Universidad de Vigo

Grado en Ingeniería de
Tecnologías de Telecomunicación

# Definición y Especificación del Proyecto
# Gestión del Juego Master Mind

Programación I

Curso 2015-2016

# Tabla de Contenidos

# Definición y Especificación del Proyecto
# Gestión del Juego Master Mind

## 1. Introducción

El Proyecto tiene como objetivo que el alumno lleve a la práctica los **conceptos teóricos** de la asignatura.

El programa del Proyecto se tiene que desarrollar en el marco de la programación **estructurada**, la programación **modular** y el **buen estilo de programación**. Para actuar dentro de ese marco, es importante que el Proyecto posea una adecuada **estructura** de **módulos** -funciones en el lenguaje **C**- y que los algoritmos que implementan estas funciones se realicen usando las **estructuras de control** de **C**. Estos algoritmos deben optimizar la transformación de los datos que trata cada función construyendo estructuras de control lo más **sencillas** y **simples** posible, ayudándose con el ya citado buen estilo de programación: sangrados, adecuados identificadores de variables, constantes y funciones, comentarios, etc.

Como todo programa, el Proyecto tiene que proporcionar unos resultados que concuerden con los esperados, que son los establecidos en las especificaciones del problema objeto del Proyecto. Esta concordancia no se logra de forma **inmediata** y **sin esfuerzo**, salvo en problemas pequeños y sencillos. Por ello, en general hay que recurrir a un proceso reiterado de **prueba** y **depuración** hasta lograr que los resultados del programa concuerden con los esperados (para programas grandes y complejos, el coste de la prueba/depuración puede representar más del **60%** del coste total).

Lo anterior nos lleva a insistir en la idea de que, para minimizar el proceso de **prueba/depuración,** los programas deben realizarse bajo la óptica de la programación estructurada y modular.

Otro aspecto importante es el concepto de Documentación del Proyecto. Tiene que contener toda la **información** generada en el **desarrollo** (diagramas de flujo y algoritmos), en la **codificación** (listado del programa fuente en **C**) y en la **prueba/depuración**.

El Proyecto elegido para lograr los objetivos anteriores es un **Gestor del Juego Master Mind**[1] (al que, en adelante, denominaremos **MasterMind**).

[1] Si no conoce el juego, en internet puede encontrar muchas versiones del mismo, que le permitirán hacerse una idea de lo que debe hacer su programa.
Por ejemplo, en http://www.web-games-online.com/mastermind/index.php puede ver una implementación gráfica (con colores, en lugar de números).

## 2. Definición del Proyecto MasterMind

El objetivo de este proyecto es el desarrollo de una aplicación denominada MasterMind que permita jugar al juego del Master Mind y gestionar las partidas jugadas.

La aplicación que se va a diseñar debe permitir al usuario jugar partidas del citado juego, y guardar un historial de partidas en un fichero en disco. Así mismo, debe permitirle realizar operaciones como listar las partidas previamente jugadas y establecer el nivel de dificultad del juego.

Se proponen dos versiones de este proyecto:

a) Proyecto MasterMind **básico**. Esta versión es **obligatoria**: todos los grupos de trabajo han de implementarla.

b) Proyecto MasterMind **avanzado**. Esta versión, con funcionalidad mejorada, es **opcional**: los grupos de trabajo que lo deseen, pueden implementarla.

Obviamente, con la versión básica se puede aprobar el proyecto, pero sólo con la versión avanzada se puede conseguir la máxima calificación.

## 3. Mecánica del juego

El programa mantiene un fichero, llamado “partidas.txt”, donde va a ir guardando el historial de las partidas jugadas por el usuario.

Cada vez que se juega una partida, el programa genera, de manera aleatoria, un código secreto de 4 dígitos diferentes, que el usuario debe descubrir.

En primer lugar, guarda el código secreto en el citado fichero de historial.

A continuación, y de manera repetitiva, el programa invita al usuario a introducir una apuesta, es decir, a descubrir el código secreto, mediante el mensaje:

```
Escriba un número de 4 dígitos diferentes + ENTER:
```

El programa comprueba que el usuario proporciona una apuesta **sintácticamente correcta**:

- formada por exactamente 4 caracteres,
- todos ellos son dígitos decimales y
- todos son diferentes entre sí.

De no ser así, el programa no la acepta, y vuelve a invitar al usuario a introducir su apuesta.

Cuando el usuario introduce una apuesta válida, el programa muestra la apuesta hecha por el usuario, y una respuesta, consistente en:

- Por cada dígito acertado, y colocado en la posición correcta, un símbolo ‘*’.
- Por cada dígito acertado, pero en una posición incorrecta, un símbolo ‘|’.
- Por cada dígito que no esté en el código secreto, un símbolo ‘_’.

Por ejemplo, una posible salida podría ser:

```
Escriba un número de 4 dígitos diferentes + ENTER: 3456
Su apuesta es: 3456  <||__>
```

Después de mostrar esa salida, el programa guarda en el fichero de historial la apuesta del usuario (sólo aquéllas para las que ha mostrado una salida, es decir, las que sean sintácticamente correctas) y la respuesta del programa.

