Facultad de Informática – Universidad Complutense de Madrid

Fundamentos de la programación I – Grupos D, G, I, DG-ADE, DG-Mat

Examen de la Convocatoria Ordinaria – 2021-2022

Tiempo disponible: 3 horas

Se quiere construir un programa en C++ que gestione el envío de los pedidos que los clientes realizan a través de la plataforma *JustLunch*, asignando cada envío a uno de los 8 *riders* que tienen contratados. La empresa puede gestionar un máximo de 50 pedidos (el número de riders y el máximo de pedidos pueden variar con el tiempo, pero no durante la ejecución del programa).

La información de los pedidos realizados está en el archivo `pedidos.txt`. Cada línea contiene los datos de un pedido, separados por un espacio: momento en el que se ha solicitado (horas y minutos), minutos que tarda un rider en gestionar el envío y si se trata de un pedido prioritario (`1`) o no (`0`). La última línea contiene -1 como horas (centinela).

`pedidos.txt`

```
13 0 17 1
13 0 14 0
13 0 16 0
13 0 22 1
13 3 19 0
13 5 17 1
13 7 18 0
...
13 59 10 1
-1
```

Además de los datos obtenidos del archivo (`solicitado`, `duración` y si `prioritario`), mantendremos para cada pedido también otros datos: el momento del `envío` del pedido (que es la hora a la que el rider lo va a llevar), el número de `rider` al que se asigna el envío (`0..NumRiders-1`) y si se ha `procesado` o no (es decir, si se le ha asignado un rider).

De cada rider se guarda la hora en la que está `disponible` para un nuevo pedido (todos empiezan a trabajar a las 13:00) y las `ganancias` acumuladas, teniendo en cuenta que se les paga, actualmente, 25 céntimos por cada minuto que dura un envío. A los riders se les identifica por su posición en la lista.

Detalles de implementación

En el programa se usará un tipo `tTiempo` que incluya las `horas` y los `minutos` de un instante de tiempo. **[1 punto]**

Debes sobrecargar los siguientes operadores para el tipo `tTiempo`:

- `bool operator<(tTiempo const& tiempoIzq, tTiempo const& tiempoDer)`: dados dos instantes de tiempo devuelve `true` si `tiempoIzq` es anterior a `tiempoDer` y `false` en caso contrario.

- `tTiempo operator+(tTiempo const& tiempo, int min)`: dado un instante de tiempo y un número de minutos, devuelve el instante temporal resultado de sumar al anterior esos minutos.
- `std::ostream& operator<< (std::ostream& out, tTiempo const&tiempo)`: dado un instante lo muestra por pantalla con el formato `HH:MM` (hora y minutos siempre con dos dígitos).

Crea los tipos `tRider`, con los datos que se quieren tener de cada rider (`ganancias` y `disponible`), y `tArrayRiders` para el array de `NumRiders` riders. **[0,5 punto]**

Declara también los tipos `tPedido`, con los datos de cada pedido (`solicitado`, `iniciado`, `duración`, `rider`, `prioritario` y `procesado`) y `tListaPedidos` para la lista de pedidos. **[1 punto]**

Implementa, al menos, los siguientes subprogramas:

- `carga()`: vuelca la información de `pedidos.txt` en la lista de pedidos. Devuelve `true` si se pudo abrir el archivo o `false` en otro caso. Si en el archivo hay más pedidos que el máximo permitido, se ignorarán los demás. **[1 punto]**
- `siguiente()`: dada la lista de pedidos devuelve la posición (índice), en la lista, del primer pedido prioritario no procesado. Si no quedan pedidos prioritarios sin procesar, devuelve el primero no procesado. **[1,5 puntos]**
- `eligeRider()`: dado el array de riders devuelve la posición (índice), en el array, del rider que antes esté disponible. **[1 punto]**
- `planifica()`: dada la lista de pedidos y el array de riders, procesa todos los pedidos, empezando por los prioritarios, asignando cada pedido al rider que antes esté disponible, rellenando los datos del pedido de hora de inicio y rider asignado, así como marcándolo como procesado. También actualizará las ganancias del rider asignado y su nueva disponibilidad. Además, para cada pedido mostrará la asignación, tal como se muestra en el ejemplo de ejecución. **[2,5 puntos]**
- `muestra()`: dada la lista de pedidos y el array de riders, muestra para cada rider los pedidos que gestiona y sus ganancias (*ver el ejemplo de ejecución*). Se utilizará una función auxiliar que muestre los pedidos de un sólo rider. No está permitido ordenar la lista de pedidos de ninguna forma. **[1 punto]**

