Uso de pilas y colas

- Este documento contiene ejercicios que hay que resolver en el Jutge (en la lista correspondiente del curso actual) y que aquí están señalados con la palabra *Jutge*.
- Recomendamos resolver los ejercicios en el orden en el que aparecen en este documento. No se supervisarán los problemas del Jutge si antes no se han resuelto los ejercicios previos.

En esta sesión trabajaremos con pilas (`stack`) y colas (`queue`) de la la Standard Template Library de `C++`. Podemos consultar como usarlas en los respectivos ficheros de documentación `.pdf`. Como se trata de clases estándar, no hará falta mencionar sus correspondientes ficheros `.o` al enlazar nuestros programas. Eso sí, recordad que en estas clases no se controla el cumplimiento de las precondiciones de las operaciones, por lo que interesará usar la opcion de compilación `-D_GLIBCXX_DEBUG` para obtener información adicional en caso de `segmentation fault`.

En la carpeta correspondiente se muestran ejemplos resueltos con pilas y colas de enteros. Además se incluyen ficheros con operaciones de lectura y escritura de pilas de enteros: `stackIOint.hh` y `stackIOint.cc` y para colas de enteros: `queueIOint.hh` y `queueIOint.cc`. Notad que dichas operaciones no pueden ser genéricas, pues dependen directamente del tipo del contenido de las estructuras.

1. Uso de la clase `stack`

En los apuntes de teoría tenéis una serie de ejemplos sobre pilas de enteros. Queremos obtener programas similares sobre pilas cuyos elementos sean objetos de clases com `Estudiant` u otras. Cada ejercicio requiere programar una o más operaciones de pilas y un método `main` para probarlas. Tened en cuenta el contenido del fichero `stack.pdf`. Al final incluimos ejercicios que requieren más específicamente el uso de la clase.

1.1. Ejercicio: Búsqueda en una pilas de ParInt (X18958) de la Lista *Sessió 5 (Jutge)*

1.2. Ejercicio: búsqueda en una pila de estudiantes

Dado un número y una pila de estudiantes, programad una operación que busque si hay algun estudiante en la pila que lo tenga como DNI. En caso de éxito, debe recuperar información sobre la nota del estudiante. Si existe más de un par en esas condiciones, se ha de obtener el más cercano a la cima de la pila.

El resultado del programa principal puede ser uno de éstos

```
El estudiante no está en la pila

El estudiante está en la pila, pero no tiene nota

El estudiante está en la pila y su nota es <la que sea>
```

Necesitaréis implementar los ficheros con las operaciones de lectura y escritura de pilas de Estudiant: PilaIOEstudiant.hh y PilaIOEstudiant.cc.

1.3. Ejercicio: modificar los elementos de una pila

Para los siguientes ejercicios, producid dos soluciones, una en la que se obtenga una pila nueva a partir de la original y otra en la que las modificaciones se apliquen directamente sobre la original.

1. Con la misma estructura del ejercicio anterior, escribid una operación que dada una pila de ParInt le sume k al segundo elemento de cada par. Podéis encontrar la clase ParInt ya implementada en los ficheros públicos del problema del Jutge X18958.

2. Aplicad las ideas del problema anterior para redondear las notas de una pila de estudiantes.

1.4. Ejercicio: Parentización de expresiones (X36902) de la Lista *Sessió 5 (Jutge)*

Dada una secuencia de caracteres marcada por un punto y formada por paréntesis abiertos y cerrados de dos tipos, comprobar si es sintácticamente correcta, es decir, si todo paréntesis abierto se cierra más adelante y no aparece un paréntesis cerrado antes de lo que le toca. Los caracteres han de tratarse según se leen, de forma que si un carácter determinado da lugar a un error de parentización ya no se traten los siguientes.

En la versión del jutge se pide que simplemente se diga si la expresión es correcta o no. Realizad otra versión que además diga cuál es el error y la posición donde se produce. Excepcionalmente, este ejercicio se puede resolver solo con el método main, leyendo y tratado cada caracter al vuelo.

Ejemplos:

`[(()[]())()]. correcto`

`([)(]). incorrecto (lo sabemos al tratar el tercer carácter)`

`()()[](). incorrecto (lo sabemos al final)`

1.5. Ejercicio: mostrador de devoluciones de una biblioteca

Programad una miniaplicación que simule los movimientos del mostrador de devoluciones de una biblioteca. No es necesario que el programa sea completamente modular, pero es obligatorio usar la clase `stack`.

Los libros y otros materiales disponibles en la biblioteca están clasificados en N categorías y cada una de ellas tiene asociada una pila en el mostrador. N es un entero mayor que cero y será un dato del programa. Un libro tendrá dos atributos: su título, que será una `string`, y su categoría, que será un entero entre 1 y N.

Se contemplan dos tipos de movimientos:

- un usuario devuelve un libro y lo coloca en la pila de devoluciones correspondiente a su categoría (opción -1)
- el personal de la biblioteca retira un cierto número de libros correlativamente a partir de la cima de una de las pilas del mostrador; dicho número no puede ser mayor que la altura de la pila (opción -2)

Además, incluiremos una opción -3, consistente en escribir el contenido de una de las pilas del mostrador.

Organizad el programa principal como un proceso iterativo que, tras inicializar todos sus componentes, lea un valor entre -1 y -4 indicativo de la opción que queremos aplicar (la opción -4 será la de terminar el programa), después lea los datos necesarios para ella y proceda a aplicarla. Los demás procesos iterativos (escribir una pila y quitar elementos de una pila) han de programarse en operaciones aparte.

