Práctica. Segunda parte
Minero 2.0

Fecha de entrega: 15 de mayo a las 23:55

1. Eficiencia en el uso de la memoria

A pesar de que los equipos informáticos actuales disponen de recursos hardware razonablemente por encima de lo que generalmente necesitamos, es importante usar la memoria de forma eficiente y no desperdiciar ni replicar contenido. Por este motivo, vamos a ampliar la práctica del minero utilizando memoria dinámica y punteros, de esta forma la implementación será más eficiente.

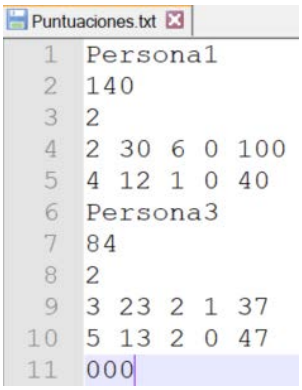
2. Modificaciones a realizar en la práctica del minero

Vamos a incorporar al juego del minero un módulo que nos permita puntuar las minas recorridas por cada usuario (jugador). La información relativa a los usuarios y las minas recorridas se almacenará en un fichero cuyos registros tendrán la siguiente estructura:

1. Nombre de usuario. Una cadena de caracteres sin blancos.
2. Puntuación total obtenida en todas las minas recorridas. Esta puntuación es la suma de la puntuación obtenida en cada mina recorrida.
3. Número de minas recorridas por el usuario. Máximo 5.
4. Para cada mina recorrida por el usuario se almacenan 5 valores en una línea:
    a. El identificador de la mina.
    b. Numero de movimientos que se han necesitado para alcanzar la salida.
    c. Número de gemas que se ha recogido en esa mina
    d. Número de dinamitas que se han necesitado para abrirse camino hasta la salida.
    e. Puntuación del minero al recorrer la mina. La puntuación se obtiene a partir de las dimensiones del plano, del número de movimientos realizados, el número de gemas recogidas y el número de dinamitas utilizado. Se aplicará la siguiente fórmula:

        Puntuación = Ancho del plano * alto del plano + A * número de gemas recogidas - número de movimientos - B * número de dinamitas utilizadas

        Donde A y B son dos constantes del programa con valores 10 y 2 respectivamente.



El fichero está **ordenado** por nombre de usuario y termina con una cadena de caracteres formada por tres ceros: 000.

Al comenzar el juego se carga del fichero toda la información de los usuarios en un **array dinámico**. A continuación se pide al usuario que se identifique con un nombre. Pueden darse dos casos: que el usuario ya estuviese dado de alta en el marcador o no.

a) Si el usuario ya estaba dado de alta en el sistema se le muestra un listado con las minas que ya tiene recorridas, ordenadas por nivel, y los valores obtenidos al recorrerlas.

```
SEGUNDA PARTE DE LA PRÁCTICA DEL MINERO

                 Introduce tu nombre de jugador/a: Persona1

         Ya estás registrado/a.

          Mira las minas que has recorrido ordenadas por nivel.

  Persona1  Movimientos   Gemas  Dinamitas Puntos      Puntos en total
  Mina  2        30          6       0       100                 140
  Mina  4        12          1       0        40

 Persona1, ¿Qué mina quieres explorar?.
 Introduce un número entre 1 y 5 para explorar una mina y 0 para salir
0
```

Figura 1

b) Si el usuario no está dado de alta en el sistema se le dará de alta en ese momento y se le muestra las puntuaciones de los demás usuarios ordenados por orden alfabético

```
SEGUNDA PARTE DE LA PRÁCTICA DEL MINERO

        Introduce tu nombre de jugador/a: Persona2


        Eres nuevo/a:Persona2

        Mira las puntuaciones de los otros jugadores.

           ------------   LISTA DE JUGADORES  -----------------

                               Persona1 140
                               Persona2 0
                               Persona3 84

Presione una tecla para continuar . . .

 Persona2, ¿Qué mina quieres explorar?.
 Introduce un número entre 1 y 5 para explorar una mina y 0 para salir

2
```

