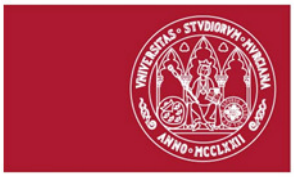

Estructura y Tecnología de Computadores

Examen final de prácticas (3 de febrero de 2021)

Se incluyen dos versiones incompletas de un programa «progexamen»: una en C y otra en ensamblador MIPS. El programa, una vez completo, mostrará un menú donde se nos ofrecerán varias opciones para probar los ejercicios del examen que se describen a continuación.

Los códigos C y ensamblador que se proporcionan incluyen el menú y algún código auxiliar y de prueba, además de la cabecera de los procedimientos que se deberán implementar en ensamblador durante el examen. Sin embargo, no se incluye en la versión en C el código de algunos procedimientos que deberán ser diseñados e implementados durante el examen, por lo que no se puede compilar el programa dado. Por ello, también **se puede comprobar el funcionamiento que debe tener el programa una vez terminado en** `https://ditec.um.es/~rfernandez/etc-demo-examen/2021-02/`.

Teniendo en cuenta lo anterior, haga lo que piden los siguientes apartados, implementando los procedimientos que se piden en el fichero `progexamen.s`:

1. (2 puntos) **Traduzca** el procedimiento `contar_vecinos`, cuyo código se proporciona en el programa en C. Este procedimiento recibe como parámetro la dirección de memoria de una estructura de tipo `Tablero` y dos enteros especificado las coordenadas de una celda, y devuelve el número de celdas adyacentes a la indicada cuyo contenido es distinto del carácter de espacio en blanco. Este procedimiento se utiliza para simular el juego de la vida de Conway.

   ```
   int contar_vecinos(Tablero* tablero, int fila, int columna);
   ```

2. (2 puntos) **Implemente** el procedimiento `comparar_tableros_celda_a_celda` que recibe la dirección de una estructura de tipo `Tablero` (*`resultado`*) y la dirección de otras dos estructura de tipo `Tablero` (*`a`* y *`b`*) y devuelve un booleano. El procedimiento debe comprobar si los tableros *`a`* y *`b`* son del mismo tamaño. Si lo son, debe devolver `true`, inicializar el tablero *`resultado`* con el mismo tamaño y almacenar en cada celda de él el resultado de comparar las respectivas celdas de *`a`* y *`b`*, usando el carácter `'<'` para representar que el contenido de la celda de *`a`* es menor que de la de *`b`*, `'>'` si es menor o `'='` si son iguales. En caso de que los tableros no sean del mismo tamaño debe devolver `false`.

   ```
   bool comparar_tableros_celda_a_celda(Tablero* resultado, Tablero* a, Tablero* b);
   ```

   Nota: Puede usar las funciones `tablero_init`, `tablero_get_celda`, etc. para manipular las estructuras de datos.

3. (2 puntos) **Traduzca** el procedimiento `clasificar_numeros`, cuyo código se proporciona en el programa en C. El procedimiento recibe un puntero al comienzo de un array de enteros (*`entrada`*), un entero indicando el número de elementos en el array de entrada (*`n_elementos`*), y dos punteros al comienzo de dos arrays de enteros (*`pares`* e *`impares`*) previamente reservados de tamaño suficiente. El procedimiento copia los elementos pares del array de entrada en el en array *`pares`* y los elementos impares en el array (*`impares`*, y devuelve el número de elementos pares.

   ```
   int clasificar_numeros(int* entrada, int n_elementos, int* pares, int* impares);
   ```

4. (2 puntos) **Implemente** el procedimiento `eliminar_caracteres` que recibe la dirección de una estructura de tipo `Tablero` (*`tablero`*), la dirección de comienzo de un array de caracteres (*`eliminar`*) y un entero indicando el número de elementos en el array de caracteres (*`n_eliminar`*). El procedimiento debe eliminar del *`tablero`* todos los caracteres que aparezcan en el array *`eliminar`* sustituyéndolos (en el *`tablero`*) por un espacio en blanco.

   ```
   void eliminar_caracteres(Tablero* tablero, char* eliminar, int n_eliminar);
   ```

Nota: Puede usar las funciones `tablero_get_celda`, `tablero_set_celda`, etc. para manipular las estructuras de datos.

5. (2 puntos) **Corrija** los errores de la traducción del procedimiento `ordenar_items`, añadiendo un comentario al final del procedimiento explicando muy brevemente cada error corregido. Este procedimiento se llama desde la opción número 5 del menú y ordena un array de elementos del tipo estructura `Item` definido en el programa.

```
void ordenar_items(Item* array, int n_items);
```

Notas importantes a tener en cuenta para la realización de los ejercicios:

- Los procedimientos auxiliares y el menú están bien implementados y no es necesario ni recomendable gastar tiempo en entender cómo están implementados. Sólo hay que usarlos para comprobar el funcionamiento de los procedimientos implementados durante el examen.
- Para poder realizar los ejercicios, debe de **tener en cuenta las definiciones de los diferentes tipos de datos** utilizados que `se encuentran en la versión en C` del programa.
- En el caso de los ejercicios en los que se pide que se traduzca una función, la traducción debe de ser lo más literal posible y se debe evitar cualquier tipo de optimización (en particular, debe respetar fielmente la estructura de los bucles originales).
- En el caso de los ejercicios en los que se pide que se implemente alguna función, se pueden utilizar funciones auxiliares siempre que se considere oportuno.
- A la hora de evaluar, sólo se tendrá en cuenta el código ensamblador (se ignorará cualquier modificación de la versión en C). **Como respuesta al examen debe entregar solo el fichero progexamen.s modificado**.
- Puede comprobar si su solución se comporta como la solución correcta comparando la salida de su programa con la versión online, pero tenga en cuenta que el código de la solución puede ser incorrecto aún cuando se comporte correctamente (por ejemplo, si no se han seguido correctamente todos los convenios de programación o si se ha realizado una traducción incorrecta).