Fundamentos de la Programación I

Ayuda de la Práctica (Versión 3)

(Basado en la práctica de Luis Garmendia, Ramón González del Campo, Pablo Rabanal y Luis Hernández)

Índice

1. Introducción
2. Planificación
3. Proyecto, Tipos y Modificaciones
4. Nuevas Funciones del Juego
5. Ampliaciones en Main y Ficheros

1. Introducción

- ✓ Con esta presentación se pretende ayudar en la implementación de la versión 3 de la práctica.
- ✓ Necesario: un mínimo de práctica con **structs** y **listas**.
- ✓ ¡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.
- ✓ La entrega de esta versión de la práctica es el dia **22 de diciembre**.

2. Planificación

- ✓ Para realizar la versión 3, debes seguir la siguiente planificación de laboratorios:
  - 4/12: Proyecto, Tipos y Modificaciones.
  - 11/12: Nuevas Funciones del Juego.
  - 18/12: Ampliaciones en Main y Ficheros.

3. Proyecto, Tipos 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.
- ✓ No te olvides de añadir el comentario con el nombre y apellidos de los integrantes del grupo de laboratorios.

- ✓ Define las constantes del programa:

```
const int NumFichas = 28;
const int MinJugadores = 2, MaxJugadores = 4;
```

- ✓ Define los siguientes tipos para las fichas y las listas de fichas:

```
typedef struct {
   short int izquierda, derecha;
} tFicha;

typedef tFicha tFichaArray[NumFichas];

typedef struct {
   tFichaArray fichas;
   int contador;
} tListaFichas;
```

- ✓ Define los siguientes tipos para los jugadores y sus puntos:

```
typedef tListaFichas tJugadores[MaxJugadores];
typedef int tPuntosJugadores[MaxJugadores];
```

- ✓ Define el siguiente tipo para toda la información del juego:

```
typedef struct {
   string tablero;
   tListaFichas pozo;
   tJugadores jugadores;
   tPuntosJugadores puntos;
   int numJugadores;
} tJuego;
```

- ✓ Recuerda que las constantes y los tipos deben aparecer antes de los prototipos de las funciones.

- ✓ Modifica los prototipos de las siguientes funciones de la versión 2 para usar la nueva definición de tipos:

```
void generaPozo(tListaFichas &pozo, int maxDig);
void desordenaPozo(tListaFichas &pozo);
string fichaToStr(tFicha ficha);
void mostrarTablero(const tJuego &juego);
bool robaFicha(tListaFichas &pozo, tFicha &ficha);
void eliminaFicha(tListaFichas &list, int indice);
bool puedePonerIzq(string tablero, tFicha ficha);
bool puedePonerDer(string tablero, tFicha ficha);
void ponerFichaIzq(string &tablero, tFicha ficha);
void ponerFichaDer(string &tablero, tFicha ficha);
bool puedeColocarFicha(const tListaFichas &jugador, string tablero);
int sumaPuntos(const tListaFichas &jugador);
```

- ✓ Modifica las implementaciones de todas estas funciones.

- ✓ En la función *generaPozo*, no te olvides de dar un valor al contador del pozo al final de la función.
- ✓ En la función *desordenaPozo*, sólo hace falta una variable *tmp* del tipo *tFicha*. Además, el bucle for comienza en *pozo.contador - 1*.
- ✓ Al comienzo de la función *mostrarTablero*, haz una llamada a *system("cls")* (para limpiar la pantalla) e incluye la librería *windows.h* en el programa.
- ✓ En esta función, las fichas de las máquinas (a partir del jugador 1) salen después del tablero y antes de las fichas del jugador humano.

- ✓ La nueva función *robaFicha* devuelve false si el contador del pozo es cero y, en caso contrario, devuelve la ficha de la ultima posición (*contador – 1*).
- ✓ La función *eliminaFicha* elimina la posición *indice* de la lista de fichas haciendo los desplazamientos a la izquierda, como en la versión 2.
- ✓ En las funciones *puedePonerIzq* y *puedePonerDer* se compara *ficha.izquierda* y *ficha.derecha* con el valor correspondiente.
- ✓ Las funciones *ponerFichaIzq* y *ponerFichaDer* se convierten en procedimientos pasando el *tablero* por referencia.
- ✓ El bucle while de la función *puedeColocarFicha* usa *jugador.contador* en una de sus condiciones. El bucle for de la función *sumaPuntos* finaliza en *jugador.contador* – 1.

