
# Fundamentos de la Programación

## Ayuda de la Práctica 1 (Versión 3)

(Basado en la práctica de Mercedes Gómez, Luis Hernández, Ramón González y Federico Peinado)

# Índice

# 1. Introducción

- ✓ Con esta presentación se pretende ayudar en la implementación de la versión 3 de la práctica 1 cuyo resultado debes entregar a través del campus.
- ✓ Necesario: un mínimo de práctica con **estructuras** y **listas de longitud variable**.
- ✓ ¡Practica con los ejercicios del tema 5 si no lo has hecho ya!
- ✓ Esta ayuda es opcional. No es obligatorio seguirla para implementar la solución de la versión 3.

# 2. Planificación

- ✓ Para realizar la versión 3 en el tiempo estimado, aconsejamos cumplir la siguiente planificación:
  - 21/11: Proyecto, Modificaciones y JuegoModoD.
  - 28/11: ModoDHumano y ModoDMaquina
  - 5/12: EscribeArchivo y Pruebas Finales.
  - 9/12: Último Día de Entrega (23:55).

# 3. Proyecto y Modificaciones

- ✓ Crea un **nuevo proyecto** en Visual Studio para la versión 3.
- ✓ Copia el archivo *main.cpp* de la versión 2 en el directorio del nuevo proyecto de Visual Studio.
- ✓ Añade tu archivo *main.cpp* al proyecto desde el explorador de la solución (en la carpeta de fuentes).
- ✓ Pon un comentario en el archivo *main.cpp* indicando que se trata de la solución de la versión 3 de la práctica 1.
- ✓ No te olvides de añadir el comentario con el nombre y apellidos de los integrantes del grupo de laboratorios.

- ✓ Define el tipo *tConjuntoCartas* para listas de cartas con array y contador. Define una constante para el tamaño y un tipo array *tArrayCartas* para el array de la lista.
- ✓ Estas nuevas definiciones deben aparecer en *main.cpp* antes de los prototipos de las funciones.
- ✓ Modifica el programa *main.cpp* añadiendo los prototipos de las nuevas funciones para las listas de cartas:

```
void inicializa(tConjuntoCartas &mazo);
void sacarCarta(tConjuntoCartas &mazo, int &carta);
void annadirCarta(tConjuntoCartas &mazo, int carta);
void crearMazo(tConjuntoCartas &mazo);
```

- ✓ Añade, además, los siguientes prototipos para el modo D:

```
void juegoModoD(int numPartida); // Nueva función juego con las cartas en memoria en lugar de archivo

float modoDhumano(tConjuntoCartas &mazo, tCartasPorAparecer cartas, tConjuntoCartas &cartasHumano);

float modoDmaquina(tConjuntoCartas &mazo, tCartasPorAparecer cartas, tConjuntoCartas &cartasMaquina, float puntosHumano);

int determinaGanadorD(float puntosHumano, float puntosMaquina, int numCartasHumano, int numCartasMaquina);

void escribeArchivo(int numPartida, int gana, float puntosHumano, float puntosMaquina, tConjuntoCartas cartasHumano, tConjuntoCartas cartasMaquina);

void imprimeCartasArchivo(ofstream &outFile, tConjuntoCartas mazo);
```

- ✓ Prepara las funciones (sin código) después de la función *main*.
- ✓ En esta versión, **si es necesario** modificar la función *main()*. Ahora debes llamar a *juegoModoD()* si la opción es 4 y seguir llamando a *juego()* si la opción es 1, 2 ó 3.
- ✓ Modifica la función *menu()* incluyendo una opción 4 (Play Modo D).

