ANALISIS DE PROBLEMAS ACTIVIDAD 1:

William Yair Lamprea Blanco

2184588

**1- Leer un número por teclado y definir si es par o impar:**

SOLUCION

Análisis: un número se define como par es cuando es divisible por 2 y su residuo de cero

Solución: se pide un numero por teclado y utilizamos la condición SI es divisible por es el impar SI NO es impar. Así mismo nos imprime la respuesta.

**Pseudocódigo**

**Algoritmo** testParImpar

**Escribir** "Indique un número: "

**Leer** numero

**Si** numero % 2 = 0 **Entonces**

**Escribir** numero, " es par"

**SiNo**

**Escribir** numero, " no es par"

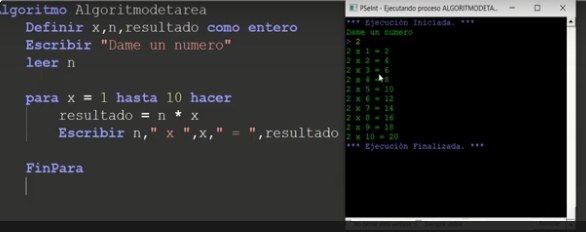
**Fin Si**

**FinAlgoritmo**

**2- Leer un número por teclado y generar con ese número la tabla de multiplicar del número 1 al 10**

R/

Solución: Para este ejercicio generaremos un bucle, lo hacemos con el ciclo for, de tal forma que utilizando un contado +1 HASTA 10 ya que mutilaremos del 1 al 10. Solicitaremos un número por teclado al usuario y lo multiplicaremos por el crescendo.



**3- Generar las tablas de multiplicar desde el numero 2 al 9, cada una desde el numero 1 al 10**

Solución: Para estos ejercicios necesitamos solicitar un numero al usuario lo ingresara por consola.

Este numero debe ser entero (int), luego utilizaremos un ciclo FOR y vamos a iterar in rango desde 1 hasta 10

Podemos almacenar el numero que ingresa el usuario en una variable la cual as su vez se operara aritméticamente con la sentencia:

Print ({c} x {numero} = {v \* numero} );

**5- Dado el vector edad = [12,50,23,11,18,35,41,85,16,45], ordenarlos en forma ascendiente**

Para el siguiente ejercicios lo desarrollaremos utilizando un arreglo unidimensional ( vector ) el cual solicita el usuario que ingrese cierto cantidad de valores numéricos (int) y luego con un ciclo (for) (int f = 0; f < numeros.Length; f++) genera un incremento de uno en uno mayor a 0

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace PruebaVector

{

class PruebaVector

{

private int[] numeros;

public void Cargar()

{

numeros = new int[5];

for (int f = 0; f < numeros.Length; f++)

{

Console.Write("Ingrese numero " + (f + 1) + ": ");

string linea = Console.ReadLine();

numeros[f] = int.Parse(linea);

}

}

public void Ordenar()

{

for (int k = 0; k < 4; k++)

{

for (int f = 0; f < 4 - k; f++)

{

if (numeros[f] > numeros[f + 1])

{

int aux;

aux = numeros[f];

numeros[f] = numeros[f + 1];

numeros[f + 1] = aux;

}

}

}

}

public void Imprimir()

{

Console.WriteLine("Ordenado Ascendentemante");

for (int f = 0; f < numeros.Length; f++)

{

Console.Write(numeros[f] + " ");

}

Console.ReadKey();

}

static void Main(string[] args)

{

PruebaVector pv = new PruebaVector();

pv.Cargar();

pv.Ordenar();

pv.Imprimir();

}

}

}

**8. Dado una palabra definir si es palíndromo o no.**

Las variables para determinar si la palabra es palíndromo o no son constantes, porque la función verificar es de tipo entero y retornara un 0 o 1 respectivamente.

Con un arreglo de tipo char, que contendrá la letra a introducir pedimos al usuario que introduzca una palabra

Luego asignamos lo que teclea el usuario a la variable

instanciamos la var "caracteres" indicando el tamaño de el arreglo que en este caso es el tamaño de la palabra introducida.

Para saber cuantos caracteres tienes nuestra cadena lo hacemos copiando los caracteres de una subcadena especificada esto lo hace el metodo ToCharArray

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

comparamos si lo que nos regresa la función es un 1 . si es así la palabra es palindrome, sino pues no lo es

\*Aquí es donde comparamos las letras de inicio a fin

**static int verificar(int pos, char[] palabra, int largo)**

LA primera condición, si la palabra en su posición 0 es igual a la ultima, por ejemplo "ojo", verifica si la pos 0 ('o') es igual a la ultima (3-0-1= 2) que también es una 'o'

aquí solo nos indica que si las pos es menor que largo - pos entra e incrementa pos cuando llega al centro de la palabra, nos indica que es un palindrome

**if (pos < largo - pos)**

**return (verificar(++pos, palabra, largo));**

**else**

**return (palindrome);**

**9-Resolver el algoritmo de Fibonacci, mostrando por consola la serie hasta una cantidad determinada de números:**

R/

Análisis: la serie de Fibonacci empieza con los numero 0 y 1 y consecutivamente se van sumando y el resultado de suma con el anterior progresivamente.

Para desarollarlo contamos con 4 variables los cuales una es para inicializar otra para hacer un cambio de posición y otra se pide al usuario el numero pedido al usuario es para equivaler en el for la cantidad de números que pide la persona y asi mostrándonos por pantalla el resultado