Figura 2

En ambos casos, a) y b), se le pregunta al usuario qué mina quiere recorrer o si quiere salir. Puede seleccionar cualquiera de las 5 minas, pudiendo ser desconocida (no recorrida) o conocida (ya recorrida). Una mina no ha sido recorrida si sus movimientos son cero. Puede escoger una mina ya recorrida para intentar obtener mejor puntuación. Tanto si el usuario elige

recorrer una mina nueva o una mina conocida, comenzará a recorrerla según la práctica ya realizada. Si termina de recorrer la mina con éxito se actualizarán los datos del recorrido en el array dinámico y se mostrarán los datos de todas las minas. Se volverá a preguntar si quiere recorrer una mina o salir. Si termina sin éxito, se mantiene la puntuación anterior, si la tenía.

```
             -----   JUGADORES ORDENADOS POR NOMBRE -----

  Persona1  Movimientos   Gemas  Dinamitas Puntos      Puntos en total
  Mina  2        30          6       0       100                 140
  Mina  4        12          1       0        40

  Persona2  Movimientos   Gemas  Dinamitas Puntos      Puntos en total
  Mina  2        30          6       0       100                 100

  Persona3  Movimientos   Gemas  Dinamitas Puntos      Puntos en total
  Mina  3        23          2       1        37                  84
  Mina  5        13          2       0        47

Presione una tecla para continuar . . .

        Persona2, ¿Qué mina quieres explorar?.
         Introduce un número entre 1 y 5 para explorar una mina y 0 para salir
```

Figura 3

Cuando el usuario abandona definitivamente el juego se guardan los datos del array dinámico en el fichero de puntuaciones.

3. Implementación del módulo Puntuaciones

Definimos un tipo estructurado tPuntuacionJugador que contenga una cadena de caracteres que representa el nombre del usuario, un entero con la puntuación total obtenida, y un array estático con los datos de cada recorrido de la mina. Las minas están numeradas del 1 al 5. La información de cada mina se guarda en su correspondiente componente del vector. Si la mina correspondiente a un índice del vector no está recorrida, la componente del vector debe tener los valores a cero. Como máximo 5 minas.

```
typedef struct {
       int IdMina;
int numMovimientos;
int numGemas;
       int numDinamitas;
       int puntosMina;
} tDatosMina;

typedef struct {
string nombre;
int punt_total;
       int minasRecorridas;
       tDatosMina vMinasRecorridas[NUM_TOTAL_MINAS];
} tPuntuacionJugador;
```

Para almacenar el contenido del fichero se declara un array dinámico de `tPuntuacionJugador`. Este array es dinámico porque no conocemos el número de usuarios

cuando los cargamos desde el fichero. Inicialmente el array tendrá tamaño 2, y duplicará su tamaño cada vez que se necesite. Por lo tanto, el array irá tomando tamaños 2, 4, 8, …

```
typedef struct{
 int capacidad;
 int num_jugadores;
 tPuntuacionJugador *array_clasificacion;
} tPuntuaciones;
```

También se deben incorporar en el correspondiente módulo al menos las siguientes funciones:

- `bool cargar_Marcador(tPuntuaciones& marcador)`: Introduce en el array dinámico los datos disponibles en el fichero `Puntuaciones.txt`.
- `bool guardar_Marcador(tPuntuaciones& marcador)`: Vuelca el contenido del array dinámico en el fichero `Puntuaciones.txt`.
- `void mostrar_Minas_Usuario (const tPuntuaciones & marcador, , int cont)`: Lista las minas recorridas por una persona, ordenadas por nivel (Figura 1).
- `void mostrar_Alfabetico(const tPuntuaciones & marcador)`: Lista los jugadores y sus puntuaciones totales, ordenados por orden alfabético (Figura 2).
- `void mostrar_Datos_Usuario (const tPuntuaciones& marcador)`: Muestra todos los datos de todos los usuarios (Figura 3). Los usuarios están ordenados alfabéticamente y los identificadores de las minas, al ser numéricos, están ordenados crecientemente.

Y para el tratamiento del vector dinámico

- `void inicializar_Marcador(tPuntuaciones& marcador)`: Inicializa el array dinámico.
- `void aumentar_Capacidad(tPuntuaciones& marcador)`: duplica el tamaño del array.
- `void destruir(tPuntuaciones& marcador)`: libera la memoria dinámica.
- `bool buscar(const string& nombre, const tPuntuaciones& marcador, int& pos)`: Busca un nombre en el array dinámico y devuelve si se encuentra o no. Si el nombre está en el array entonces se devuelve la posición en la cual se encuentra y si no está en el array entonces se devuelve la posición dónde debería estar. Los datos del array dinámico se encuentran ordenados por orden alfabético. Debe realizarse una **búsqueda binaria iterativa** de los datos.
- `void insertar(tPuntuaciones& marcador, string const& nombre, int pos)`: Inserta **ordenadamente** en la posición pos un nuevo jugador (nombre). Al realizar la inserción de un nuevo jugador, si no hay suficiente espacio en el array, se amplia.

1. Entrega de la práctica

La práctica se entregará a través del Campus Virtual. Se habilitará una nueva tarea **Entrega de la Práctica** que permitirá subir un archivo comprimido con todos los archivos de código fuente .h y .cpp.