# 4. Implementación de la Función CrearMazo

- ✓ Esta función recibe un mazo vacío, lo rellena con las 40 cartas y después baraja estas cartas de forma aleatoria.
- ✓ La siguientes implementación necesita la librería *algorithm*.

```
Rellenar el mazo con 40 cartas (pueden estar ordenadas)
// Visualizar mazo sin barajar
random_shuffle(&(mazo.cards[0]), &(mazo.cards[NumCards]));
// Visualizar mazo tras barajar
```

# 5. Implementación de Funciones del Mazo

- ✓ Estas funciones permiten sacar una carta del mazo y añadir una carta al mazo.
- ✓ Para sacar una carta del mazo, el contador debe decrementarse en 1.
- ✓ Para añadir una carta al mazo, el contador debe incrementarse en 1.

# 6. Implementación de la Función JuegoModoD

```
Inicializar playerScore, machineScore y las cartas por aparecer
Inicializar el mazo, cartasHumano y cartasMaquina
crearMazo(mazo)
Mostrar "Player turn"
Ejecutar modoDhumano guardando playerScore
Si (playerScore > 7.5)
      Mostrar "Machine wins!"
      gana = 2;
Sino
      Mostrar "Machine turn"
      Ejecutar modoDmaquina guardando machineScore
      Si (machineScore > 7.5)
            Mostrar "Player wins!"
            gana = 1;
      Sino
         Ejecutar determinaGanadorD guardando gana
escribeArchivo(numPartida, gana, playerScore, machineScore,
cartasHumano, cartasMaquina);
```

# 7. Implementación de la Función ModoDHumano

```
Inicializar variables
Mientras (mazo.counter != 0), no se haya pasado y no haya stop
    sacarCartaMazo(mazo, carta);
    annadirCarta(cartasHumano, carta);
    Si el valor de la carta > 7
       Sumar 0.5 a la puntuación
       Actualizar el array de cartas quitando la carta
    Sino
       Sumar la carta a la puntuación
       Actualizar el array de cartas quitando la carta
    Mostrar cartasHumano y la puntuación actualizada
    Si (puntuación > 7.5)
         se ha pasado
    Sino
         Si (mazo.counter != 0)
              Preguntar si desea plantarse
              Actualizar stop según la respuesta
Devolver la puntuación
```

# 8. Implementación de la Función ModoDMaquina

```
Inicializar variables
Mientras (mazo.counter != 0), no se haya pasado y no haya stop
    sacarCartaMazo(mazo, carta);
    annadirCarta(cartasMaquina, carta);
    Si el valor de la carta > 7
       Sumar 0.5 a la puntuación
       Actualizar el array de cartas quitando la carta
    Sino
       Sumar la carta a la puntuación
       Actualizar el array de cartas quitando la carta
    Mostrar cartasMaquina y la puntuación actualizada
    Si (puntuación > 7.5)
         se ha pasado
    Sino
         Si (mazo.counter != 0)
              stop = (puntuación == 7.5) o (puntuación > puntuación
                 del humano) o ((puntuación == puntuación del humano)
                 y esProbablePasarse(puntuacion, cartas))
Devolver la puntuación
```

# 9. Implementación de la Función DeterminaGanadorD

```
Si puntosHumano > puntosMaquina
   ganador = 1
Sino si puntosHumano < puntosMaquina
   ganador = 2
Sino
   Si numCartasHumano == numCartasMaquina
      Sacar un número aleatorio entre 1 y 2 para determinar el ganador
   Sino
      Si numCartasHumano < numCartasMaquina
         ganador = 1
      Sino
         ganador = 2
Devolver ganador
```

# 10. Implementación de la Función EscribeArchivo

```
Declarar flujo de salida outFile
outFile.open(to_string(numPartida) + ".txt");
Escribir en outFile el número de partida
Escribir en outFile el ganador
Escribir en outFile "Resultados humano:“
Escribir en outFile la puntuación del humano
imprimeCartasArchivo(outFile, cartasHumano);
Escribir en outFile "Resultados maquina:“
Escribir en outFile la puntuación de la máquina
imprimeCartasArchivo(outFile, cartasMaquina);
Cerrar outFile
```