Laura Carratalá Castillo

Pruebas de caja negra y caja blanca

A continuación, se reflejarán de forma clara y concisa los pasos para la elaboración de las pruebas de caja negra y caja blanca de dos métodos pertenecientes a dos de nuestras aplicaciones que componen la Navaja Suiza. En nuestro caso, hemos elegido la aplicación 1, que consiste en la determinación de si un número positivo introducido es primo o no, y la aplicación 2, que consiste en mostrar los múltiplos de 3 y 5 encontrados hasta un número introducido, estando este comprendido entre el 1 y el 100.

Pruebas para la aplicación 1:

Como hemos indicado anteriormente, la aplicación 1 consiste en determinar si un número positivo introducido es primo o no. El método con el cual haremos las pruebas se llama **EsPrimo** y su código es el siguiente:

```
/// <summary>
/// Función que verifica si un número positivo introducido es o no es primo.
/// </summary>
/// <remarks>----</remarks>
/// <param name="numIntroducido">Número que introduce el usuario.</param>
/// <returns>Si es primo o no.</returns>
public bool EsPrimo(int numIntroducido)
{
    bool esPrimo = true;

    if (numIntroducido > 0)
    {
        for (int i = 2; i < numIntroducido && esPrimo; i++)
        {
            if (numIntroducido % i == 0)
            {
                esPrimo = false;
            }
        }
    }
    return esPrimo;
}
```

En primer lugar, realizaremos las pruebas de **caja negra**. Para ello, definiremos las clases de equivalencia elegidas para cubrir el mayor número de pruebas posible.

Clases de equivalencia:

- **numIntroducido:** Es un int, deberá ser mayor que 0.
    1) numIntroducido > 3 (que sea primo): Introducción de cualquier número entero positivo mayor que 3 que sea primo. Valor límite: 5.
    2) numIntroducido > 3 (que no sea primo): Introducción de cualquier número entero positivo mayor que 3 que no sea primo. Valor límite: 4.
    3) 0 < numIntroducido <= 3: Introducción de valores positivos menores o iguales a 3. Son valores especiales puesto que son susceptibles de dar algún error. Valores límite: 1, 2, 3.
    4) numIntroducido == 0: Es un valor especial. Ya no entra en el rango de valores.

5) numIntroducido < 0: Introducción de valores negativos. Valores límite: -1, - 2, -3, -4.
6) numIntroducido == decimal: Introducción de un número decimal, ya sea positivo o negativo.
7) numIntroducido == letra: Introducción de letras.
8) numIntroducido == símbolo: Introducción de caracteres no alfanuméricos.
9) NumIntroducido == "": No introducir ningún carácter.

Prueba	Clase de equivalencia	Valores de entrada	Resultado	Comentarios
1	numIntroducido > 3 (que sea primo)	5, 89	Válido	Los números son primos.
2	numIntroducido > 3 (que no sea primo)	4, 100	Válido	Los números no son primos.
3	0 < numIntroducido <= 3	1, 2, 3	Para 2 y 3: Válido Para 1: No Válido	El 2 y el 3 son primos, pero el 1 no se considera primo y el programa indica que si lo es.
4	numIntroducido == 0	0	No válido	El 0 no es primo y el programa indica que si lo es.
5	numIntroducido < 0	-1, -2, -3, -4, -50	No válido	El programa indica que son primos cuando no lo son.
6	numIntroducido == decimal	1.5, -1.5	No válido	Solo se permite la introducción de enteros y el programa nos indica que introduzcamos un elemento válido.
7	numIntroducido == letra	A, a, aa	No válido	Solo se permite la introducción de enteros y el programa nos indica que introduzcamos un elemento válido.
8	numIntroducido == simbolo	>, *	No válido	Solo se permite la introducción de enteros y el programa nos indica que introduzcamos un elemento válido.
9	numIntroducido == ""	(vacío)	No válido	Solo se permite la introducción de enteros y el programa nos indica que introduzcamos un elemento válido.

Vemos que con todas estas pruebas hemos cubierto la totalidad del código de nuestro método, con lo que ya no sería necesario realizar más pruebas.

Sin embargo, realizaremos las pruebas de **caja blanca**, cubriendo las distintas partes de las que se compone nuestro método. En total hay 3 bucles que se enumeran en rojo de la forma que vemos en la siguiente imagen:

```
/// <summary>
/// Función que verifica si un número positivo introducido es o no es primo.
/// </summary>
/// <remarks>----</remarks>
/// <param name="numIntroducido">Número que introduce el usuario.</param>
/// <returns>Si es primo o no.</returns>
public bool EsPrimo(int numIntroducido)
{
    bool esPrimo = true;

 1  if (numIntroducido > 0)
    {
      2 for (int i = 2; i < numIntroducido && esPrimo; i++)
        {
          3 if (numIntroducido % i == 0)
            {
              4 esPrimo = false;
            }
        }
    }
    return esPrimo;
}
```

Para cubrir todos los bucles, será necesario realizar únicamente las siguientes pruebas:

