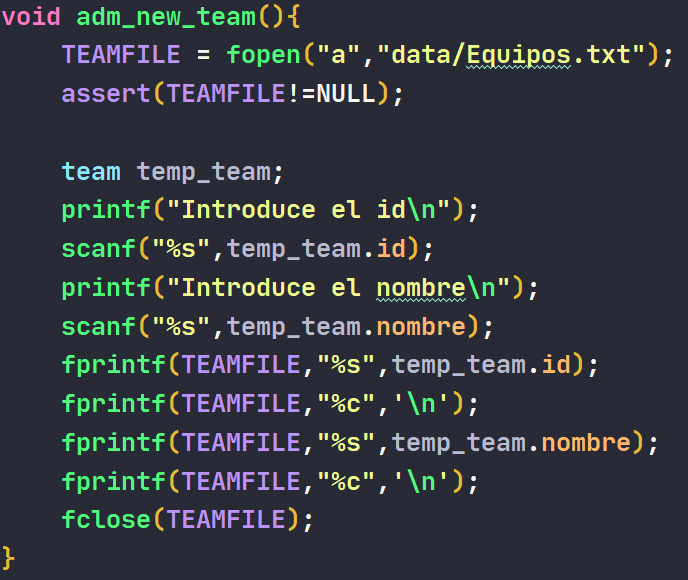
**Pruebas de software:**

**1. Módulo admin:**

**1.1 Prueba de caja negra:**

En esta prueba se cogerá la siguiente función del módulo, llamada ¨adm\_new\_team¨:



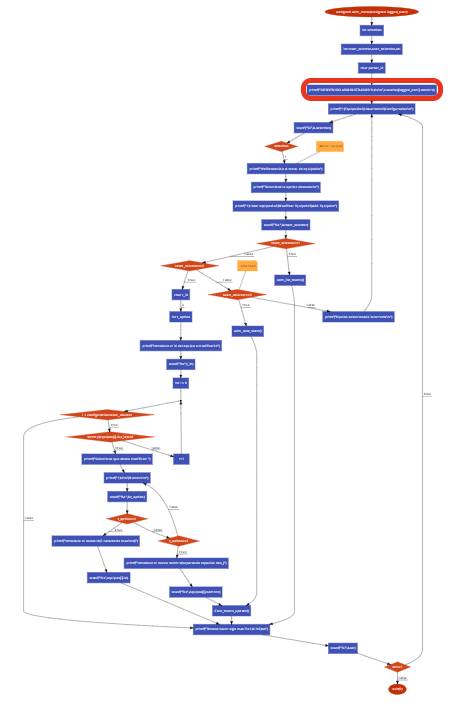
|  |  |
| --- | --- |
| **ENTRADA** | **SALIDA** |
| El fichero tiene un problema a la hora de acceder a él o simplemente no existe. | El programa no continúa con la operación gracias a assert(TEAMFILE!=NULL). |
| Introduzco los datos del nuevo equipo | Se escriben dichos parámetros en el fichero TEAMFILE. |

Se comprueba dicho cometido de la función al introducir un nuevo equipo en su fichero correspondiente, poseyendo previamente el usuario la característica de ̈administrador¨.

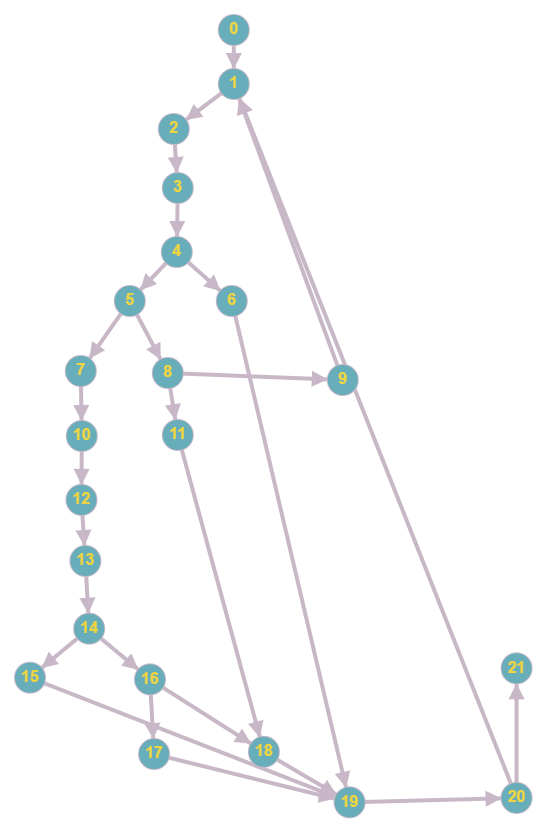
**1.2 Pruebas de caja blanca:**

1. **Prueba de ruta básica:**

Para esta parte, se usará la función de ¨adm\_menu¨. SU diagrama de flujo sería de la siguiente forma:



