NavajaSuiza.Aplicación_1_NumerosPrimos

PRUEBA DE CAJA NEGRA

Fecha de la prueba: 09/03/2017

Persona encargada de la prueba: Antonio Victoriano Pérez-Castejón Martínez

- **Clases de equivalencia**
    - **iNumero**: ha de ser un número entero mayor que 0
        - iNumero = = número entero positivo > 0.
          Valores límite: 1, 2, 2.147.483.646, 2.147.483.647, 2.147.483.648
        - iNumero == 0
        - iNumero = = número entero negativo.
          Valores límite: -2.147.483.649, -2.147.483.648, -2.147.483.647, -2, -1.
        - iNumero = = número real.
        - iNumero = = cadena de caracteres alfanuméricos
        - iNumero = = textbox vacío

- **Casos de prueba**

  Salida → Valor = resultado esperado o correcto
  Salida → Valor = resultado no esperado o incorrecto (Valor) = resultado esperado

CLASES EQUIVALENCIA	CASOS DE PRUEBA	SALIDA
iNumero == número entero positivo > 0 Valores límite: 1, 2, 2.147.483.646, 2.147.483.647, 2.147.483.648	iNumero = 1	Primo
	iNumero = 2	Primo
	iNumero = 3	Primo
	iNumero = 4	No primo
	iNumero = 2.147.483.646	No primo
	iNumero = 2.147.483.647	Primo
	iNumero = 2.147.483.648	No es un número entero correcto
iNumero == 0	iNumero = 0	Primo (Error: el 0, por definición, no se considera un número primo)
iNumero = = número entero negativo. Valores límite: -2.147.483.649, -2.147.483.648, -2.147.483.647, -2, -1.	iNumero = -2.147.483.649	No es un número entero correcto
	iNumero = -2.147.483.648	Primo (Error: el número introducido es negativo)
	iNumero = -2.147.483.647	Primo (Error: el número introducido es negativo)
	iNumero = -2	Primo (Error: el número introducido es negativo)
	iNumero = -1	Primo (Error: el número introducido es negativo)
iNumero = = número real	iNumero = 2,7	No es un número entero correcto
iNumero = = cadena de caracteres alfanuméricos	iNumero = 5abcd	No es un número entero correcto
iNumero = = textbox vacío	iNumero = “”	No es un número entero correcto

PRUEBA DE CAJA BLANCA

Fecha de la prueba: 09/03/2017

Persona encargada de la prueba: Antonio Victoriano Pérez-Castejón Martínez

```
bool ComprobarPrimo(int iNumero)
{
    bool bEsPrimo = true;
    int iModulo = 0;
    int i = 2;


    while(i <= iNumero / 2 && bEsPrimo)
    {
        iModulo = iNumero % i;

        if(iModulo == 0)
        {
            bEsPrimo = false;
        }

        i++;
    }

    return bEsPrimo;
}
```

Salida → Valor = resultado esperado o correcto
Salida → Valor = resultado no esperado o incorrecto (Valor) = resultado esperado

COBERTURA	CASOS <iNumero>	SALIDA
(1) Entrar en el While 0 veces	<1>	Primo
	<2>	Primo
	<3>	Primo
(1) Entrar en el While 1 vez	<4>	No primo
	<5>	Primo
	<6>	No primo
(1) Entrar en el While varias veces	<7>	Primo
	<35>	No primo
(2) Entrar en el If: iModulo == 0	<4>	No primo
	<6>	No primo
	<66>	No primo
(2) No entrar en el If: iModulo != 0	<3>	Primo
	<5>	Primo
	<11>	Primo

Tabla simplificada

COBERTURA	CASOS <iNumero>	SALIDA
(1) Entrar en el While 0 veces	<1>, <2>, <3>	Primo
(1) Entrar en el While 1 vez	<4>, <6>	No primo
	<5>	Primo
(1) Entrar en el While varias veces	<7>	Primo
	<35>	No primo
(2) Entrar en el If: iModulo == 0	<4>, <6>, <66>	No primo
(2) No entrar en el If: iModulo != 0	<3>, <5>, <11>	Primo

