



PRÁCTICA FUNDAMENTOS DE PROGRAMACIÓN

CURSO: 2023/24

HALINA CWIERZ LÓPEZ
CENTRO UNIVERSITARIO DE MÉRIDA. UNIVERSIDAD DE EXTREMADURA

Índice

1. Práctica Propuesta Fundamentos de Programación 2023/24	6
1.1. Introducción y descripción	6
1.2. Requisitos mínimos del código	9
1.3. Plagio	9
1.4. Mejoras	10
1.5. Recomendaciones	10
1.6. Nomenclatura del programa	10
1.7. Bibliografía	10

1. Práctica Propuesta Fundamentos de Programación 2023/24

1.1. Introducción y descripción

Puede parecer sencillo o muy complicado aprender a programar, en la web puedes encontrar mucha información y tutoriales (unos mejores y otros peores) que intentan enseñar los elementos básicos de un lenguaje de programación y cómo hacer uso de estos elementos para desarrollar tu software. Lo verdaderamente complicado y que se adquiere tras mucha práctica, es programar de forma estructurada, precisa y rigurosa.

En este curso, la práctica consistirá en implementar una adaptación del juego de la **Oca** tradicional, usando el lenguaje de programación C/C++ [1, 2, 3] y el entorno de desarrollo integrado (IDE) Eclipse¹. Puedes encontrar documentación sobre el juego en los siguientes enlaces:

<https://www.juegodelaoca.com/Reglamento/reglamento.htm>

https://es.wikipedia.org/wiki/Juego_de_la_oca



Figura 1. Tablero Oca tradicional

Reglas del juego de la OcaFP

Se necesita un mínimo de dos jugadores. Se recomienda a partir de los 8 años y hasta un máximo de 99 años. Cada jugador participará con una ficha de diferente (color, letra o número). El objetivo del juego es llegar el primero al final del tablero, es decir, alcanzar a la Gran Oca. Se deben respetar los turnos para lanzar los dados. El juego es el único que decide cuándo esta regla puede ser incumplida. Las casillas del tablero están todas numeradas, y algunas pueden tener un significado especial, que se describen a continuación:

- **Oca:** casillas 5, 9, 14, 18, 23, 27, 32, 36, 41, 45, 50, 54 y 59. Si se cae en una de estas casillas, se puede avanzar hasta la siguiente casilla en la que hay una oca y volver a tirar. ***“De oca a oca y tiro porque me toca”***.
- **Puente:** casilla 6 y 12. Si se cae en una de estas casillas hay que avanzar o retroceder hasta la otra casilla Puente y volver a tirar. ***“De puente a puente y tiro porque me lleva la corriente”***.

¹ <https://www.eclipse.org/downloads/packages/release/2021-06/r/eclipse-ide-cc-developers>

- **PoSada**: casilla 19. Si se cae en esta casilla se pierde un turno.
- **PoZo**: casilla 31. Si se cae en esta casilla, NO se puede volver a jugar hasta que otro jugador caiga en esa casilla o pasen tres turnos.
- **Laberinto**: casilla 42. Si se cae en esta casilla, se está obligado a retroceder a la casilla 30.
- **Cárcel**: casilla 56. Si se cae en esta casilla, hay que permanecer dos turnos sin jugar.
- **Dados**: casillas 26 y 53. Al caer en esta casilla, se avanza o se retrocede hasta la otra casilla dados y se vuelve a tirar.
- **CalaVera**: Casilla 58. Si se cae en esta casilla, hay que volver a empezar.
- **Entrar al Jardín de la Oca**: Es necesario sacar los puntos justos para entrar, en caso de exceso se retroceden tantas casillas como puntos sobrantes.

Objetivo del juego de la OcaFP

Ser el primero en llegar a la casilla final de la Gran Oca, saltando de posiciones, según la tirada de los dados y sometido a las Reglas del Juego, establecidas por cada casilla.