Su grafo simplificado correspondiente sería:



Al calcular su complejidad ciclomática da los siguientes resultados:

V(G) = NNP + 1 = 9

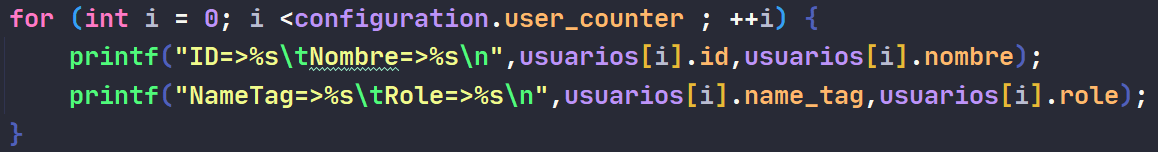
V(G) = NA - NN + 2 = 9

Al conocerse el número de rutas independientes, se mostrarán en la siguiente tabla:

|  |  |
| --- | --- |
| **RUTA BÁSICA** | **ORDEN** |
| 1 | 0, 1, 2, 3, 4, 5, 7, 10, 12, 13, 14, 15, 19, 20, 21 |
| 2 | 0, 1, 2, 3, 4, 5, 7, 10, 12, 13, 14, 16, 17, 19, 20, 21 |
| 3 | 0, 1, 2, 3, 4, 5, 7, 10, 12, 13, 14, 16, 18, 19, 20, 21 |
| 4 | 0, 1, 2, 3, 4, 5, 8, 11, 18, 19, 20, 21 |
| 5 | 0, 1, 2, 3, 4, 5, 8, 9, 1… (Continua como la ruta 1) |
| 6 | 0, 1, 2, 3, 4, 6, 18, 19, 20, 1… (Continua como en la ruta 1) |
| 7 | 0, 1, 2, 3, 4, 5, 8, 9, 1… (Continua como la ruta 2) |
| 8 | 0, 1, 2, 3, 4, 6, 18, 19, 20, 1… (Continua como en la ruta 2) |
| 9 | 0, 1, 2, 3, 4, 6, 19, 20, 21 |

1. **Prueba de bucle:**

Se realiza la prueba de bucle sobre este fragmento de código:



|  |  |  |  |
| --- | --- | --- | --- |
| **Precondiciones** | **Acciones** | **Después de iterar** | **Deseado** |
| i = configuartion.user\_counter | …….. | i es igual a la variable sobre la que se itera | No entra en el bucle.  Ocurre el resultado esperado |
| i = 0 | i++  (Primera iteración) | i = 1 | Imprime por pantalla los datos del usuario 1 |
| i = 1 | i++  (Segunda iteración) | i = 2 | Imprime por pantalla los datos del usuario 2 |
| i = configuartion.user\_counter- 1 | i++  (Última iteración) | i = configuartion.user\_counter | Imprime los datos del último usuario |
| i = configuartion.user\_counter+ 1 | …….. | i = configuartion.user\_counter + 1 | No entra en el bucle. |

**2. Módulo participante:**

**2.1 Prueba de caja negra:**

Para esta parte, se usará la función de ¨part\_list\_plantilla¨. Su diagrama de flujo sería de la siguiente forma:



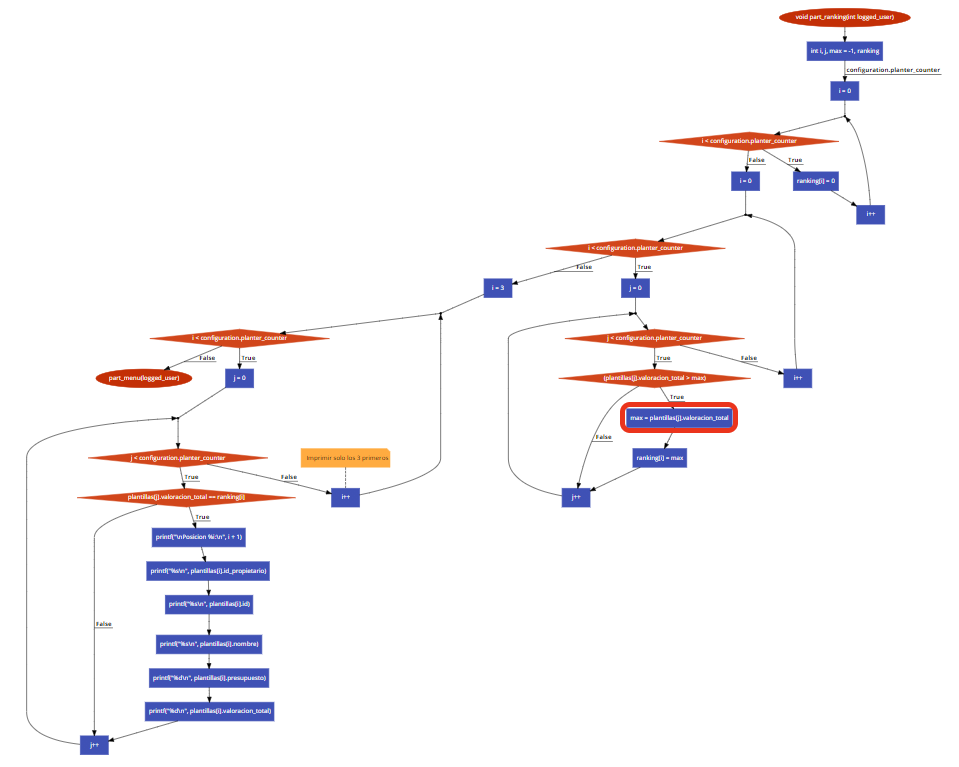
|  |  |
| --- | --- |
| **ENTRADA** | **SALIDA** |
| Recibe la id del usuario | Imprime por pantalla los datos de sus plantillas |
| El usuario posee al menos una plantilla | Imprime por pantalla dicha plantilla y vuelve al menú |
| El usuario no posee ninguna plantilla | Se le informa de que no posee ninguna y es mandado al menú |

Como se puede observar, la función cumple con su cometido, siempre y cuando el usuaria posea la característica de ¨participante¨.

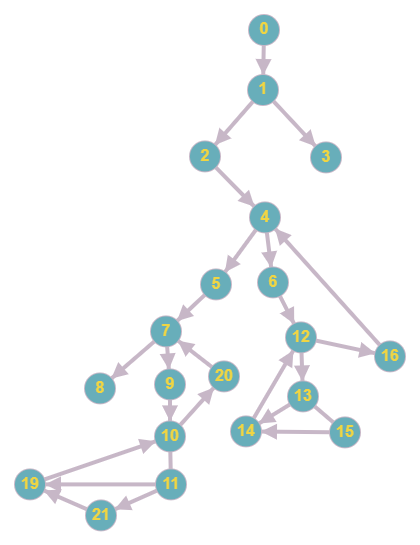
**2.2 Pruebas de caja blanca:**

1. **Prueba de ruta básica:**

Para esta prueba, se usará la función ¨part ranking¨, cuyo flujo es el siguiente:



Su grafo simplificado correspondiente sería:



Al calcular su complejidad ciclomática da los siguientes resultados:

V(G) = NNP + 1 = 6

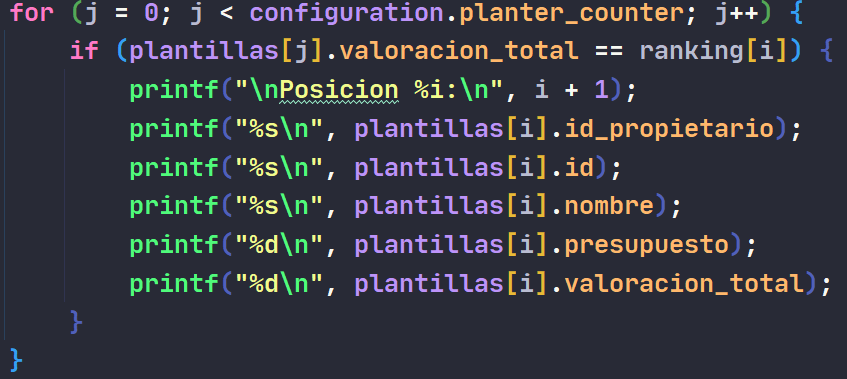
V(G) = NA - NN + 2 = 6

Al conocerse el número de rutas independientes, se mostrarán en la siguiente tabla:

|  |  |
| --- | --- |
| **RUTA BÁSICA** | **ORDEN** |
| 1 | 0, 1, 2, 4, 5, 7, 8 |
| 2 | 0, 1, 2, 4, 5, 7, 9, 10, 11, 21, 19, 19, 29, 7, 8 |
| 3 | 0, 1, 2, 4, 5, 6, 9, 10, 11, 19, 10, 20, 7, 8 |
| 4 | 0, 1, 3 |
| 5 | 0, 1, 2, 4, 6, 12, 13, 15, 14, 12, 16, 4… (Continúa como ruta 1) |
| 6 | 0, 1, 2, 4, 6, 12, 13, 15, 14, 12, 16, 4… (Continúa como ruta 2) |

1. **Prueba de bucle:**

Se realiza la prueba de bucle sobre este fragmento de código:

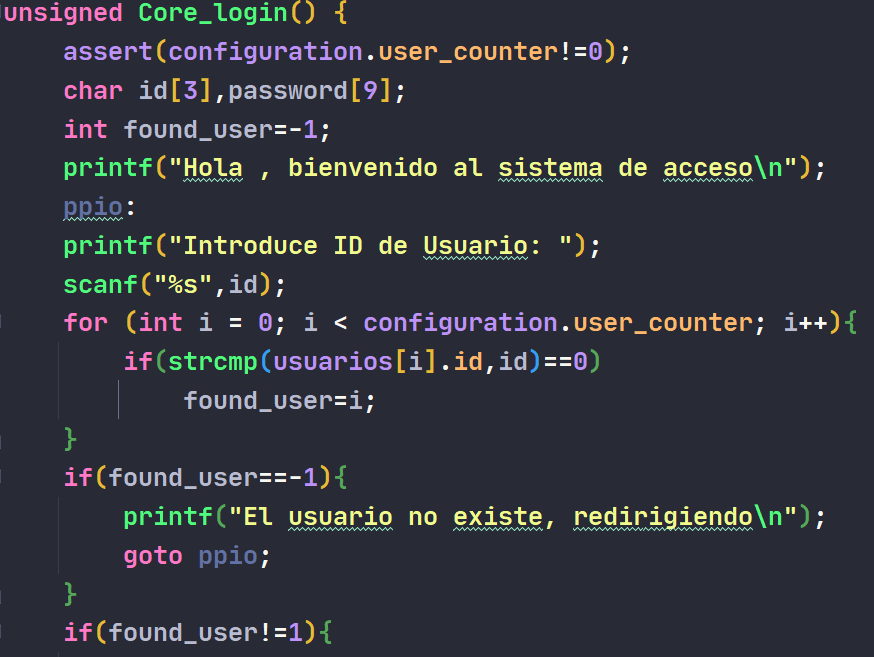


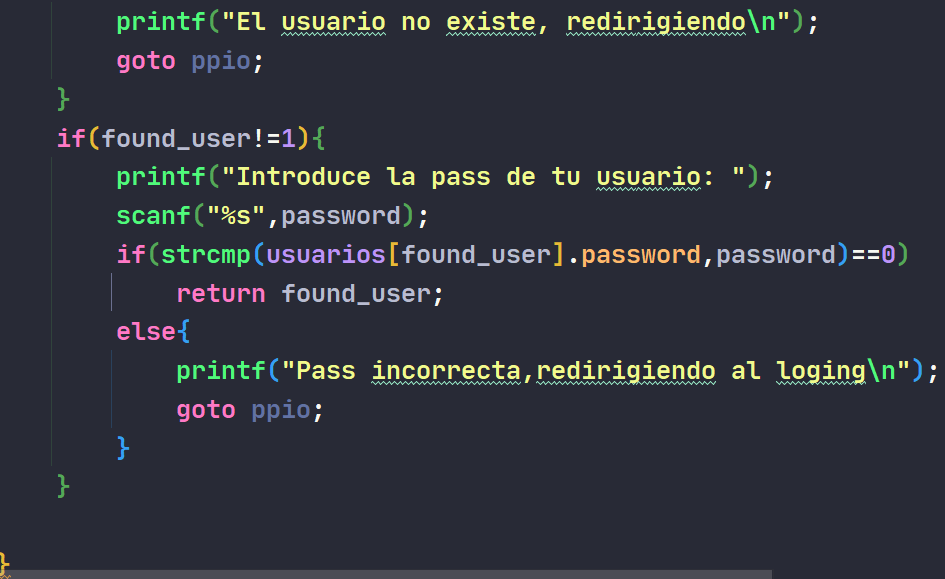
|  |  |  |  |
| --- | --- | --- | --- |
| **Precondiciones** | **Acciones** | **Después de iterar** | **Deseado** |
| j = configuartion.planter\_counter | …….. | j es igual a la variable sobre la que se itera | No entra en el bucle.  Ocurre el resultado esperado |
| j = 0 | j++  (Primera iteración)  Comprueba que la valoración está en el ranking | j = 1 | Imprime por pantalla los datos de la plantilla 1 |
| j = 1 | j++  (Segunda iteración)  Comprueba que la valoración está en el ranking | j = 2 | Imprime por pantalla los datos de la plantilla 2 |
| j = configuartion.planter\_counter- 1 | j++  (Última iteración)  Comprueba que la valoración está en el ranking | j = configuartion.planter\_counter | Imprime los datos de la última plantilla |
| j = configuartion.planter\_counter+ 1 | …….. | j = configuartion.planter\_counter + 1 | No entra en el bucle. |

