Estructura de datos y de la información
Boletín de problemas - Tema 5

1. Uno de los problemas con la implementación estática de la pila mediante vectores es que esta se puede llenar. Con el fin de detectar esta posibilidad vamos a implementar una función *full* que nos devolverá *true* si la pila está llena y *false* en caso contrario. Dado que el usuario final del tipo pila no tiene porque poder acceder a esta nueva función, vamos a añadirla en la parte privada de la clase en C++, y por tanto tan sólo podrá ser usada por las funciones miembros de la clase. Implementar esta nueva función y modificar las operaciones de la clase que la puedan utilizar.

2. En la implementación de la pila presentada en el Tema 5, los errores producidos en las distintas operaciones se han tratado mediante la escritura de un mensaje por pantalla.

   a) Modificar las operaciones de forma adecuada para que no escriban ningún mensaje de error por pantalla, sino que devuelvan un valor booleano indicando si se ha producido o no.

   b) Será entonces el usuario final de la pila el que se encargará de detectar este tipo de errores si lo desea. Implementar un pequeño programa que utilice la pila, que provoque errores debidos a "pila llena" y "pila vacía", los detecte y escriba por pantalla los mensajes adecuados.

3. Vamos a implementar una operación *bottom* que nos devuelva el elemento situado en el fondo de la pila sin modificarla.

   a) Implementar la operación anterior como una función miembro de la clase. Esto es, la función se incluirá en la parte pública de la clase y podrá acceder a las operaciones y datos de la parte privada de la misma.

   b) Implementar la operación anterior como una función externa a la clase. Por lo tanto, la función no podrá acceder a la parte privada de la clase, sino que tendrá que manejar la pila utilizando la operaciones miembro.

4. Se desea implementar dos pilas, y se dispone de un solo vector de N componentes. Implementar ambas pilas de manera que se pueda aprovechar al máximo el vector. Las operaciones de pila tendrán que llevar un parámetro adicional que indique sobre qué pila se quiere realizar la operación (1 ó 2). (Ayuda: Las dos pilas crecen la una hacia la otra.)

5. Utilizando sólo operaciones de pila, realizar un algoritmo que:

   a) Invierta una pila

   b) Dada una pila y un valor umbral, devuelva otras dos pilas, en una de las cuales se han introducido los valores menores que el umbral, y en la otra los valores mayores o iguales que el umbral. (La pila inicial desaparece).

   c) Cambie todos los elementos de la pila iguales a X por Y, sin modificar el resto de la pila.

6. Implementar una pila de enteros en un vector de 0 a 100 elementos, donde el elemento 0 se emplea como tope de la pila, y el resto de elementos se emplean para albergar los elementos de la pila.

7. Un estacionamiento de coches contiene una sola línea, la cual puede guardar hasta 10 coches. Existe solamente una entrada/salida al estacionamiento en uno de los extremos de la línea. Si un usuario llega a retirar su coche que no está cerca de la salida, todos los coches que están bloqueando su salida son retirados, el coche del usuario sale, y los otros coches vuelven al sitio en que estaban originalmente, dejando el hueco en la parte más cercana a la salida.

   a) Define los tipos de datos necesarios para resolver el problema anterior.

   b) Escribe un algoritmo que procese un grupo de líneas de entrada. Cada línea de entrada contiene una 'e' para llegada o 's' para salida, y el número de matrícula. Se asume que los coches llegan y salen en el orden especificado en la entrada. El programa debe imprimir un mensaje cada vez que llega o sale un coche. Cuando un coche llega, el mensaje debe especificar si hay o no espacio para el coche en el estacionamiento. Si no hay espacio, el coche no podrá entrar al estacionamiento. Cuando un coche ha estado dentro del estacionamiento y sale, el mensaje debe incluir el número de veces que fue movido el coche para permitir que otros coches salieran.

8. Observando el siguiente algoritmo, se pide contestar a las preguntas que siguen:

```
void VaciarPila (Stack & p) {
  if ( ! p.empty() ) {
    p.pop();
    VaciarPila (p);
  }
}
```

a) ¿Finaliza la ejecución de este algoritmo?

b) ¿Cuál es el resultado de su ejecución? (esto es, ¿qué hace?)

9. Observando el siguiente algoritmo, se pide contestar a las preguntas que siguen:

```
int CuentaElemsPila (Stack & p) {
  Tbase x; int n;
  if (! p.empty()) {
    x = p.top ();
    p.pop ();
    n = CuentaElemsPila (p);
    p.push (x);
    n++;
  }
  return n;
}
```

a) ¿Cuál es el error del algoritmo?

b) ¿Cómo lo solucionarías?

c) Una vez solucionado el error, y sabiendo que cuando termina el algoritmo los elementos de la pila no se han modificado, ¿por qué se pasa la pila p por referencia?

d) ¿Se podría resolver de manera iterativa? ¿Cómo?

e) Escribe el algoritmo que resuelva el problema de manera iterativa.

10. Se utiliza, para implementar la estructura de datos Pila, vectores, que a lo sumo pueden contener MAXIMO elementos. (Siendo MAXIMO una constante que se supone definida).

a) Definir la estructura de datos necesaria para poder utilizar pilas con una capacidad máxima de 2*MAXIMO elementos.

b) Desarrollar los algoritmos que implementan las operaciones básicas sobre datos del tipo pila, según la estructura elegida en el apartado a).