Este proceso se repite hasta que ocurre una de dos circunstancias:

- el usuario introduce una apuesta correcta, es decir descubre el código secreto, o
- se alcanza el máximo número admitido de intentos; por defecto, este número es 100, pero este valor va a ser configurable por el usuario.

En cualquiera de los dos casos, alcanzado el final del proceso, el programa guarda en el fichero de historial el número de intentos, y una puntuación:

- `10`, si se ha acertado el código en 3 o menos intentos.
- `0`, si no se ha acertado el código, o se han empleado 12 o más intentos.
- `10 – 10 * (intentos – 3) / 9`, en cualquier otro caso.

Tras eso, se muestra por pantalla un mensaje informando del éxito o fracaso del juego:

```
Ha descubierto el código secreto (3980) en 7 intentos
Ha obtenido 5.56 puntos
```

o:

```
NO ha descubierto el código secreto (5728) tras 100 intentos
Ha obtenido 0.00 puntos
```

Con esto, se termina la partida.

# 4. Especificación del Proyecto MasterMind básico

## 4.1. Funcionalidades del programa

El programa proporciona cinco funcionalidades al usuario que lo ejecuta:

- Jugar partida,
- Jugar partida de prueba,
- Establecer nivel de dificultad,
- Listar historial de partidas y
- Salir.

## 4.2. Carátula del programa

Cuando se ejecuta el programa, se debe mostrar por la pantalla la siguiente carátula:

```
**************************************************************************
**************************************************************************
**************************************************************************
*******************************  MASTER MIND  ****************************
**************************************************************************
**************************************************************************
**************************************************************************
```

## 4.3. Interfaz de usuario

A continuación, el programa debe mostrar al usuario el siguiente **menú principal** con las cinco funcionalidades de la especificación del apartado 4.1:

```
1) Jugar partida
2) Jugar partida de prueba
3) Establecer nivel de dificultad
4) Listar historial de partidas
0) Salir

¿Siguiente operación?
```

Si del menú se selecciona ‘1’, se juega una partida “normal” (ver apartado 4.6). Si se selecciona ‘2’, se juega una partida de prueba (ver apartado 4.6). Si se selecciona ‘3’, se va a establecer el nivel de dificultad del juego (ver apartado 4.7). Si se selecciona ‘4’, se lista el historial de partidas (ver apartado 4.8). Si se selecciona ‘0’, se sale del programa (ver apartado 4.9).

***Robustez del programa desde el punto de vista del usuario.***

En general, un programa se dice que es robusto si responde de forma adecuada ante cualquier circunstancia. El menú del programa debe ser robusto frente a los caracteres introducidos por el usuario.

Si el usuario selecciona los caracteres '1', '2', '3', '4' o '0', el programa debe responder ejecutando la funcionalidad correspondiente.

En cambio, si proporciona cualquier otra entrada, debe responder mostrando el mensaje "Has realizado una selección no válida", y volviendo a mostrar el menú. Este comportamiento debe reiterarse mientras no se seleccione '1', '2', '3', '4' o '0'.

Este criterio de robustez debe aplicarse a todos los menús del Proyecto.

A continuación se muestra un ejemplo de funcionamiento:

```
1) Jugar partida
2) Jugar partida de prueba
3) Establecer nivel de dificultad
4) Listar historial de partidas
0) Salir

¿Siguiente operación? 30

Has realizado una selección no válida

1) Jugar partida
2) Jugar partida de prueba
3) Establecer nivel de dificultad
4) Listar historial de partidas
0) Salir

¿Siguiente operación?
```

## 4.4. Vuelta al menú principal del programa

Cuando se termina la ejecución de las funcionalidades **1**, **2**, **3** o **4** del menú principal, el programa debe volver a dicho menú.

## 4.5. Fichero de historial de partidas jugadas: *partidas.txt*

El fichero de texto **partidas.txt** contiene toda la información relevante sobre las partidas jugadas previamente. La información de cada partida se almacena en el fichero de historial en grupos de 4 líneas más una línea por cada intento del usuario que contuviera una apuesta sintácticamente correcta[2], con el siguiente formato:

**Separador**
**Código Secreto**
**Apuesta 1  <Respuesta 1>**
**...**
**Apuesta n  <Respuesta n>**
**Número de Intentos**
**Puntuación**

La siguiente tabla describe los valores de las líneas de cada partida:

| Línea | Valores |
|---|---|
| **Separador** | "----------" (12 guiones) |
| **Código secreto** | Cadena de 4 dígitos decimales diferentes |
| **Apuesta i** | Cadena de 4 dígitos decimales diferentes |
| **Respuesta i** | Cadena de 4 caracteres del conjunto {'*', '\|', '_'} |
| **Número de intentos** | Entero entre 1 y 100 |
| **Puntuación** | Real entre 0 y 10 con 2 decimales |

El formato de las líneas que contienen las apuestas y las respuestas es exactamente:

```
Apuesta i  <Respuesta i>
```

Es decir:

- *Apuesta i*
- 2 espacios en blanco
- El carácter '<'
- *Respuesta i*
- El carácter '>'

[2] Es decir, las apuestas que no sean sintácticamente correctas **no** se escriben en el fichero.