**3. Módulo participante:**

**3.1 Prueba de caja negra:**

Para esta prueba, se usará la función ¨core\_login¨:





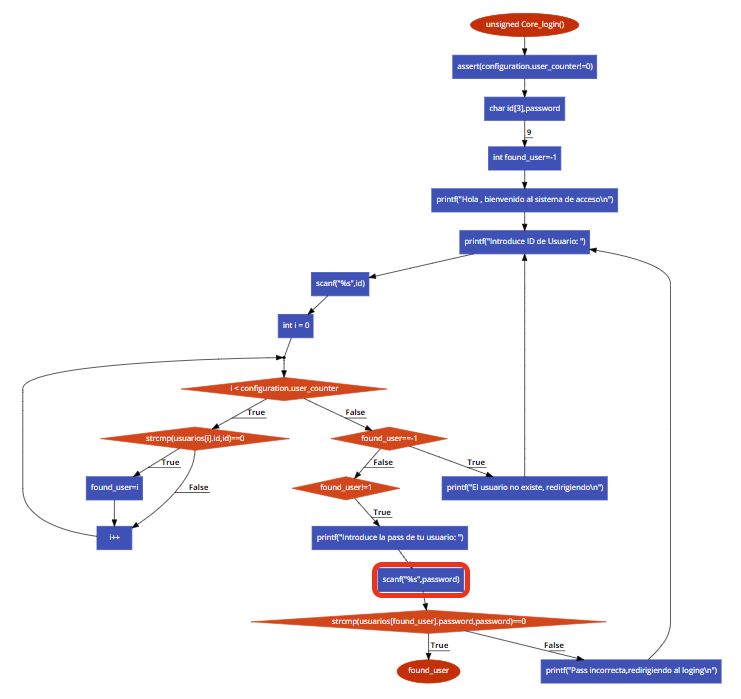
|  |  |
| --- | --- |
| **ENTRADA** | **SALIDA** |
| Se introduce un id que existe y su respectiva contraseña | La función devuelve el usuario encontrado |
| Se introduce un id que no existe | Imprime por pantalla que dicho usuario no existe y reinicia la consulta |
| Se introduce id que existe pero la contraseña es incorrecta | Imprime por pantalla la restricción de acceso a esa cuenta y reinicia la consulta |

La función realiza sus operaciones como es debido, sin dejar ningún tipo de vulnerabilidad al acceso de cualquier usuario sin el conocimiento de su código identificador y su contraseña.

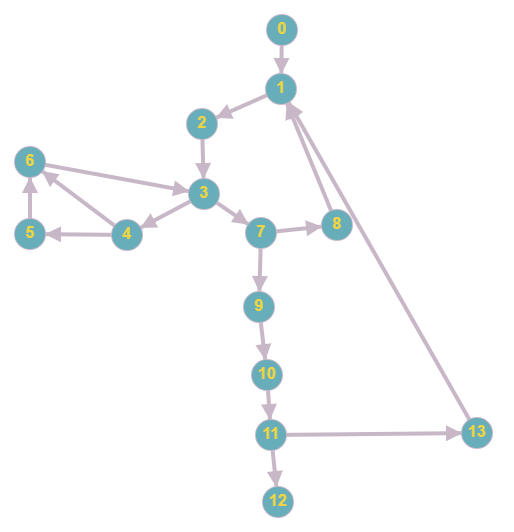
**3.2 Pruebas de caja blanca:**

1. **Prueba de ruta básica:**

Para esta prueba, se usará la función ¨core\_login¨, cuyo flujo es el siguiente:



Su grafo simplificado sería el siguiente:



Al calcular su complejidad ciclomática da los siguientes resultados:

V(G) = NNP + 1 = 5

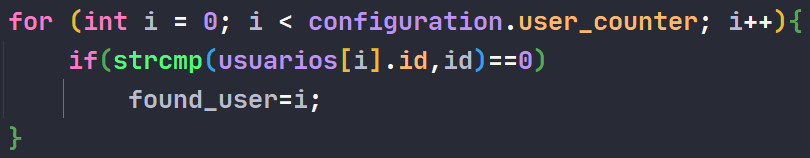
V(G) = NA - NN + 2 = 5

Al conocerse el número de rutas independientes, se mostrarán en la siguiente tabla:

|  |  |
| --- | --- |
| **RUTA BÁSICA** | **ORDEN** |
| 1 | 0, 1, 2, 3, 4, 5, 6, 3, 7, 9, 10, 11, 12 |
| 2 | 0, 1, 2, 3, 7, 9, 10, 11, 13, 1… (Continúa como la ruta 1) |
| 3 | 0, 1, 2, 3, 7, 8, 1... (Continúa como la ruta 1) |
| 4 | 0, 1, 2, 3, 4, 6, 3, 7, 9, 10, 11, 12 |
| 5 | 0, 1, 2, 3, 7, 9, 10, 11, 12 |

1. **Prueba de bucle:**

Se realiza la prueba de bucle sobre este fragmento de código:



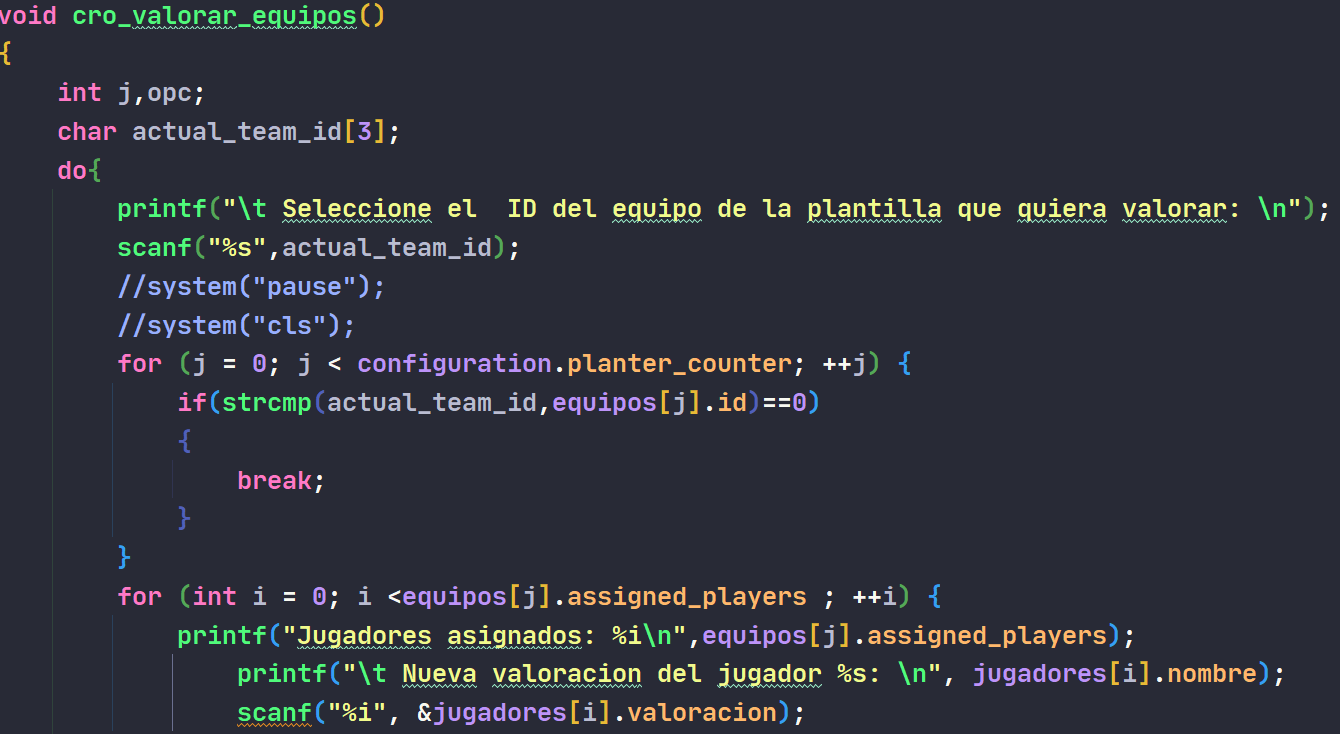
|  |  |  |  |
| --- | --- | --- | --- |
| **Precondiciones** | **Acciones** | **Después de iterar** | **Deseado** |
| i = configuartion.user\_counter | …….. | i es igual a la variable sobre la que se itera | No entra en el bucle.  Ocurre el resultado esperado |
| i = 0 | i++  (Primera iteración)  Comprueba si el id escrito existe | i = 1 | Si son iguales, recoge dicho  valor en ¨found\_user¨ |
| i = 1 | i++  (Segunda iteración)  Comprueba si el id escrito existe | i = 2 | Si son iguales, recoge dicho  valor en ¨found\_user¨ |
| i = configuartion.user\_counter- 1 | i++  (Última iteración)  Comprueba si el id escrito existe | i = configuartion.user\_counter | Si son iguales, recoge dicho  valor en ¨found\_user¨ |
| i = configuartion.user\_counter+ 1 | …….. | i = configuartion.user\_counter + 1 | No entra en el bucle. |