Cobertura	Valor de entrada	Resultado
(1)	0	No válido
(2)	1	No válido
(3)	3	Válido
(4)	4	Válido

Pruebas para la aplicación 2:

Como hemos indicado anteriormente, la aplicación 2 consiste en mostrar los múltiplos de 3 y 5 encontrados hasta un número introducido, estando este comprendido entre el 1 y el 100. El método con el cual haremos las pruebas se llama **MostrarSerieMultiplos** y su código es el siguiente:

```
/// <summary>
/// Función que devuelve la cadena de números múltiplos de 3 y 5
/// comprendidos hasta el número que introduzca el usuario.
/// </summary>
/// <remarks>----</remarks>
/// <param name="numeroIntroducido">Número que introduce el usuario.</param>
/// <returns>Cadena con los múltiplos de 3 y 5.</returns>
public string MostrarSerieMultiplos(int numeroIntroducido)
{
    string cadenaTexto = "";

    for (int i = 1; i <= numeroIntroducido; i++)
    {
        if (i % 3 == 0 || i % 5 == 0)
        {
            cadenaTexto = cadenaTexto + i + ", ";
        }
    }
    return cadenaTexto;
}
```

En primer lugar, realizaremos las pruebas de **caja negra**. Para ello, definiremos las clases de equivalencia elegidas para cubrir el mayor número de pruebas posible.

Clases de equivalencia:

- **numIntroducido:** Es un int.
    1) numIntroducido > 5: Introducción de cualquier número entero positivo mayor que 5. Valor límite: 6.
    2) 0 < numIntroducido <= 5: Introducción de valores enteros positivos entre el 3 y el 5 incluidos. Valores límite: 3, 4, 5.
    3) numIntroducido < 3: Introducción de valores positivos hasta el 2. Valores límite: 1, 2.
    4) numIntroducido == 0: Es un valor especial, ya no entra en el rango de valores permitidos.
    5) numIntroducido < 0: Introducción de valores negativos.
    6) numIntroducido == decimal: Introducción de un número decimal, ya sea positivo o negativo.
    7) numIntroducido == letra: Introducción de letras.
    8) numIntroducido == símbolo: Introducción de caracteres no alfanuméricos.
    9) NumIntroducido == “”: No introducir ningún carácter.

Prueba	Clase de equivalencia	Valor de entrada	Resultado	Comentarios
1	numIntroducido > 5	6, 7, 100	Válido	Como resultado obtenemos la cadena de múltiplos. Como el 7 no es múltiplo ni de 3 ni de 5, no se añade a la cadena.
2	3 <= numIntroducido <= 5	3, 4, 5	Válido	Como resultado obtenemos la cadena de múltiplos. Como el 4 no

				es múltiplo ni de 3 ni de 5, no se añade a la cadena.
3	numIntroducido < 3	1, 2	Válido	Como el 1 y el 2 no son múltiplos ni de 3 ni de 5, no se añaden a la cadena.
4	numIntroducido == 0	0	No válido	No entra en el rango de valores permitidos.
5	numIntroducido < 0	-1, -3, -5	No válido	No entran en el rango de valores permitidos.
6	numIntroducido == decimal	1.5, -1.5	No válido	Solo se permite la introducción de enteros y el programa nos indica que introduzcamos un elemento válido.
7	numIntroducido == letra	A, a, aa	No válido	Solo se permite la introducción de enteros y el programa nos indica que introduzcamos un elemento válido.
8	numIntroducido == símbolo	>, *	No válido	Solo se permite la introducción de enteros y el programa nos indica que introduzcamos un elemento válido.
9	numIntroducido == “”	(vacío)	No válido	Solo se permite la introducción de enteros y el programa nos indica que introduzcamos un elemento válido.

Vemos que con todas estas pruebas hemos cubierto la totalidad del código de nuestro método, con lo que ya no sería necesario realizar más pruebas.

Sin embargo, realizaremos las pruebas de **caja blanca**, cubriendo las distintas partes de las que se compone nuestro método. En total hay 2 bucles que se enumeran en rojo de la forma que vemos en la siguiente imagen:

```
/// <summary>
/// Función que devuelve la cadena de números múltiplos de 3 y 5
/// comprendidos hasta el número que introduzca el usuario.
/// </summary>
/// <remarks>----</remarks>
/// <param name="numeroIntroducido">Número que introduce el usuario.</param>
/// <returns>Cadena con los múltiplos de 3 y 5.</returns>
public string MostrarSerieMultiplos(int numeroIntroducido)
{
    string cadenaTexto = "";

  1 for (int i = 1; i <= numeroIntroducido; i++)
    {
      2 if (i % 3 == 0 || i % 5 == 0)
        {
          3 cadenaTexto = cadenaTexto + i + ", ";
        }
    }
    return cadenaTexto;
}
```

Para cubrir todos los bucles, será necesario realizar únicamente las siguientes pruebas:

Cobertura	Valor de entrada	Resultado
(1)	0	No válido, no permite entrar al bucle al no ser un valor permitido
(2)	1	Válido
(3)	3	Válido