Por ejemplo, el contenido del fichero de historial, tras jugar dos partidas, podría ser:

```
------------
2478
1234  <||__>
5678  <**__>
1278  <**|_>
3456  <*___>
1378  <**__>
2478  <****>
6
6.67
------------
3980
1234  <|___>
5678  <|___>
1590  <*|__>
2690  <*|__>
3790  <**|_>
3890  <**||>
3980  <****>
7
5.56
```

Si, al arrancar el programa, no existe el fichero de historial, éste será creado cuando se empiece a guardar la primera partida que se juegue.

## 4.6. Funcionalidad “Jugar partida”

Cuando se selecciona la opción **1** o la opción **2** del menú, se procede a jugar una partida. La diferencia entre ambas opciones es que:

- Con la opción **1**, se juega una partida en modo “**normal**”, lo que quiere decir que el programa creará un código **secreto** aleatorio de 4 dígitos decimales, sin repetición.

- Con la opción **2**, se juega una partida en modo “**de prueba**”, lo que quiere decir que el programa creará un código **predefinido**, con el valor “1234”.
  Esta segunda funcionalidad sirve para probar la interfaz de usuario (sin tener que perder tiempo en "descubrir" el código generado), y para poder pasarle al programa una batería automatizada de pruebas, si se desea, de la forma:

```
usuario@maquina:dir$ ./programa < fichero_de_pruebas.txt
```

La función que lleva a cabo esta funcionalidad, distinguirá estas opciones mediante un **parámetro** que le indicará el **modo** de juego.

En cualquiera de los dos casos, el juego seguirá según lo descrito en el apartado 3.

Un ejemplo de desarrollo de una partida en modo normal podría ser el siguiente:

```
Escriba un número de 4 dígitos diferentes + ENTER: 1234
Su apuesta es: 1234  <|___>


Escriba un número de 4 dígitos diferentes + ENTER: 5678
Su apuesta es: 5678  <|___>


Escriba un número de 4 dígitos diferentes + ENTER: 1590
Su apuesta es: 1590  <*|__>


Escriba un número de 4 dígitos diferentes + ENTER: 2690
Su apuesta es: 2690  <*|__>


Escriba un número de 4 dígitos diferentes + ENTER: 45678

La longitud de la apuesta es incorrecta

Escriba un número de 4 dígitos diferentes + ENTER: 25

La longitud de la apuesta es incorrecta

Escriba un número de 4 dígitos diferentes + ENTER: 56t8

La apuesta sólo debe contener dígitos

Escriba un número de 4 dígitos diferentes + ENTER: 5668

Los dígitos de la apuesta deben ser diferentes

Escriba un número de 4 dígitos diferentes + ENTER: 3790
Su apuesta es: 3790  <**|_>


Escriba un número de 4 dígitos diferentes + ENTER: 3890
Su apuesta es: 3890  <**||>


Escriba un número de 4 dígitos diferentes + ENTER: 3980
Su apuesta es: 3980  <****>


Ha descubierto el código secreto (3980) en 7 intentos
Ha obtenido 5.56 puntos
```

Como se puede apreciar en el ejemplo, esta funcionalidad puede dar tres tipos de aviso al usuario:

- Si introduce una apuesta que no tenga exactamente 4 caracteres, le mostrará el mensaje: “La longitud de la apuesta es incorrecta".
- Si la longitud de la apuesta es correcta, pero contiene algún carácter que no sea un dígito decimal, le mostrará el mensaje: “La apuesta sólo debe contener dígitos".
- Si los 4 caracteres son dígitos decimales, pero alguno está repetido, le mostrará el mensaje: “Los dígitos de la apuesta deben ser diferentes”.

En cualquiera de los tres casos, tras mostrar el mensaje correspondiente y descartar la apuesta como sintácticamente incorrecta, volverá a invitar al usuario a introducir una apuesta.

En el ejemplo, también se puede apreciar que el programa sólo proporciona información al usuario acerca de **cuántos** de los dígitos proporcionados se han “acertado”, pero no de **cuáles** son esos dígitos acertados. Podemos ver que, con un código de “3980”:

- Cuando el usuario introduce “1234”, donde hay un dígito acertado, aunque fuera de posición (el ‘3’), la respuesta del programa es “|___”, indicando ese hecho. Pero no da información acerca de que dicho dígito es el ‘3’.

- Cuando el usuario introduce “3790”, donde hay dos dígitos acertados y colocados en la posición correcta (el ‘3’ y el ‘0’), y un dígito acertado, aunque fuera de posición (el ‘9’), la respuesta del programa es “**|_”, indicando ese hecho. Pero no da información acerca de cuáles son esos dígitos (salvo por casualidad).

