Examen Práctica Barcas — X43982_es

Este único problema es el examen completo sobre la práctica "Alquiler de Barcas en el Río".

Para hacer el examen es necesario **descargar** el fichero `.tar` que se encuentra en el icono del gatito, descomprimirlo y **trabajar sin usar el Jutge en absoluto**. Es decir, el envío al Jutge solo sirve como **entrega**, el Jutge no pasará ningún juego de pruebas y aceptará cualquier envío, sin ni siquiera compilarlo. La corrección será totalmente manual. Ignorad el ZIP con los cubos de colores, ya que todo lo necesario está contenido en el ZIP del gatito.

Para ayudar con el desarrollo, se proporcionan juegos de pruebas (`test-*`), y el flujo de trabajo consiste en modificar los ficheros de código y hacer `make test`, que compila el proyecto y también ejecuta unos juegos de prueba cortos (y que se pueden leer con relativa facilidad), y que permiten comprovar si el programa produce la salida esperada. Los ficheros `test-*.inp` son las entradas y los ficheros `test-*.cor` son las salidas correctas.

Cada juego de prueba testea un aspecto del proyecto, y los primeros 3 ya dan la salida correcta con el código entregado (`test-1-*` a `test-3-*`). El resto de juegos de prueba, a partir de `test-a-*`, se corresponden a los apartados del examen. Al completar un apartado, y hacer `make test`, si éste es correcto, se producirá un mensaje que lo confirma (`OK` o `WA`).

Ejercicios

a) [*1 punto*] Añade una opción al menú principal, llamada "`escribir_estructura`", que muestre la estructura del río, usando el método correspondiente de la clase `Rio`. Respeta al máximo la organización del código dado, en estilo y al elegir dónde situar el código nuevo.

b.1) [*1 punto*] Añade un campo para el día actual al `Rio` (un `int` solamente), e inicialízalo al valor 1. Crea una opción en el menú principal llamada "`dia_siguiente`" que pase al dia siguiente (incremente el día en una unidad). Haz los métodos y funciones necesarios.
b.2) [*2 puntos*] Modifica `Viaje` para que cada viaje tenga un campo `dia` para guardar en qué día se hizo el viaje. Cambia `Barca::viaje_anyade` y `CjtBarcas::barca_mueve` para pasar el día desde el `Rio`. Finalmente cambia también `Barca::viajes_lista` para que se muestre el día en los viajes de una barca, después del destino, tal como se ve en `test-b-viajes-barca.inp`.

c) [*2 puntos*] Añade la opción "`modificar_capacidad`" que permite cambiar la capacidad de una estación. Esta operación lee el ID de una estación, la nueva capacidad, y muestra una línea del tipo "`#mc <ID> <capacidad>`" con los datos entrados (como el resto de operaciones), comprueba que la estación con ese ID exista con el método `Rio::estacion_existe` (de lo contrario muestra "error: la estacion no existe"), y finalmente cambia la capacidad, haciendo un nuevo método en la clase `Rio`, que use los métodos apropiados de `Estacion`.

d) [*2 puntos*] La operación que asigna una barca a una estación recibe el ID de una barca y busca la mejor estación (usando el coeficiente de ocupación). La opción de menú "`asignar_estacion`" aún no está hecha en el programa principal, pero ya disponemos del método recursivo `Rio::barca_asigna`
Terminal la implementación esta opción añadiendo una función para el programa principal que haga las comprobaciones necesarias y llame al método `Rio::barca_asigna_estacion`, completando también ese método.

e) [*2 puntos*] A veces los clientes olvidan objetos personales en las barcas, y se necesita una forma de saber el ID de la barca en la que habría que comprobar si el objeto aún está. Haz

una operación nueva "`buscar_barca_por_viaje`" que haga una lista de los IDs de las barcas que hicieron un viaje desde cierta estacion de origen, hasta cierto destino, y en cierto día. La operación leerá los IDs de la estación de origen y destino, el día y mostrará a la salida los IDs de las barcas encontradas (si hay), ordenadas por el ID mismo. Consulta los juegos de pruebas `test-e-*` para ver entrada y salida en más detalle. Introduce lo que consideres necesario (funciones, métodos, etc.) para implementar la nueva operación.

Antes de hacer la entrega final, repasa en detalle el código que has hecho para que sea fácil de leer y entender por otra persona.

IMPORTANTE: **Para hacer la corrección se usará la última entrega**.

Observación

Los ficheros públicos (icono del gatito) contienen:

`Makefile`	para compilar con `make`
`*.cc`	implementación de las clases del proyecto
`*.hh`	ficheros de cabecera de las clases del proyecto
`.clang-format`	para el formateo automático
`program.cc`	programa principal
`test-*.inp`	juegos de prueba (entrada)
`test-*.cor`	juegos de prueba (salida correcta)

Información del problema

Autoría: PRO2

Generación: 2026-01-25T21:12:25.751Z