- ✓ Declara las siguientes **nuevas** variables en la función *main*:

```
int jugador;
tJuego juego;
tFicha ficha;
```

- ✓ Las siguientes variables ya **no** serán necesarias:

```
int numColocadas = 0;
int numRobadas = 0;
boolean haRobado = false;
string tablero = "";
short int fichaN1, fichaN2;
tArray pozo1, pozo2, fichas1, fichas2;
short int pozoCont, fichasCont;
```

4. Nuevas Funciones del Juego

- ✓ Añade los siguientes prototipos de funciones nuevas:

```
int quienEmpieza(const tJuego &juego, int& indice);
void iniciar(tJuego &juego, int &jugador);
bool sinSalida(const tJuego &juego);
bool estrategia1(tJuego &juego, int jugador);
bool estrategia2(tJuego &juego, int jugador);
```

- ✓ Para implementar la función *quienEmpieza*, añade las siguientes funciones al programa:

```
bool operator==(tFicha opLeft, tFicha opRight);
bool contiene(tListaFichas fichas, tFicha ficha, int &indice);
```

- ✓ La función *operator==* devuelve *true* si las fichas que se pasan por parámetro son iguales.
- ✓ La función *contiene* devuelve *true* si la ficha que se pasa como parámetro está en la lista de fichas *fichas*. Además, devuelve el índice donde se ha encontrado. Para implementarla haz una búsqueda en la lista *fichas*.

- Para implementar la función *quienEmpieza*:

```
jugador = -1, p, dd = 6
mientras ((jugador < 0) && (dd >= 0)) {
    Poner dd en izquierda y derecha de una ficha
    p = 0
    mientras((p < juego.numJugadores) && (jugador p no contiene la
   ficha))
        p++
    si (p == juego.numJugadores)
        dd--
    sino
       jugador = p
Devolver jugador
```

- Esta función devuelve el jugador que empieza y el índice que devuelve la función *contiene* cuando se encuentra la ficha. Además, modifica el parámetro por referencia *juego*.

- ✓ Puedes usar el siguiente pseudocódigo para la función *iniciar*

```
mientras(!partidaIniciada)
    Generar pozo
    Desordenar pozo
    Robar 7 fichas para cada uno de los jugadores
    jugador = quienEmpieza(juego, indice)
    si (jugador >= 0) {
       Poner ficha de indice del jugador en el tablero
       Eliminar la ficha del jugador
       Mostrar un mensaje diciendo quien empieza
       partidaIniciada = true
    sino
       Mostrar un mensaje diciendo que nadie tiene dobles
jugador = (++jugador) % juego.numJugadores;
```

- ✓ Una vez implementada la función *iniciar*, ya puedes modificar el código de la función *main* para **probar tu código**.
- ✓ Dentro de *main*, elimina la inicialización del juego (antes del bucle principal) de la versión 2.
- ✓ Ahora debes pedir un valor para *juego.numJugadores*. Además debes inicializar la puntuación de todos los jugadores a 0 y llamar a la función *iniciar* para iniciar el juego.
- ✓ Esta función *iniciar* roba 7 fichas para cada jugador y coloca el primer doble en el tablero, si es que se puede.
- ✓ No te olvides de modificar todas las llamadas a las funciones según los nuevos prototipos de la versión 3 y prueba tu código con **un único jugador**, es decir, el jugador humano. Para ello pon la constante *minJugadores* a 1.

- ✓ La función *sinSalida* debe devolver *true* si ningún jugador puede colocar ninguna ficha y el contador del pozo es 0.
- ✓ Para ver si se pueden colocar las fichas necesitaras dos bucles. Uno exterior de los jugadores del juego y otro interno de las fichas del jugador.
- ✓ Realiza la búsqueda como en implementaciones anteriores.

- La *estrategia1* puede implementarse:

```
encontrado = false
fichaNum = 0
mientras (!encontrado && fichaNum < contador de fichas del jugador
     si puede poner fichaNum a la izquierda
          Poner fichaNum por la izquierda
          Eliminar fichaNum
          encontrado = true
     sino si puede poner fichaNum a la derecha
          Poner fichaNum por la derecha
          Eliminar fichaNum
          encontrado = true
     sino
          fichaNum++
devolver encontrado
```

- La *estrategia2* puede implementarse:

```
encontrado = false
posMejor = -1, puntosMejor = -1, puntos
Desde i = 0 hasta el contador de fichas del jugador -1
    Si puede colocar la ficha i por la izquierda o por la derecha
        puntos = suma de puntos de izquierda y derecha de ficha i
        si (jugador > puntosMejor)
            posMejor = I
            puntosMejor = puntos
Si (posMejor >= 0)
   enontrado = true
   Si puede poner por la izquierda la ficha de posMejor
      Poner ficha de posMejor por la izquierda
   Sino
      Poner ficha de posMejor por la derecha
   Eliminar ficha de posMejor del jugador
Devolver encontrado
```