## 4.7. Funcionalidad “Establecer nivel de dificultad”

Cuando se selecciona la opción **3** del menú, se pasa a establecer un nuevo nivel de dificultad, es decir, un nuevo número máximo de intentos permitidos, antes de dar la partida por perdida al jugador.

Por defecto, al arrancar el programa, dicho nivel de dificultad es de 100, lo que quiere decir que, como máximo, se permitirá que el usuario haga 100 intentos (con apuestas sintácticamente correctas) de descubrir el código secreto.

Para realizar esta funcionalidad, se le mostrará al usuario el nivel actual, y se le pedirá que introduzca un nuevo valor entre 1 y 100. Si el usuario introduce un valor que no sea un número entero entre 1 y 100, el programa le informará de que la entrada no es válida, mediante el mensaje “Nivel seleccionado no válido”, y le volverá a pedir que introduzca un valor.

Este proceso seguirá hasta que el usuario introduzca un valor correcto. Finalmente, se mostrará el nuevo nivel seleccionado.

Un ejemplo de la ejecución de esta funcionalidad podría ser el siguiente:

```
Nivel actual: 100
Elige nuevo nivel [1..100]: p
Nivel seleccionado no válido
Elige nuevo nivel [1..100]: 300
Nivel seleccionado no válido
Elige nuevo nivel [1..100]: 56

Nivel seleccionado: 56
```

## 4.8. Funcionalidad “Listar historial de partidas”

Cuando se selecciona la opción **4** del menú, se procede a mostrar por pantalla el contenido del fichero de historial de partidas jugadas “partidas.txt”.

Un ejemplo de la ejecución de esta funcionalidad sería similar al ejemplo mostrado en el apartado 4.5.

Si el fichero de historial aún no existe (porque aún no se ha jugado ninguna partida), mostrará el mensaje: “No puedo leer el fichero de historial”.

## 4.9. Funcionalidad “Salir”

Cuando se selecciona la opción **0** del menú, se termina inmediatamente la ejecución del programa.

## 4.10. Modularidad

Se recomienda muy vivamente a los alumnos construir su programa alrededor de una estructura modular. Por ejemplo, podría estar formado por 3 módulos:

- El módulo principal (mastermind.c), consistente en el intérprete de comandos de la interfaz de usuario.
- El módulo de gestión del juego (gestion_juego.c), consistente en una librería de funciones (jugar_partida(), establecer_nivel() y listar_historial() ) y su correspondiente fichero de cabecera (gestion_juego.h).
- El módulo de gestión de la pantalla (pantalla.c), consistente en una librería de funciones (caracteres(), confirmar() y titulo() ) y su correspondiente fichero de cabecera (pantalla.h).

## 4.11. Lista de mensajes del programa

- Nombre del juego en la carátula: "MASTER MIND"
- Mensajes del menú principal:
  - "1) Jugar partida"
  - "2) Jugar partida de prueba"
  - "3) Establecer nivel de dificultad"
  - "4) Listar historial de partidas"
  - "0) Salir"
  - Invitación a introducir una operación: "¿Siguiente operación? "
  - Error de selección: "Has realizado una selección no válida"
- Mensajes de la función de jugar:
  - Invitación a jugar: "Escriba un número de 4 dígitos diferentes + ENTER: "
  - Error de longitud de la apuesta: "La longitud de la apuesta es incorrecta"
  - Error de contenido de la apuesta: "La apuesta sólo debe contener dígitos"
  - Error de dígitos repetidos: "Los dígitos de la apuesta deben ser diferentes"
  - Mensaje de salida: "Su apuesta es: XXXX  <RRRR>"
  - Mensaje de éxito: "Ha descubierto el código secreto en XX intentos"
  - Mensaje de fracaso: "NO ha descubierto el código secreto tras XX intentos"
    - En ambos casos, si XX es 1, dirá “intento” en lugar de “intentos”
  - Mensaje de puntuación: "Ha obtenido XX.XX puntos”
  - Error de acceso al fichero: "No puedo escribir en el fichero de historial"
- Mensajes de la función de establecer nivel:
  - Información de valor actual: "Nivel actual: XX"
  - Invitación a introducir nuevo valor:  "Elige nuevo nivel [1..100]: "
  - Error de valor: "Nivel seleccionado no válido"
  - Información de nuevo valor: “Nivel seleccionado: XX”
- Mensajes de la función de listar el historial:
  - Error de acceso al fichero: "No puedo leer el fichero de historial"

# 5. Especificación del Proyecto MasterMind avanzado

La implementación de las mejoras mostradas en esta versión es opcional. Los grupos de trabajo que lo deseen pueden introducir, a partir del Proyecto MasterMind básico, todas o algunas de las mejoras que se describen en esta sección.

Es decir, es obligatorio que el programa del Proyecto MasterMind avanzado sea una versión incremental del programa del Proyecto MasterMind básico, que también se debe implementar.

Las especificaciones de la versión avanzada son las siguientes.