NavajaSuiza.Aplicación_4_PotenciaNumero

PRUEBA DE CAJA NEGRA

Fecha de la prueba: 09/03/2017

Persona encargada de la prueba: Antonio Victoriano Pérez-Castejón Martínez

- **Clases de equivalencia**

    - **iBase**: ha de ser un número entero (positivo o negativo)
        - iBase == número entero positivo.
          Valores límite: 0, 1, 2, 2.147.483.646, 2.147.483.647, 2.147.483.648
        - iBase == número entero negativo.
          Valores límite: -2.147.483.649, -2.147.483.648, -2.147.483.647, -2, -1.
        - iBase == número real.
        - iBase == cadena de caracteres alfanuméricos
        - iBase == “”

    - **iExponente**: ha de ser un número entero (positivo o negativo)
          iExponente == número entero positivo.
          Valores límite: 0, 1, 2, 2.147.483.646, 2.147.483.647, 2.147.483.648
          iExponente == número entero negativo.
          Valores límite: -2.147.483.649, -2.147.483.648, -2.147.483.647, -2, -1.
        - iExponente == número real.
        - iExponente == cadena de caracteres alfanuméricos
        - iExponente == “”

- **Casos de prueba**

  Salida → Valor = resultado esperado o correcto
  Salida → Valor = resultado no esperado o incorrecto (Valor) = resultado esperado

CLASES EQUIVALENCIA	CASOS DE PRUEBA	SALIDA
iBase == número entero positivo. Valores límite 0, 1, 2, 2.147.483.646, 2.147.483.647, 2.147.483.648 **iExponente** == número entero positivo. Valores límite 0, 1, 2, 2.147.483.646, 2.147.483.647, 2.147.483.648	iBase = 0 iExponente = 0	1
	iBase = 0 iExponente = 1	0
	iBase = 0 iExponente = 2	0
	iBase = 0 iExponente = 3	0
	iBase = 0 iExponente = 4	0
	iBase = 0 iExponente = 2.147.483.646	0
	iBase = 0 iExponente = 2.147.483.647	Error El programa se congela
	iBase = 0 iExponente = 2.147.483.648	¡Error! El exponente introducido debe de ser un número entero
	iBase = 1 iExponente = 0	1
	iBase = 1 iExponente = 1	1
	iBase = 1 iExponente = 2	1
	iBase = 1 iExponente = 3	1
	iBase = 1 iExponente = 4	1

	iBase = 1 iExponente = 2.147.483.646	1
	iBase = 1 iExponente = 2.147.483.647	Error El programa se congela
	iBase = 1 iExponente = 2.147.483.648	¡Error! El exponente introducido debe de ser un número entero
	iBase = 2 iExponente = 0	1
	iBase = 2 iExponente = 1	2
	iBase = 2 iExponente = 2	4
	iBase = 2 iExponente = 3	8
	iBase = 2 iExponente = 4	16
	iBase = 2 iExponente = 2.147.483.646	0
	iBase = 2 iExponente = 2.147.483.647	Error El programa se congela
	iBase = 2 iExponente = 2.147.483.648	¡Error! El exponente introducido debe de ser un número entero
	iBase = 2.147.483.646 iExponente = 0	1
	iBase = 2.147.483.646 iExponente = 1	2147483646
	iBase = 2.147.483.646 iExponente = 2	4
	iBase = 2.147.483.647 iExponente = 0	1

	iBase = 2.147.483.647 iExponente = 1	2147483647
	iBase = 2.147.483.647 iExponente = 2	1
	iBase = 2.147.483.648 iExponente = 0	¡Error! La base introducida debe de ser un número entero
	iBase = 2.147.483.648 iExponente = 1	¡Error! La base introducida debe de ser un número entero
	iBase = 2.147.483.648 iExponente = 2	¡Error! La base introducida debe de ser un número entero
iBase == número entero negativo. Valores límite: -2.147.483.649, -2.147.483.648, -2.147.483.647, -2, -1. iExponente == número entero negativo. Valores límite: -2.147.483.649, -2.147.483.648, -2.147.483.647, -2, -1.	iBase = -1 iExponente = -2.147.483.649	¡Error! El exponente introducido debe de ser un número entero
	iBase = -1 iExponente = -2.147.483.648	Error El programa se congela
	iBase = -1 iExponente = -2.147.483.647	Error El programa se congela
	iBase = -1 iExponente = -1	-1
	iBase = -1 iExponente = -2	1
	iBase = -1 iExponente = -3	-1
	iBase = -1 iExponente = -4	1
	iBase = -2 iExponente = -1	0 (-0,5)
	iBase = -2 iExponente = -2	0 (-0,25)
iBase == número entero positivo. Valores límite = 0, 1, 2	iBase = 0 iExponente = -1	Error de excepción
	iBase = 0	Error de excepción