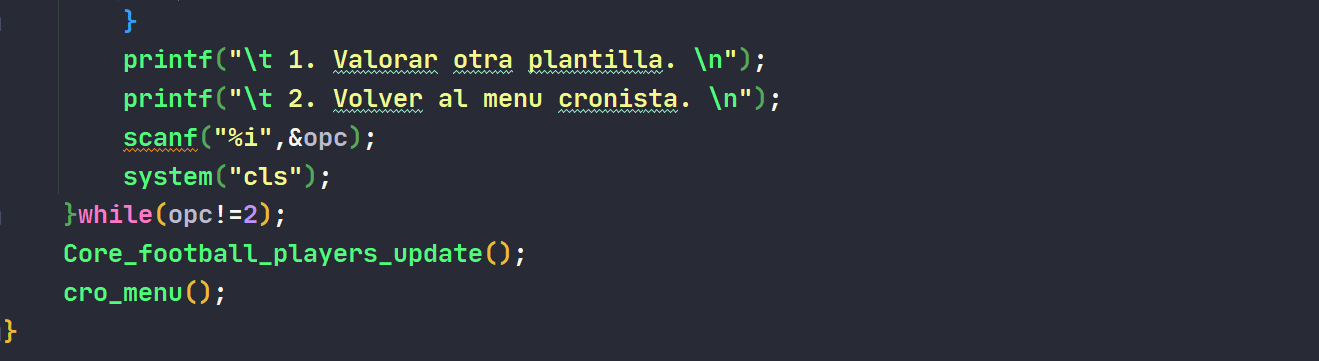
Como se puede observar, en cada iteración se realiza la misma operación, ya que al recorrer todos los id’s existentes, tan solo podrá ser uno el que coincida (o no) con el id introducido.

**4. Módulo cronista:**

**4.1 Prueba de caja negra:**

Para esta prueba, se usará la función ¨cro\_valorar\_equipos¨:





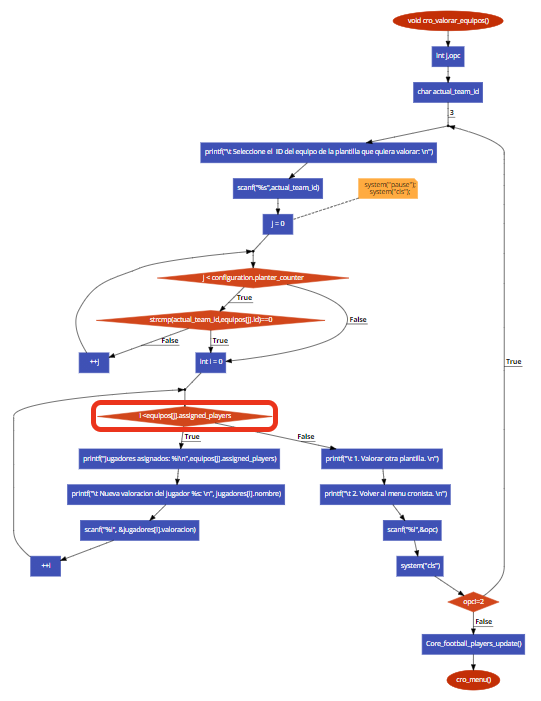
|  |  |
| --- | --- |
| **ENTRADA** | **SALIDA** |
| Se introduce un id de plantilla que existe | Te permite asignar valoraciones nuevas a jugadores dentro de dicha plantilla |
| Se introduce un id que no existe | Imprime por pantalla si desea valorar otra plantilla o volver al menú de cronista |
| Se introduce que se vuelva a valorar plantilla | La función vuelve a ejecutarse a través de un bucle do\_while |
| Se introduce que se quiere volver al menú | La función retorna al menú de cronista saliendo del bucle do\_while |