## 5.1. Funcionalidades de la versión

El Proyecto MasterMind avanzado incorpora, junto a las cinco funcionalidades del Proyecto MasterMind básico, dos nuevas funcionalidades:

- Guardar partida jugada y
- Establecer idioma.

## 5.2. Interfaz de usuario

Una vez mostrada la carátula, como en la versión básica, el programa debe mostrar al usuario el siguiente **menú principal** con las cinco funcionalidades de la especificación del apartado 4.1, más las dos nuevas:

```
1) Jugar partida
2) Jugar partida de prueba
3) Establecer nivel de dificultad
4) Listar historial de partidas
5) Guardar partida jugada
6) Establecer idioma
0) Salir

¿Siguiente operación?
```

Aparte de lo ya especificado, si del menú se selecciona ‘5’, se guarda la última partida jugada (ver apartado 5.5). Y, si se selecciona ‘6’, se va a establecer el idioma del programa (ver apartado 5.6).

## 5.3. Cambios en la funcionalidad “Jugar partida”

En esta versión del programa hay dos cambios:

a) En vez de ir guardando la partida en el fichero de historial según se va jugando, simplemente la **registra** (se sugiere hacerlo en un **parámetro** tipo **struct** que la función recibe por referencia desde el intérprete de comandos de la interfaz de usuario); la decisión

de guardar esta partida en el fichero de historial la podrá tomar **después** el usuario, usando la funcionalidad "Guardar partida jugada".

b) Cuando se selecciona la opción **1** o la opción **2** del menú,

- Si no existe una partida jugada y sin guardar, se irá a jugar la partida en el modo seleccionado (tal y como se explica en el apartado 4.6, salvo la modificación relativa a no guardar la partida).
- Si existe una partida jugada y sin guardar, se mostrará el mensaje de invitación: "Hay una partida sin guardar, ¿descartarla? (s/n) ". Si el usuario introduce:
  - "S" o "s", se **descartará** esa partida, es decir, no se guardará en el fichero de historial, y se irá a jugar una nueva partida.
  - "N" o "n", se volverá al **menú principal**, es decir, **no** se irá a jugar una nueva partida, aunque tampoco se guardará la partida en el fichero de historial.
  - Cualquier otra entrada, se volverá a mostrar el mensaje de invitación, hasta que el usuario introduzca una entrada correcta.

## 5.4. Cambios en la funcionalidad "Salir"

Ahora, cuando se selecciona la opción **0** del menú:

- Si no existe una partida jugada y sin guardar, se terminará inmediatamente la ejecución del programa.
- Si existe una partida jugada y sin guardar, se mostrará el mensaje de invitación: "Hay una partida sin guardar, ¿está seguro de que desea salir? (s/n) ". Si el usuario introduce:
  - "S" o "s", se **descartará** esa partida, es decir, no se guardará en el fichero de historial, y se terminará inmediatamente la ejecución del programa.
  - "N" o "n", se volverá al **menú principal**, es decir, **no** se terminará la ejecución del programa, aunque tampoco se guardará la partida en el fichero de historial.
  - Cualquier otra entrada, se volverá a mostrar el mensaje de invitación, hasta que el usuario introduzca una entrada correcta.

## 5.5. Nueva funcionalidad “Guardar partida jugada”

Si hay una partida jugada y sin guardar en el fichero de historial, el menú principal queda tal y como se muestra en el apartado 5.2, y, entonces, cuando se selecciona la opción **5** del menú, se guarda dicha partida en el fichero de historial, tras lo que se vuelve al menú principal.

Si no hay tal partida jugada y sin guardar, no se muestra esta opción en el menú principal, que quedaría como sigue:

```
1) Jugar partida
2) Jugar partida de prueba
3) Establecer nivel de dificultad
4) Listar historial de partidas
6) Establecer idioma
0) Salir

¿Siguiente operación?
```

En estas condiciones, obviamente, no se acepta un **5** como entrada correcta, resultando en el mismo mensaje que para cualquier entrada no válida: “Has realizado una selección no válida”, y en la vuelta al menú principal.

## 5.6. Nueva funcionalidad “Establecer idioma”

Cuando se selecciona la opción **6** del menú, el programa pasa a invitar al usuario a introducir una entrada para elegir el idioma del programa, mostrándole el siguiente submenú:

```
0) Español
1) Gallego
2) Inglés

Elige idioma:
```

Si el usuario proporciona cualquier entrada que no sea ‘0’, ‘1’, o ‘2’, el programa debe responder con el mensaje “Idioma seleccionado no válido”, y reiterar el submenú mientras no se seleccione una entrada correcta:

```
0) Español
1) Gallego
2) Inglés

Elige idioma: 5

Idioma seleccionado no válido

0) Español
1) Gallego
2) Inglés

Elige idioma:
```

En cambio, si el usuario selecciona uno de los caracteres ‘0’, ‘1’, o ‘2’, el programa debe pasar a mostrar todos sus mensajes en el idioma elegido:

```
0) Español
1) Gallego
2) Inglés

Elige idioma: 2


1) Play game
2) Play test game
3) Set skill level
4) List game history
6) Set language
0) Quit

Next command? 1


Type a number with 4 different digits + ENTER: 1234
Your guess: 1234  <*___>


Type a number with 4 different digits + ENTER:
```

Esto, naturalmente, obliga a pasarle un parámetro que indique cuál es el idioma actual a todas aquellas funciones que muestren algún mensaje por pantalla, para que lo muestren en el idioma establecido.