5. Ampliaciones en Main y Ficheros

- La nueva función *main* debe implementar el siguiente pseudocódigo:

```
Inicializar el juego  (= elegir entre reanudar una partida guardada
en un archivo, o solicitar nº de jugadores, inicializar los puntos y
el juego)
Mientras no finalice el juego (jugar == true)
        jugar una ronda (puede acabar porque el humano la aborte,
porque alguien gana o porque no hay salida)
        si la ronda se abortó
              se puede guardar en archivo y jugar = false
        si no
              si hubo ganador → mostrar quién fue
              actualizar los puntos que lleva cada jugador
(acumulando los que suma ahora)
              mostrar los puntos acumulados que lleva cada jugador
              si se quiere jugar otra ronda → iniciar un nuevo juego
              si no → jugar =  false
```

- ✓ Introduce variables bool para ganar, abortar y jugar. Por ejemplo, *ganado*, *interrumpido* y *jugar*.
- ✓ Después de llamar a la función *iniciar* (que inicializa el juego) debes introducir el bucle exterior que repite las diversas rondas del juego. Para salir de este bucle usaremos *jugar = false*.
- ✓ El bucle interior (antiguo bucle principal) se ejecutará miestras *(opcion != 0) && !ganado*.
- ✓ Inicializa las variables *opcion*, *interrumpido* y *ganado* justo antes de este bucle interior.
- ✓ Ahora, este bucle interior debe implementar la jugada de una ronda completa en la que juegan el jugador humano y las máquinas. El pseudocódigo es el siguiente:

```
Mientras (opcion != 0) && !ganado
        Mostrar el tablero
        Si es el turno del humano (jugador == 0)
             Mostrar el menú y leer la opción
             Actuar según la opción (switch)
        Sino
             Ejecutar la estrategia de la máquina que tiene el turno
             Si la máquina ha podido colocar ficha
                  Si se ha quedado sin fichas, ganado = true
                  Sino, pasar al siguiente jugador
             Sino
                  Si no quedan fichas en el pozo
                       Pasar al siguiente jugador
                  Sino
                       Robar ficha
                       Añadir ficha al final de sus fichas
      Si no hay salida en el juego
           Mostrar el tablero
           opcion = 0
```

- Ahora en el switch de la actuación del jugador humano, haremos prácticamente lo mismo que en la version 2:

```
si la opción elegida es poner ficha a la izquierda
        solicitar una posición de ficha válida
        si se puede poner ficha a la izquierda
                poner ficha a la izquierda
                eliminar la ficha de la lista del jugador
                si se ha quedado sin fichas → ganado = true
                sino → pasar al siguiente jugador (jugador = 1)

        sino
                 mostrar mensaje de error
```

- De la misma manera implementaremos el case 2 (colocar ficha a la derecha).
- En el case 3, si no puede colocar ficha y el contador del pozo es 0, pasaremos al siguiente jugador (*jugador = 1*) .
- En el caso 0, indicaremos que el jugador ha abortado la ronda poniendo *interrumpido = true*.

- ✓ Para pasar al siguiente jugador en el turno de las máquinas, haremos:

```
jugador = (++jugador) % juego.numJugadores;
```

- ✓ Una vez implementada la nueva función *main*, **prueba tu código** y comprueba las actuaciones del jugador humano así como los turnos de las máquinas.
- ✓ Finalmente, añade los siguientes prototipos de funciones para la gestión de ficheros y prepara sus implementaciones al final del programa.

```
bool leerJuego(tJuego &juego);
void leerListaFichas(ifstream &entrada, tListaFichas &listaFichas);
void escribirJuego(const tJuego& juego);
void escribirListaFichas(ofstream &salida, const tListaFichas
   &listaFichas);
```

- ✓ La función *leerJuego* debe solicitar el nombre del fichero, abrirlo y cargar en su parámetro por referencia *juego* todos los datos del fichero.
- ✓ Cada línea de fichas se carga llamando a la función *leerListaFichas*.
- ✓ La función *leerListaFichas* utiliza un bucle for para leer los números de la izquierda y de la derecha de cada ficha del parámetro *listaFichas*.
- ✓ La función *escribirJuego* solicita el nombre del fichero, lo abre y escribe todos los datos de su parámetro *juego*. Para escribir cada lista de fichas utiliza la función *escribirListaFichas*.
- ✓ La función *escribirListaFichas* usa un bucle for para escribir los números de la izquierda y de la derecha de su parámetro *listaFichas*.

- ✓ Fíjate en el pseudocódigo de la función main (transparencia…) y utiliza las funciones *leerJuego* y *escribirJuego* donde corresponde.
- ✓ **Prueba tu código** y comprueba todas las situaciones posibles. Recuerda que la prueba funcional es una parte importante de la nota final de tu práctica.