
        return fact;
    }
```

Esta otra función simplemente es para recorrer el arreglo e imprimir cada elemento dentro de él.



```
2 references
public static void ImprimirArray(int[] array)
{
    //Se recorre el arreglo y se imprime cada elemento
    foreach (int i in array)
    {
        Console.Write(i + "  ");
    }
}
}
```

Volviendo a main, simplemente se imprimen los dos arreglos, el original y el de factoriales, llamando a nuestra función ImprimirArray.

```
//Se imprimen los arreglos
Console.WriteLine("----Arreglo original----");
Class.ImprimirArray(array);

Console.WriteLine("\n\n----Arreglo de factoriales----");
Class.ImprimirArray(factArray);
Console.ReadKey();
```

Ejercicio 5

Programa para Invertir un Arreglo de Enteros

En primer lugar, solicitamos el tamaño del arreglo que desee el usuario. Con este tamaño, se crean dos arreglos. Uno para el arreglo original y otro para el arreglo invertido.

```
Console.WriteLine("---PROGRAMA QUE INVIERTE UN ARREGLO DE ENTEROS---");

//Se solicita el tamaño del arreglo
Console.Write("\nIngrese el tamaño del arreglo: ");
int n = Convert.ToInt32(Console.ReadLine());
//Se crean los arreglos
int[] array = new int[n];
int[] invert = new int[n];
```

Se establece un bucle for para la lectura de los elementos del arreglo original.

```
for (int i = 0; i < n; i++)
{
    //Se solicita un número al usuario
    Console.Write("\nIngrese un número: ");
    array[i] = Convert.ToInt32(Console.ReadLine());
}
```

Dentro de nuestra Clase externa, llamada Funciones, definimos un procedimiento para invertir el arreglo. Recorremos el arreglo original de "derecha a izquierda" y almacenamos de "izquierda a derecha" cada elemento en el arreglo invertido.

```
public static void InvertirArray(int[] array, int[] invert)
{
    //Se recorre el arreglo de forma inversa y se almacena en el arreglo invertido
    for (int i = array.Length - 1, j = 0; i >= 0; i--, j++)
    {
        invert[j] = array[i];
    }
}
```

Definimos otra función para calcular la cantidad de números impares dentro de un arreglo. Utilizando un bucle foreach, se evalúa cada uno de los elementos con una condición (que el residuo sea distinto de 0, es decir que sea impar). En caso de que esta condición se cumpla, incrementamos en uno nuestra variable "count" que llevará control de la cantidad de números impares. Finalmente, retornamos el valor de count.

```
1 reference
public static int cantidadImpares(int[] array)
{
    //Se recorre el arreglo y se cuenta la cantidad de números impares
    int count = 0;
    foreach (int i in array)
    {
        //Condicion para evaluar si el número es impar
        if (i % 2 != 0)
        {
            count++;
        }
    }
    return count;
}
```

Otro procedimiento, será para recorrer el arreglo para imprimir cada elemento.

```
2 references
public static void ImprimirArray(int[] array)
{
    //Se recorre el arreglo y se imprime cada elemento
    foreach (int i in array)
    {
        Console.Write(i + "  ");
    }
}
```

De regreso al Main, llamamos nuestras dos funciones, la de invertir y la de los impares. Se le dan los parámetros necesarios y guardamos el valor de cantidadImpares dentro de una variable "impares".

```
Funciones.InvertirArray(array, invert);
int impares = Funciones.cantidadImpares(array);
```

Finalmente, utilizando nuestro procedimiento, imprimimos el arreglo original, el arreglo invertido y la cantidad de números impares.

```
//Se imprimen los arreglos
Console.WriteLine("\n---Arreglo Original---");
Funciones.ImprimirArray(array);
Console.WriteLine("\n\n---Arreglo Invertido---");
Funciones.ImprimirArray(invert);
Console.WriteLine("\n\n\nCantidad de números impares: " + impares);
```