En el fichero `bibli.dat` tenéis un ejemplo de entrada de datos y en `bibli.sal`, la salida esperada. Probad vuestro programa en otras situaciones.

2. Uso de la clase `queue`

Para probar la clase `queue` se pueden utilizar ejercicios análogos a los que vimos para la clase `stack`: búsqueda en una cola `ParInt` o `Estudiant` y modificar los elementos de una cola de `ParInt` o `Estudiant`. Nuevamente, deberéis producir los ficheros `queueIOParInt.hh` y `queueIOParInt.hh` o `queueIOEstudiant.hh` y `queueIOEstudiant.cc` cuando corresponda. También incluimos ejercicios que requieren más específicamente el uso de la clase. Cada ejercicio requiere programar una o más operaciones de colas y un método `main` para probarlas. Tened en cuenta el contenido del fichero `queue.pdf`.

2.1. Ejercicio: búsqueda en una cola de pares de enteros

Dado un número y una cola de objetos `ParInt` que representen pares de números enteros, programad una operación que compruebe si dicho número aparece como primer elemento de algún par de la cola y, en caso de éxito, escribid el par completo. Si existe más de un par en esas condiciones, se ha de obtener el más cercano al principio de la cola.

Por ejemplo, si la cola contiene el par (3, 6) y buscamos el elemento 3, el programa ha de retornar algo como

```
El compañero del 3 en la cola es el 6.
```

Podéis encontrar la clase `ParInt` ya implementada en los ficheros públicos del problema del Jutge X18958.

2.2. Ejercicio: Búsqueda en una cola de estudiantes (X21591) de la Lista *Sessió 5 (Jutge)*

2.3. Ejercicio: modificar los elementos de una cola

Como se muestra en los apuntes de teoría, este problema varía significativamente respecto al de pilas, ya que no es tan sencillo colocar un elemento en la primera posición de una cola ya creada.

Para los siguientes ejercicios, producid dos soluciones, una en la que se obtenga una cola nueva a partir de la original y otra en la que las modificaciones se apliquen directamente sobre la original.

1. Con la misma estructura de los ejercicios anteriores, escribid una operación que dada una cola de objetos `ParInt` le sume un valor k al segundo elemento de cada par.

2. Aplicad las ideas del problema anterior para redondear las notas de una cola de estudiantes.

2.4. Ejercicio: convertir una cola en una pila

Escribid una operación que convierta una cola en una pila con los mismos elementos y en el mismo orden. El primer elemento de la cola ha de convertirse en la cima de la pila y así sucesivamente.

2.5. Ejercicio: Distribución justa de colas (X13425) de la Lista *Sessió 5 (Jutge)*

Consideremos una cola de pares de naturales, donde cada par contiene un identificador y un valor. La cola representa personas (*identificadores*) que esperan delante de una ventanilla para realizar un trámite burocrático que tiene un coste en tiempo (*valores*). En un momento determinado, se cierra la ventanilla y se abren dos ventanillas nuevas, de forma que las personas de la cola original

(que quedará vacía) se han de distribuir de forma justa en dos nuevas colas teniendo en cuenta el coste en tiempo asociado a los trámites.

Una distribución (`c1, c2`) es justa respecto de una cola original `c` si ninguna persona ha de esperar más que otra que tenía detrás en `c` y todas esperan el mínimo tiempo posible. Para que la solución de este ejercicio sea única para cada cola original `c`, diremos que si una persona tiene la opción de ir tanto a `c1` como a `c2`, irá siempre a `c1`.

La distribución de una cola ha de programarse en una operación aparte. Organizad el programa de forma que el método `main` solamente lea los datos, realice la llamada a la mencionada operación y escriba los resultados.

Reformulad el ejercicio suponiendo que la ventanilla original no se cierra y que solo se abre una ventanilla nueva, con su correspondiente cola vacía. De esta forma, la distribución justa solo requerirá que se cambien de cola las personas necesarias para que se cumplan los requisitos anteriormente expuestos. Intentad que la solución no use estructuras auxiliares, ya sean colas o de otro tipo.

En el fichero `distrib_justa.dat` se incluye un ejemplo de entrada para este ejercicio y en `distrib_justa.sal` se encuentra la salida correspondiente. Probad vuestro programa en otras situaciones.

2.6. Ejercicio: estadísticas de una secuencia de enteros con borrado

Consideremos una secuencia de números enteros comprendidos entre -1000 y 1000, con instrucciones de borrado intercaladas en cualquier momento. Cada vez que se lea o se borre un número, se ha de obtener el mínimo, el máximo y la media de los números que se hayan leído hasta el momento, excepto los que se hayan borrado.

Cualquier número mayor que 1000 o menor que -1001 marca el final de la secuencia. El número -1001 representa una instrucción de borrado, en concreto del elemento más antiguo de la secuencia.

Si tras un borrado la secuencia está vacía (tanto si es porque se ha borrado su único elemento, o porque ya lo estaba), solo se ha de escribir un cero.

Todo proceso iterativo auxiliar de las estadísticas ha de programarse en operación aparte. Organizad el programa de forma que el método `main` solamente lea los datos uno por uno, aplique el tratamiento oportuno tras cada lectura o borrado y escriba el resultado correspondiente. En cada tratamiento sólo se puede recorrer la secuencia una vez como máximo y sólo si es estrictamente necesario.

En el fichero `estadisticas_cola.dat` se incluye un ejemplo de entrada para este ejercicio y en `estadisticas_cola.sal` se encuentra la salida correspondiente. Probad vuestro programa en otras situaciones.