Tablero de la OcaFP

↓
columnas

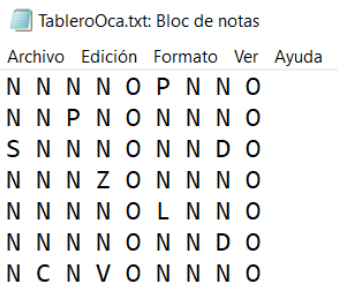
Tablero OcaFP (matriz de 7 filas x 9 columnas)

	0	1	2	3	4	5	6	7	8
filas → 0	Inicio 1 [0,0]	2	3	4	5	6	7	8	9
1	10 [1,0]	11	12	13	14	15	16	17	18
2	19	20	21	22	23	24	25	26	27
3	28	29	30	31	32	33	34	35	36
4	37	38	39	40	41	42	43	44	45
5	46	47	48	49	50	51	52	53	54
6	55	56	57	58	59	60	61	62	Final 63

Oca	Laberinto	[i,j]:fila i, col j	GREEN	CYAN
Puente	Cárcel		YELLOW	DARKGRAY
PoSada	Dados		MAGENTA	BLUE
PoZo	CalaVera		BROWN	RED

Los colores de las casillas especiales son a título orientativo, el estudiante podrá elegir qué colores utilizar o si decide usar letras o símbolos.

Las casillas especiales se leerán de un fichero de texto **"TableroOca.txt"**, cuyo contenido será el siguiente:



```

TableroOca.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
N N N N O P N N O
N N P N O N N N O
S N N N O N N D O
N N N Z O N N N O
N N N N O L N N O
N N N N O N N D O
N C N V O N N N O
  
```

N:no es una casilla especial.

¿Cómo se juega?

Para determinar el jugador que comenzará la partida, cada uno lanzará el dado y empezará el que obtenga la puntuación más alta, si hay empate, se volverá a lanzar tantas veces como sea necesario para poder decidir quién comienza el juego.

El jugador que haya obtenido la puntuación más alta comenzará lanzando el dado y se posicionará en la casilla contando desde el inicio tantos puntos como haya salido en el dado. Por ejemplo, si ha salido un 4, el jugador se posicionará en la casilla con valor 4.

A continuación lanzará el dado el segundo jugador y se repetirá la misma acción hasta que alguno alcance la casilla 63 que será el ganador de la partida (se deberá actualizar el fichero de Jugadores.txt cuando salga del programa).

Dependiendo de la casilla en la que caiga el jugador, se deberán realizar las acciones que se han descrito en el apartado Reglas del juego.

El estudiante deberá crear un fichero de jugadores **"Jugadores.txt"** en el que se recoja la siguiente información:

- Nombre del jugador: máximo 10 letras sin espacios.
- Número de veces que ha jugado.
- Número de veces que ha ganado.

Ejemplo del fichero de Jugadores.txt y significado de cada uno de los campos:

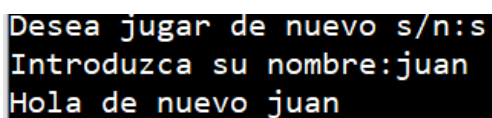
halina 15 5

Nombre del jugador: halina

Número de veces que ha jugado: 15

Ha ganado 5 veces.

Si el nombre del jugador existe en el fichero de Jugadores.txt, deberá mostrar un mensaje que indique que el jugador ya ha jugado otras veces.



```

Desea jugar de nuevo s/n:s
Introduzca su nombre:juan
Hola de nuevo juan
  
```

Si el nombre del jugador no existe en el fichero de Jugadores.txt, deberá mostrar otro mensaje dando la bienvenida al nuevo jugador.

```
Desea jugar de nuevo s/n:s
Introduzca su nombre:angel
Bienvenido angel este juego te enganchara
```

1.2. Requisitos mínimos del código

La práctica deberá cumplir los siguientes requisitos mínimos:

- El código debe ser eficiente, lo más modular posible y funcionar correctamente.
- Se debe poder ejecutar todas las veces que el usuario desee sin salir del programa.
- Debe controlar en la medida de lo posible, los errores que el usuario pueda cometer al introducir información.
- No se debe implementar ninguna de las prohibiciones expresamente expuestas en el campus virtual de la asignatura de FP (Figura 2).
- Se deben cumplir las buenas prácticas de programación.