## 5.7. Argumentos por línea de comando

La versión avanzada del proyecto admitirá que, al invocar el programa, se le pasen argumentos en la línea de comandos que modifiquen su comportamiento.

Concretamente, la sintaxis de invocación del programa será la siguiente:

```
usuario@maquina:dir$ ./programa [ --help | código_prueba ]
```

Es decir, el programa podrá ser invocado: o **sin** argumentos o con **un** argumento.

Si se invoca sin argumentos, actuará como hasta aquí (es decir, en el modo de prueba, el usuario deberá "descubrir" un código **predefinido**, con el valor "`1234`").

Si es invocado con un argumento, éste puede ser:

- Exactamente la cadena "`--help`", lo que producirá el comportamiento descrito en la sección 5.7.1
- O un código de prueba, lo que hará que el programa actúe según se describe en 5.7.2

Si se invoca con más de un argumento, informará de este hecho, y saldrá inmediatamente:

```
usuario@maquina:dir$ ./programa 2468 3456

Uso: ./programa [ --help | código_prueba ]
```

### 5.7.1. Petición de ayuda por línea de comando

El usuario podrá utilizar la versión avanzada del proyecto con el único objeto de recibir un mensaje de ayuda. En este caso, el programa se invocará de la siguiente manera, que producirá la salida mostrada:

```
usuario@maquina:dir$ ./programa --help

Uso: ./programa [ --help | código_prueba ]

--help: este mensaje de ayuda
código_prueba: un código de 4 dígitos decimales diferentes para pruebas

El programa MASTER MIND permite las siguientes funcionalidades:

1) Jugar partida: descubrir un código aleatorio de 4 dígitos decimales diferentes
2) Jugar partida de prueba: "descubrir" el código pasado para pruebas
3) Establecer nivel de dificultad: elegir el máximo número de intentos admitidos
4) Listar historial de partidas: mostrar las jugadas anteriormente realizadas
5) Guardar partida jugada: guardar en un fichero de historial la última partida
6) Establecer idioma: elegir en qué idioma se quiere recibir la información
```

### 5.7.2. Código de prueba seleccionable en línea de comando

En la versión avanzada del proyecto, existe la opción de que el código que se debe "descubrir" en el modo de prueba no sea un código **predefinido**, con el valor "1234" (o sea, una constante), sino un código pasado al programa como un argumento en línea de comandos. Por ejemplo, si se desea que dicho código sea "2468", se invocará el programa de la siguiente manera:

```
usuario@maquina:dir$ ./programa 2468
```

Si el código no tiene exactamente 4 caracteres o, teniendo 4, alguno de ellos no es un dígito decimal, o, siendo todos dígitos decimales, alguno está repetido, informará de este hecho, y saldrá inmediatamente:

```
usuario@maquina:dir$ ./programa 12345

La longitud del código de prueba es incorrecta

usuario@maquina:dir$ ./programa 123p

El código de prueba sólo debe contener dígitos

usuario@maquina:dir$ ./programa 1223

Los dígitos del código de prueba deben ser diferentes
```

## 5.8. Mejoras que puede contener la versión avanzada

Por lo que hemos visto hasta aquí, la versión avanzada del proyecto puede tener hasta 3 mejoras, de las cuales, se pueden implementar 1, 2 o las 3:

- La funcionalidad "Guardar partida jugada", y los cambios asociados a ella en las funcionalidades "Jugar partida" y "Salir".
- La funcionalidad "Establecer idioma", y los cambios asociados a ella en todas las funcionalidades que muestren información por pantalla.
- La opción de pasar argumentos al programa por línea de comando, y los cambios asociados a ella en la funcionalidad "Jugar partida"

## 5.9. Lista de mensajes del programa

### 5.9.1. Versión en español

- Mensajes de la línea de comandos:
  - Mal número de argumentos: “Uso: *nombre_programa* [ --help | código_prueba ]”
    - Si el programa ha sido invocado como “./MasterMind”, el mensaje sería: “Uso: ./MasterMind [ --help | código_prueba ]”
  - Error de longitud del código de prueba: "La longitud del código de prueba es incorrecta"
  - Error de contenido del código de prueba: "El código de prueba sólo debe contener dígitos"
  - Error de dígitos repetidos en el código de prueba: "Los dígitos del código de prueba deben ser diferentes"
  - Mensaje de ayuda:

    “Uso: *nombre_programa* [ --help | código_prueba ]

    --help: este mensaje de ayuda
    código_prueba: un código de 4 dígitos decimales diferentes para pruebas

    El programa MASTER MIND permite las siguientes funcionalidades:

    1) Jugar partida: descubrir un código aleatorio de 4 dígitos decimales diferentes
    2) Jugar partida de prueba: “descubrir” el código pasado para pruebas
    3) Establecer nivel de dificultad: elegir el máximo número de intentos admitidos
    4) Listar historial de partidas: mostrar las jugadas anteriormente realizadas
    5) Guardar partida jugada: guardar en un fichero de historial la última partida
    6) Establecer idioma: elegir en qué idioma se quiere recibir la información”

    - Si el programa ha sido invocado como “./MasterMind”, la primera línea sería: “Uso: ./MasterMind [ --help | código_prueba ]”

- Nombre del juego en la carátula: "MASTER MIND"
- Mensajes del menú principal:
  - "1) Jugar partida"
  - "2) Jugar partida de prueba"
  - "3) Establecer nivel de dificultad"
  - "4) Listar historial de partidas"
  - “5) Guardar partida jugada"
  - “6) Establecer idioma"
  - "0) Salir"
  - Invitación a introducir una operación: "¿Siguiente operación? "
  - Error de selección: "Has realizado una selección no válida"
  - Aviso antes de jugar: "Hay una partida sin guardar, ¿descartarla? (s/n) "
  - Aviso antes de salir: "Hay una partida sin guardar, ¿está seguro de que desea salir? (s/n) "
- Mensajes de la función de jugar:
  - Invitación a jugar: "Escriba un número de 4 dígitos diferentes + ENTER: "
  - Error de longitud de la apuesta: "La longitud de la apuesta es incorrecta"
  - Error de contenido de la apuesta: "La apuesta sólo debe contener dígitos"
  - Error de dígitos repetidos: "Los dígitos de la apuesta deben ser diferentes"
  - Mensaje de salida: "Su apuesta es: XXXX  <RRRR>"

    - Mensaje de éxito: "Ha descubierto el código secreto en XX intentos"
    - Mensaje de fracaso: "NO ha descubierto el código secreto tras XX intentos"
        - En ambos casos, si XX es 1, dirá “intento” en lugar de “intentos”
    - Mensaje de puntuación: "Ha obtenido XX.XX puntos”
- Mensajes de la función de establecer nivel:
    - Información de valor actual: "Nivel actual: XX"
    - Invitación a introducir nuevo valor:  "Elige nuevo nivel [1..100]: "
    - Error de valor: "Nivel seleccionado no válido"
    - Información de nuevo valor: “Nivel seleccionado: XX”
- Mensajes de la función de listar el historial:
    - Error de acceso al fichero: "No puedo leer el fichero de historial"
- Mensajes de la función de guardar el historial:
    - Error de acceso al fichero: "No puedo escribir en el fichero de historial"
- Mensajes de la función de establecer idioma:
    - “0) Español”
    - “1) Gallego”
    - “2) Inglés”
    - Invitación a introducir nuevo idioma: "Elige idioma: "
    - Error al seleccionar idioma: "Idioma seleccionado no válido"

### 5.9.2. Versión en gallego

- Nombre del juego en la carátula: "MASTER MIND"
- Mensajes del menú principal:
  - "1) Xogar partida"
  - "2) Xogar partida de proba"
  - "3) Establecer nivel de dificultade"
  - "4) Listar historial de partidas"
  - "5) Gardar partida xogada"
  - "6) Establecer idioma"
  - "0) Saír"
  - Invitación a introducir una operación: "Seguinte operación? "
  - Error de selección: "Realizaches unha selección non válida "
  - Aviso antes de jugar: "Hai unha partida sen gardar, descartala? (s/n) "
  - Aviso antes de salir: "Hai unha partida sen gardar, está seguro de que desexa saír? (s/n) "
- Mensajes de la función de jugar:
  - Invitación a jugar: "Escriba un número de 4 díxitos diferentes + ENTER: "
  - Error de longitud de la apuesta: "A lonxitude da aposta non é correcta"
  - Error de contenido de la apuesta: "A aposta só debe conter díxitos"
  - Error de dígitos repetidos: "Os díxitos da apuesta deben ser diferentes"
  - Mensaje de salida: "A súa aposta é: XXXX  <RRRR>"
  - Mensaje de éxito: "Descubriu o código secreto en XX intentos"
  - Mensaje de fracaso: "NON descubriu o código secreto tras XX intentos"
    - En ambos casos, si XX es 1, dirá “intento” en lugar de “intentos”
  - Mensaje de puntuación: "Obtivo XX.XX puntos”
- Mensajes de la función de establecer nivel:
  - Información de valor actual: "Nivel actual: XX"
  - Invitación a introducir nuevo valor:  "Elixe novo nivel [1..100]: "
  - Error de valor: "Nivel seleccionado non válido"
  - Información de nuevo valor: “Nivel seleccionado: XX”
- Mensajes de la función de listar el historial:
  - Error de acceso al fichero: "Non podo ler o ficheiro de historial"
- Mensajes de la función de guardar el historial:
  - Error de acceso al fichero: " Non podo escribir no ficheiro de historial "
- Mensajes de la función de establecer idioma:
  - “0) Español”
  - “1) Galego”
  - “2) Inglés”
  - Invitación a introducir nuevo idioma: "Elixe idioma: "
  - Error al seleccionar idioma: "Idioma seleccionado non válido"