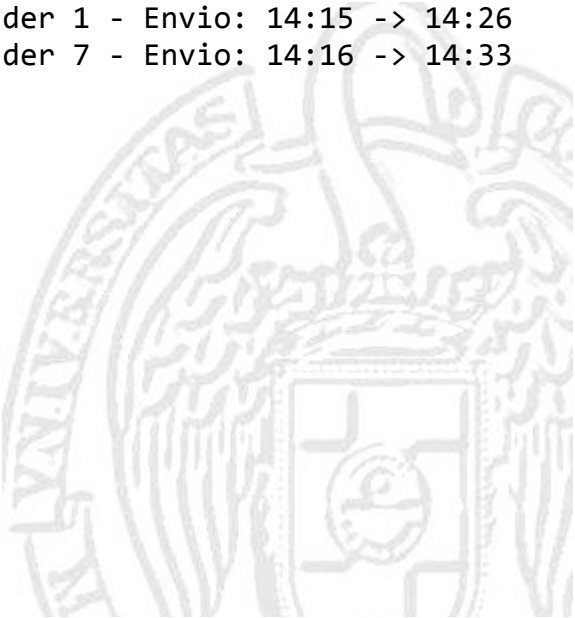
El programa principal comenzará llamando a `carga()` y si se han podido cargar los pedidos, llamará a `planifica()` para asignar un rider a cada pedido (procesando primero los pedidos prioritarios), para terminar llamando a `muestra()` para ver los pedidos asignados a cada rider y sus ganancias. **[0,5 puntos]**

Recuerda que en el código que desarrolles se valorarán la corrección del diseño, la eficiencia del código, la ausencia de código repetido, la correcta utilización de los esquemas vistos en clase, etc. Recuerda que no se permite el uso de variables globales ni de instrucciones de salto, salvo `return` al final de las funciones.

