UNIVERSIDAD DE MURCIA

Estructura y Tecnología de Computadores

Examen final de prácticas (30 de junio de 2022)

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/2022-07/`.

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

1. (2 puntos) **Corrija** los errores de la traducción del procedimiento `buscar_montana_mas_alta`, añadiendo un comentario al final del procedimiento explicando muy brevemente cada error corregido. Este procedimiento se llama desde la opción número 1 del menú y devuelve un puntero a la montaña más alta del array global `montanas`. Tenga en cuenta que también puede haber errores en el segmento de datos, en concreto, en la traducción de las variables usadas por dicho procedimiento. Para que se considere correcto el ejercicio, el programa resultante debe respetar todos los convenios de programación vistos en clase y debe ejecutarse sin generar ningún aviso (*warning*) ni error.

   ```
   void buscar_montana_mas_alta(void);
   ```

2. (4 puntos) **Traduzca** los procedimientos `buscar_producto` y `calcular_precio_pedido`, cuyo código se proporciona en el programa en C. Una vez haya hecho la traducción, puede probar el funcionamiento de ambos procedimientos usando la opción 2 del menú (los dos procedimientos se prueban a la vez con dicha opción, no es posible probar cada procedimiento por separado).

   a) (2 puntos) El procedimiento `buscar_producto` recibe un puntero a una cadena y devuelve un puntero a un elemento `ProductoAlmacen` almacenado en el array `almacen` cuyo nombre coincida con el buscado. En caso de que no se encuentre un producto con ese nombre devuelve `NULL`.

      ```
      ProductoAlmacen* buscar_producto(char* nombre);
      ```

   b) (2 puntos) El procedimiento `calcular_precio_pedido` recibe un puntero a una estructura de tipo `Pedido` y devuelve el precio total del pedido. Para ello, consulta el precio de cada ítem en el array `almacen`, usando el procedimiento `buscar_producto`, y multiplica el precio por la cantidad del ítem en el pedido. En caso de que `buscar_producto` no encuentre un producto del pedido y devuelve `NULL`, mostrará un aviso e ignorará ese producto.

      ```
      bool calcular_precio_pedido(Pedido* pedido);
      ```

3. (4 puntos) **Implemente** el procedimiento `crear_pedido_llenar_almacen_y_calcular_coste` que recibe como parámetro un número que representa la cantidad mínima de cada producto que se desea que esté disponible en el almacén (`stock_minimo`, igual para todos los productos) y un puntero a un pedido vacío (`pedido`):

   ```
   int crear_pedido_llenar_almacen_y_calcular_coste(int stock_minimo, Pedido* pedido);
   ```

   El procedimiento calcula cuánto costaría comprar todos los productos necesarios para que, al menos, de cada producto haya una cantidad igual o superior al stock mínimo y rellena el pedido que habría que realizar para conseguirlo. Para ello, para cada producto del almacén, si la cantidad de producto en stock es menor que `stock_minimo`, calcula cuánto costaría adquirir la cantidad que falta del mismo teniendo en cuenta su precio y, mediante el procedimiento `anadir_producto_a_pedido`, añade la información al pedido `pedido`. Si de un

producto ya hay existencias suficientes, no lo añade al pedido. El procedimiento devuelve el coste total de todos los productos que hay que comprar (es decir, el coste del pedido que habría que hacer).

Para implementar este procedimiento se debe hacer uso del procedimiento `anadir_producto_a_pedido`, que recibe un puntero a un pedido (`pedido`), el nombre de un producto a añadir al mismo (`nombre_pedido`) y la cantidad de ese producto (`cantidad`):

```
void anadir_producto_a_pedido(Pedido* pedido, char* nombre_producto, int cantidad);
```

Una vez implementado, el procedimiento `crear_pedido_llenar_almacen_y_calcular_coste` se puede probar con la opción 3 del menú.

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. Solo hay que usarlos para comprobar el funcionamiento de los procedimientos implementados durante el examen.
- Para poder realizar los ejercicios, debe **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, solo 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 aun 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).