Estructura de datos y de la información
Boletín de problemas - Tema 6

1. Disponemos de un fichero binario que contiene en primer lugar un número entero que indica el tamaño de una matriz cuadrada de números reales (*double*), a continuación el fichero contiene los elementos de la matriz almacenados por filas y, finalmente, contiene los elementos de un vector del mismo tamaño. Implementar una función que lea del fichero el tamaño de la matriz, reserve espacio para la matriz y el vector, y los rellene con los elementos leídos.

2. Queremos almacenar los elementos de una matriz triangular inferior de números reales (*double*). Sin embargo, no queremos almacenar los elementos de la parte triangular superior, dado que todos valen 0. Para poder almacenar sólo los elementos de la parte triangular inferior, definiremos la matriz como un vector de punteros a *double*.

   a) Escribir una función que dado el vector anterior y un tamaño de matriz, reserve espacio para almacenar los elementos de cada una de las filas de la matriz. El primer puntero del vector pasado como parámetro apuntará a la primera fila, el segundo puntero a la segunda fila y así sucesivamente.

   b) Supongamos ahora que ni siquiera tenemos reservado el espacio para almacenar el vector de punteros. Implementar una función a la que se le pase el tamaño de la matriz y reserve espacio para el vector de punteros y para cada uno de los vectores fila de la matriz triangular. La función devolverá el vector de punteros apuntando a cada una de las filas.

   c) Supongamos que tenemos un fichero binario que contiene en primer lugar un número entero con el tamaño de una matriz, a continuación contiene los valores reales de cada uno de los elementos de una matriz triangular inferior de dicho tamaño (ordenados por filas) y, finalmente, contiene los valores de un vector de reales del mismo tamaño. Implementar una función que leyendo la información del fichero, reserve espacio para el vector de punteros, para la matriz y el vector de reales y los rellene con los elementos leídos. Para acabar, el programa deberá liberar todo el espacio reservado en el orden adecuado.

3. Supongamos que tenemos el siguiente código en C++ en el que se utiliza la implementación enlazada de la clase pila incluida en las transparencias:

```
stack p, q;
for (i=1; i<11; i++)
  p.push(i);
q = p;
```

   ¿Cuál sería el contenido de la pila `q` después de ejecutarse el código anterior?

4. Añadir un operador de asignación a la pila enlazada de las transparencias. Probar el resultado de ejecutar el código del ejercicio anterior. Para ello puede

ser conveniente añadir a la clase una operación que muestre por pantalla el contenido de la pila sin destruirla.

5. Suponer una implementación enlazada de la pila como la de las transparencias, a la que le hemos añadido un operador de asignación. Suponer que realizamos la llamada `resu=Invertir(pila)` al siguiente algoritmo:

```
stack Invertir(stack p) {
  stack paux;
  int i;
  while (!p.empty()) {
    i=p.top();
    paux.push(i);
    p.pop();
  }
  return aux;
}
```

   ¿Qué resultado obtendríamos en `resu`? ¿Qué tipo de paso de parámetros estamos utilizando? ¿Cómo quedaría `pila` después de la llamada?

6. Añadir un constructor de copia a la clase pila enlazada de las transparencias. Probar el resultado de ejecutar el código del ejercicio anterior.

7. ¿Cómo se modificaría la definición de la clase pila enlazada para almacenar el número de elementos de la pila? ¿Cómo se verían afectadas las operaciones miembro de la clase?

8. Supongamos la siguiente definición de nodo:

```
struct nodo {
  tipobase info;
  nodo *sig;
 }
```

   ¿Cómo se modificaría la implementación de la pila enlazada si utilizásemos la definición anterior en lugar de la clase nodo?

9. Supongamos que queremos implementar la pila enlazada utilizando dos clases separadas: una clase nodo y una clase pila. ¿Cómo se modificaría la implementación de la pila enlazada de las transparencias?