11. Un funcionario de Juzgados tramita expedientes, de forma que siempre va cogiendo de la bandeja de expedientes a tramitar el que se encuentra más arriba. Cada expediente se puede caracterizar por un código, el asunto del que trata, la fecha de expedición y la fecha de tramitación. Un ordenanza es el encargado de traerle nuevos expedientes, que va dejando encima de los que ya había en la bandeja.

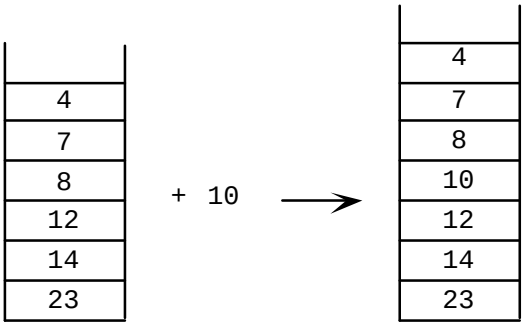
a) Definir la estructura de datos más adecuada para representar la bandeja de expedientes y la más adecuada para representar el cargamento de expedientes nuevos que trae el ordenanza.

b) Definir la operación *ApilarExp*, cuyo resultado es añadir a la estructura bandeja de expedientes los datos almacenados en la estructura ordenanza.

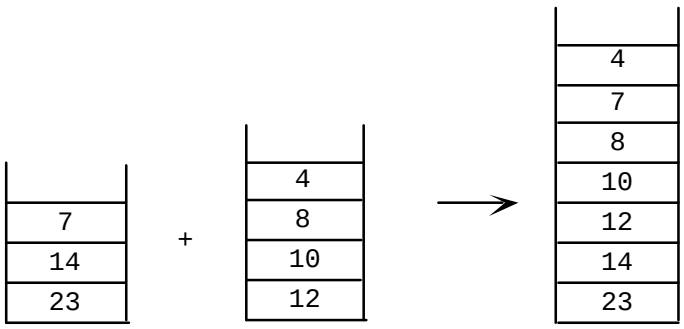
c) Como con este sistema es fácil que algún expediente se retrase mucho, de vez en cuando un juez pide que se tramite un determinado expediente urgentemente. Para ello indica el código de dicho expediente. Definir la operación *Urgencia*, que permitirá buscar un determinado expediente en la bandeja y eliminarlo de ella para tramitarlo.

12. Utilizando únicamente operaciones del TAD Pila para manejar la estructura:

a) Implementar una nueva operación *ApilarOrdenado*, tal que, suponiendo que disponemos de una pila cuyos elementos están ordenados de mayor a menor (el mayor en el fondo, el menor en el tope), permita apilar un nuevo elemento en la misma de forma que se conserve el orden. Por ejemplo:



b) Escribir un algoritmo mezclar que dadas dos pilas ordenadas como en el apartado anterior, construya una nueva pila ordenada del mismo modo en la que se mezclen los elementos de las dos originales. Por ejemplo:



13. Un profesor guarda las fichas de los alumnos por orden alfabético en un archivador de anillas. Por cada alumno guarda su nombre, dirección de correo electrónico y la nota de las 5 prácticas de la asignatura. Al abrir el archivador, todas las fichas quedan en un montón a la derecha y para acceder a una de ellas debemos ir pasando las que se encuentran por encima a la parte izquierda en orden contrario. En un momento dado el archivador puede estar abierto (con fichas a ambos lados) o cerrado (con fichas sólo en el lado derecho).

    a) Definir las estructuras de datos más adecuadas para guardar la información. Apoyarse en la estructura de datos más adecuada para implementar el archivador.

    b) Implementar la operación `BuscaFicha` que dado un archivador cerrado y el nombre de un alumno, busque su ficha y la devuelva como resultado. Si no la encuentra deberá devolver un error. Si se ha encontrado la ficha, el archivador debe quedar abierto con la misma encima del montón derecho. Si no se ha encontrado, el archivador debe cerrarse. El perfil del procedimiento será el siguiente:

    ```
    TFicha & BuscaFicha( Tarchivador & a, string nom, bool & error);
    ```

    c) Implementar un procedimiento `BorraFicha` que dado un nombre de alumno y un archivador cerrado, la extraiga del archivador y cierre este. Si no se encuentra la ficha, el procedimiento debe devolver un error. Utilizar el procedimiento del apartado b. El perfil del procedimiento será el siguiente:

    ```
    bool BorraFicha( Tarchivador & a, string nom);
    ```

14. Un trazador dibuja sobre un papel segmentos de un tamaño determinado según 4 posible movimientos: arriba ($\uparrow$), abajo($\downarrow$), izquierda ($\leftarrow$) y derecha ($\rightarrow$) Para realizar un dibujo, hay que darle órdenes de movimiento existiendo la posibilidad de deshacer el movimiento realizado en el orden contrario al que se llevó a cabo. Para ello, se necesita un histórico de movimientos realizados.

    a) Definir la estructura de datos más adecuada para implementar el histórico de movimientos.

    b) Realiza un procedimiento que imprima los movimientos realizados hasta el momento y el número de ellos. Los movimientos deben imprimirse en el mismo orden en que se realizaron y sin destruir el histórico.

c) Realiza un procedimiento que trasforme el histórico de un dibujo en el histórico del dibujo simétrico. Esto es cambiar cada movimiento por su simétrico.

d) Realiza un procedimiento que dado un histórico de un dibujo, devuelva el histórico del dibujo que se obtiene a partir del inicial pero sin movimientos repetidos consecutivos.