Ejemplo de ejecución

```
Se asignan los riders...
Pedido a las 13:00 - Prioridad: S - 17' - Rider 1 - Envio: 13:00 -> 13:17
Pedido a las 13:00 - Prioridad: S - 22' - Rider 2 - Envio: 13:00 -> 13:22
Pedido a las 13:05 - Prioridad: S - 17' - Rider 3 - Envio: 13:05 -> 13:22
Pedido a las 13:10 - Prioridad: S - 15' - Rider 4 - Envio: 13:10 -> 13:25
Pedido a las 13:12 - Prioridad: S - 16' - Rider 5 - Envio: 13:12 -> 13:28
Pedido a las 13:16 - Prioridad: S - 14' - Rider 6 - Envio: 13:16 -> 13:30
Pedido a las 13:20 - Prioridad: S - 19' - Rider 7 - Envio: 13:20 -> 13:39
Pedido a las 13:20 - Prioridad: S - 24' - Rider 8 - Envio: 13:20 -> 13:44
Pedido a las 13:30 - Prioridad: S - 14' - Rider 1 - Envio: 13:30 -> 13:44
Pedido a las 13:45 - Prioridad: S - 15' - Rider 2 - Envio: 13:45 -> 14:00
Pedido a las 13:56 - Prioridad: S - 14' - Rider 3 - Envio: 13:56 -> 14:10
Pedido a las 13:59 - Prioridad: S - 10' - Rider 4 - Envio: 13:59 -> 14:09
Pedido a las 13:00 - Prioridad: N - 14' - Rider 5 - Envio: 13:28 -> 13:42
Pedido a las 13:00 - Prioridad: N - 16' - Rider 6 - Envio: 13:30 -> 13:46
Pedido a las 13:03 - Prioridad: N - 19' - Rider 7 - Envio: 13:39 -> 13:58
Pedido a las 13:07 - Prioridad: N - 18' - Rider 5 - Envio: 13:42 -> 14:00
Pedido a las 13:10 - Prioridad: N - 16' - Rider 1 - Envio: 13:44 -> 14:00
Pedido a las 13:15 - Prioridad: N - 13' - Rider 8 - Envio: 13:44 -> 13:57
Pedido a las 13:15 - Prioridad: N - 17' - Rider 6 - Envio: 13:46 -> 14:03
Pedido a las 13:15 - Prioridad: N - 13' - Rider 8 - Envio: 13:57 -> 14:10
Pedido a las 13:18 - Prioridad: N - 18' - Rider 7 - Envio: 13:58 -> 14:16
Pedido a las 13:20 - Prioridad: N - 15' - Rider 1 - Envio: 14:00 -> 14:15
Pedido a las 13:22 - Prioridad: N - 12' - Rider 2 - Envio: 14:00 -> 14:12
Pedido a las 13:25 - Prioridad: N - 27' - Rider 5 - Envio: 14:00 -> 14:27
Pedido a las 13:30 - Prioridad: N - 11' - Rider 6 - Envio: 14:03 -> 14:14
Pedido a las 13:40 - Prioridad: N - 15' - Rider 4 - Envio: 14:09 -> 14:24
Pedido a las 13:45 - Prioridad: N - 13' - Rider 3 - Envio: 14:10 -> 14:23
Pedido a las 13:50 - Prioridad: N - 13' - Rider 8 - Envio: 14:10 -> 14:23
Pedido a las 13:50 - Prioridad: N -  8' - Rider 2 - Envio: 14:12 -> 14:20
Pedido a las 13:55 - Prioridad: N - 21' - Rider 6 - Envio: 14:14 -> 14:35
Pedido a las 13:57 - Prioridad: N - 11' - Rider 1 - Envio: 14:15 -> 14:26
Pedido a las 13:59 - Prioridad: N - 17' - Rider 7 - Envio: 14:16 -> 14:33

Rider  Solicitado    Iniciado   Terminado
  1       13:00       13:00       13:17
  1       13:10       13:44       14:00
  1       13:20       14:00       14:15
  1       13:30       13:30       13:44
  1       13:57       14:15       14:26
  Ganancias: $15.33
  2       13:00       13:00       13:22
  2       13:22       14:00       14:12
  2       13:45       13:45       14:00
  2       13:50       14:12       14:20
  Ganancias: $11.97
  3       13:05       13:05       13:22
```



```
3       13:45       14:10       14:23
3       13:56       13:56       14:10
Ganancias: $9.24
4       13:10       13:10       13:25
4       13:40       14:09       14:24
4       13:59       13:59       14:09
Ganancias: $8.40
5       13:00       13:28       13:42
5       13:07       13:42       14:00
5       13:12       13:12       13:28
5       13:25       14:00       14:27
Ganancias: $15.75
6       13:00       13:30       13:46
6       13:15       13:46       14:03
6       13:16       13:16       13:30
6       13:30       14:03       14:14
6       13:55       14:14       14:35
Ganancias: $16.59
7       13:03       13:39       13:58
7       13:18       13:58       14:16
7       13:20       13:20       13:39
7       13:59       14:16       14:33
Ganancias: $15.33
8       13:15       13:44       13:57
8       13:15       13:57       14:10
8       13:20       13:20       13:44
8       13:50       14:10       14:23
Ganancias: $13.23
```