iExponente == número negativo. Valores límite -1, -2	iExponente = -2	
	iBase = 1 iExponente = -1	1
	iBase = 1 iExponente = -2	1
	iBase = 2 iExponente = -1	0 (0,5)
	iBase = 2 iExponente = -2	0 (0,25)
iBase == número entero negativo. Valores límite = -1, -2 iExponente == número positivo. Valores límite = 0, 1 y 2	iBase = -1 iExponente = 0	1
	iBase = -1 iExponente = 1	-1
	iBase = -1 iExponente = 2	1
	iBase = -2 iExponente = 0	1
	iBase = -2 iExponente = 1	-2
	iBase = -2 iExponente = 2	4
iBase == número real iExponente == número real	iBase = 2,5 iExponente == 3,5	¡Error! La base introducida debe de ser un número entero El exponente introducido debe de ser un número entero
iBase == número real iExponente == número entero	iBase = 3,7 iExponente == -2	¡Error! La base introducida debe de ser un número entero

iBase == número entero iExponente == número real	iBase = 5 iExponente = 3,8	¡Error! El exponente introducido debe de ser un número entero
iBase == cadena de caracteres alfanuméricos iExponente == cadena de caracteres alfanuméricos	iBase = abcd iExponente = a5	¡Error! La base introducida debe de ser un número entero El exponente introducido debe de ser un número entero
iBase == cadena de caracteres alfanuméricos iExponente == número entero	iBase = 55ac iExponente = 10	¡Error! La base introducida debe de ser un número entero
iBase == número entero iExponente == cadena de caracteres alfanuméricos	iBase = 6 iExponente = xyz	¡Error! El exponente introducido debe de ser un número entero
iBase == textbox vacío iExponente == textbox vacío	iBase == “” iExponente == “”	¡Error! La base introducida debe de ser un número entero El exponente introducido debe de ser un número entero
iBase == textbox vacío iExponente == número entero	iBase == “” iExponente == 15	¡Error! La base introducida debe de ser un número entero
iBase == número entero iExponente == textbox vacío	iBase == 8 iExponente == “”	¡Error! El exponente introducido debe de ser un número entero

PRUEBA DE CAJA BLANCA

Fecha de la prueba: 09/03/2017

Persona encargada de la prueba: Antonio Victoriano Pérez-Castejón Martínez

```
int CalcularPotencia(int iBase, int iExponente)
{
    int iResultado = 1;
    bool bPositivo = true;


    if (iExponente < 0)
    {
        bPositivo = false;
        iExponente = -iExponente;
    }

    for (int i = 1; i <= iExponente; i++)
    {
        iResultado = iResultado * iBase;
    }

    if (bPositivo == false)
    {
        iResultado = 1 / iResultado;
    }

    return iResultado;
}
```

Salida → Valor = resultado esperado o correcto
Salida → Valor = resultado no esperado o incorrecto (Valor) = resultado esperado

COBERTURA	CASOS <iBase, iExponente>	SALIDA
(1) Entrar en el primer If: iExponente < 0	<6, -5>	0 (0,0001286008230452675)
(1) No entrar en el primer If: iExponente >= 0	<6, 0>	1
	<6, 6>	46656
(2) Entrar en el for 0 veces	<8, 0>	1
(2) Entrar en el for 1 vez	<9, 1>	9
(2) Entrar en el for 1 varias veces	<10, 3>	1000
(3) Entrar en el segundo If: bPositivo == false	<4, -3>	0 (0,015625)
(3) No entrar en el segundo If: bPositivo == true	<6, 8>	1679616