En la asignatura de Fundamentos de Programación está **prohibido lo siguiente:**

- Usar variables globales.
- Tener más de **un return** en main() o en cualquier otra función o procedimiento.
- Usar **break** en cualquier lugar que no sea un switch.
- Salir de un **for** antes de su finalización.
- Usar **exit()** para finalizar el programa.

Buenas prácticas de programación

- En las funciones **sólo** debe haber parámetros por valor, **NO** se deben pasar parámetros por variable.
- Una función **NO** puede devolver un vector o matriz, ni un fichero.
- En los procedimientos, si existen parámetros por variable de tipo simple o struct, siempre deberán ser **más de uno**, si es sólo uno, implicará que es una función.
- Los vectores son direcciones de memoria al primer elemento. Cuando se pasan a los módulos como parámetros por variable **NO** se debe de poner **&** y si se pasan por valor, es recomendable usar la palabra reservada **const**.
- Para gestionar ficheros, se deben usar las funciones propias de ficheros (eof(), fail(), open(), close(),...)

Figura 2. FP. Prohibiciones y buenas prácticas de programación

1.3. Plagio

Según la RAE, plagiar es copiar en lo sustancial obras ajenas dándolas como propias [5].

Si se detectan prácticas copiadas, los estudiantes implicados en las copias tendrán automáticamente un suspenso en la asignatura.

1.4. Mejoras

Se valorará “*especialmente*” que la práctica sea lo más modular posible, código bien indentado y claro. Los nombres de: constantes, variables, tipos de datos y módulos deben hacer referencia a la función que realizan.

Todas las mejoras que el alumno realice se tendrán en cuenta para la nota final (colores, sonidos, uso del cursor, extras...)

1.5. Recomendaciones

Cada vez que se programe un fragmento de código o un módulo hay que probar su correcto funcionamiento, una vez comprobado y verificado, se pasará a programar el siguiente.

Es recomendable leer “**Consejos y Buenas Prácticas en Programación**” disponible en el siguiente enlace:

<https://personales.unican.es/sanchezbp/teaching/faqs/programming.html>

Nota: Esta práctica puede estar sujeta a cambios, que la profesora notificaría en clase o en el campus virtual de la asignatura.

1.6. Nomenclatura del programa

Nomenclatura de archivos: son las reglas para poner nombre a un archivo.

- No usar puntos en el nombre, se aconseja usarlos para separar el nombre del archivo de la extensión.
- No usar tildes.
- No usar espacios de separación, mejor guion bajo (_).
- Usar nombres descriptivos y cortos.

Nombre de la tarea:

PF23_24nombreestudiante+inicial1ºapellido+inicial2ºapellido.cpp

Ejemplo: PF23_24halinacl.cpp

1.7. Bibliografía

- [1] W. Savitch y R. L. a. G. R. Escalona García, Resolución de problemas con C++, Pearson Education, 2000.
- [2] F. García y J. Carretero, El lenguaje de Programación C :Diseño e Implementación de Programas, Madrid: Prentice-Hall, 2001.
- [3] J. M. Maestre Torreblanca y J. Relinque Pérez, A programar se aprende jugando, Madrid: Paraninfo, 2017.

- [4] U. d. Extremadura, «Campus virtual,» Universidad de Extremadura, [En línea]. Available: <https://campusvirtual.unex.es/portal/>. [Último acceso: 8 2021].
- [5] RAE, «Diccionario de la lengua española,» [En línea]. Available: <https://dle.rae.es/plagiar#TIZy4Xb>. [Último acceso: 8 2021].