Como se puede observar, la función cumple con las expectativas de su funcionamiento esperado, siempre y cuando se ejecute desde el menú de cronista, siendo el usuario que lo ejecute uno de ellos.

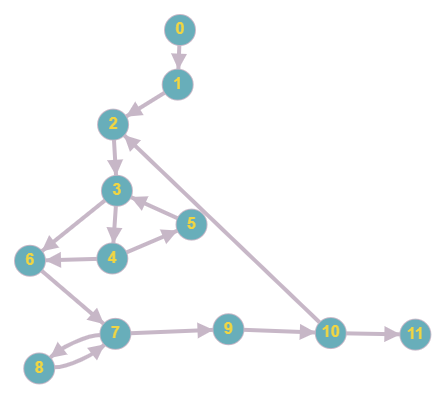
**3.2 Pruebas de caja blanca:**

1. **Prueba de ruta básica:**

Para esta prueba de ruta básica, se usará la función ¨cro\_valorar\_equipos¨, cuyo flujo sería el siguiente:



Su grafo simplificado sería el siguiente:



Al calcular su complejidad ciclomática da los siguientes resultados:

V(G) = NNP + 1 = 5

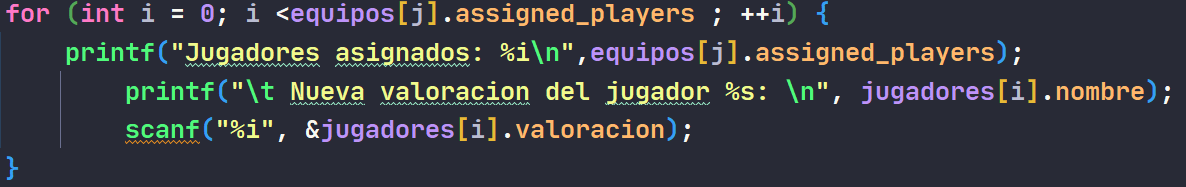
V(G) = NA - NN + 2 = 5

Al conocerse el número de rutas independientes, se mostrarán en la siguiente tabla:

|  |  |
| --- | --- |
| **RUTA BÁSICA** | **ORDEN** |
| 1 | 0, 1, 2, 3, 6, 7, 9, 10, 11 |
| 2 | 0, 1, 2, 3, 4, 5, 3, 6, 7, 8, 7, 9, 10, 11 |
| 3 | 0, 1, 2, 3, 6, 7, 9, 10, 2… (Sigue como la ruta 1) |
| 4 | 0, 1, 2, 3, 6, 7, 8, 7, 9, 10, 11 |
| 5 | 0, 1, 2, 3, 4, 6, 7, 9, 10, 11 |

1. **Prueba de bucle:**

Se realiza la prueba de bucle sobre este fragmento de código:



|  |  |  |  |
| --- | --- | --- | --- |
| **Precondiciones** | **Acciones** | **Después de iterar** | **Deseado** |
| i = equipos[j].assigned\_players | …….. | i es igual a la variable sobre la que se itera | No entra en el bucle.  Ocurre el resultado esperado |
| i = 0 | i++  (Primera iteración) | i = 1 | Asigna valoración nueva al jugador 1 |
| i = 1 | i++  (Segunda iteración) | i = 2 | Asigna valoración nueva al jugador 2 |
| i = equipos[j].assigned\_players- 1 | i++  (Última iteración) | i = equipos[j].assigned\_players | Asigna valoración nueva al último jugador de la plantilla |
| i = equipos[j].assigned\_players+ 1 | …….. | i = equipos[j].assigned\_players + 1 | No entra en el bucle. |