### 5.9.3. Versión en inglés

- Nombre del juego en la carátula: "MASTER MIND"
- Mensajes del menú principal:
  - "1) Play game"
  - "2) Play test game"
  - "3) Set skill level "
  - "4) List game history"
  - “5) Save played game "
  - “6) Set language "
  - "0) Quit"
  - Invitación a introducir una operación: “Next command? "
  - Error de selección: " You have made an invalid selection"
  - Aviso antes de jugar: "There is an unsaved game, discard it? (y/n) "
  - Aviso antes de salir: "There is an unsaved game, are you sure you want to quit? (y/n) "
- Mensajes de la función de jugar:
  - Invitación a jugar: "Type a number with 4 different digits + ENTER: "
  - Error de longitud de la apuesta: "You have not typed 4 digits"
  - Error de contenido de la apuesta: "The guess can only contain digits"
  - Error de dígitos repetidos: "The guess digits must be different"
  - Mensaje de salida: "Your guess: XXXX  <RRRR>"
  - Mensaje de éxito: "You have discovered the secret code in XX attempts"
  - Mensaje de fracaso: "You have FAILED to discover the secret code after XX attempts"
    - En ambos casos, si XX es 1, dirá “attempt” en lugar de “attempts”
  - Mensaje de puntuación: "You have got XX.XX points”
- Mensajes de la función de establecer nivel:
  - Información de valor actual: "Current level: XX"
  - Invitación a introducir nuevo valor:  "Select new level [1..100]: "
  - Error de valor: "You have selected an invalid level"
  - Información de nuevo valor: “Selected level: XX”
- Mensajes de la función de listar el historial:
  - Error de acceso al fichero: "History file can't be read"
- Mensajes de la función de guardar el historial:
  - Error de acceso al fichero: "History file can't be written"
- Mensajes de la función de establecer idioma:
  - “0) Spanish”
  - “1) Galician”
  - “2) English”
  - Invitación a introducir nuevo idioma: "Select language: "
  - Error al seleccionar idioma: "Selected language is not allowed"

# 6. Evaluación del proyecto

### *Plazo de entrega*

El código del proyecto deberá subirse a Faitic no más tarde del viernes **13 de mayo** de 2016.

### *Condiciones de la entrega*

**Todos** los grupos de trabajo deberán subir **todos** los ficheros con código C ("***.c**") y con cabeceras ("***.h**") en los que hayan organizado la versión **básica** de su proyecto, y el correspondiente ***Makefile*** para generar el ejecutable de la aplicación.

Adicionalmente, aquellos grupos que hayan realizado todas o alguna de las mejoras de la versión avanzada, ***también*** deberán subir **todos** los ficheros con código C ("***.c**") y con cabeceras ("***.h**") en los que hayan organizado dicha versión **avanzada** de su proyecto, y el correspondiente ***Makefile*** para generar el ejecutable de la aplicación.

Es responsabilidad de cada grupo de trabajo asegurarse de que el código subido:

1. se compila sin errores, y
2. tiene un comportamiento conforme a lo especificado en este documento.

Con el objeto de facilitar lo segundo, se publicará en *Faitic* un protocolo de pruebas (es decir, una batería de pruebas, y los resultados que debe producir cada una de dichas pruebas), de tal manera que cada grupo de trabajo pueda evaluar de manera autónoma[3] el correcto funcionamiento de su programa.

### *Forma de evaluación*

Tal y como se especifica en la Guía Docente de la asignatura, el proyecto será evaluado mediante la ***Prueba Final Práctica***.

En dicha Prueba, se solicitará al alumno/a que realice una o varias sencillas modificaciones de su código. Es decir, se realizará un pequeño cambio en la especificación del proyecto, y se pedirá al alumno/a que adapte su código a dicha nueva especificación.

El código así modificado deberá ser capaz de pasar satisfactoriamente un nuevo protocolo de pruebas que compruebe su conformidad a la nueva especificación. En función del grado de cumplimiento de este protocolo de pruebas, el alumno/a obtendrá una nota, que será:

- hasta un 70% de la nota máxima de la Prueba Final Práctica, si el grupo del que forma parte sólo ha entregado la versión básica del proyecto,
- hasta un 80% de dicha nota máxima, si el grupo ha entregado una de las mejoras resumidas en la sección 5.8,
- hasta un 90% de dicha nota máxima, si el grupo ha entregado dos de dichas mejoras, y
- hasta el 100% de dicha nota máxima, si el grupo ha entregado las tres mejoras.

[3] Obviamente, contando con todo el asesoramiento necesario por parte de los profesores de la asignatura, tanto en el laboratorio, durante las sesiones de clase práctica, como fuera del mismo, en horario de tutorías.