Estructura de datos y de la información
Boletín de problemas - Tema 2

1. Definir un tipo de datos *diassemana* capaz de contener cualquier día de la semana y las operaciones *diasiguiente*, *diaanterior* y *escribedia*.

2. Implementar un algoritmo usando el tipo de datos *diassemana* y sus operaciones. Este algoritmo debe hacer lo siguiente: Dado un día de la semana y un número de días, n, si $n \geq 0$ debe devolver qué día de la semana será n días después del día actual, y si $n < 0$ debe decir qué día de la semana fue n días antes.

3. Deseamos representar polinomios de grado menor o igual a N de manera que podamos acceder fácilmente a los coeficientes y también al grado del polinomio. Indica la estructura de datos estática más adecuada para representar un polinomio.

   a) Realiza un algoritmo que imprima por pantalla un polinomio dado usando el formato del siguiente ejemplo:

   $$3{,}0x^4 - 7{,}4x^1 + 2{,}8x^0$$

   b) Realiza un procedimiento en C++ que dados dos polinomios devuelva el polinomio resultante de la suma. Utiliza el siguiente perfil para el procedimiento:

   ```
   void sumapoli(const Tpoli & p, const Tpoli & q, Tpoli & r);
   ```

   c) Realiza un procedimiento que multiplique un monomio (polinomio de un solo término) por un polinomio, y devuelva el polinomio resultante del producto, o error en caso de que el polinomio resultante supere el grado máximo (N). Utiliza el siguiente perfil para el procedimiento:

   ```
   void multmono(const Tpoli & p, const Tpoli & q, Tpoli & r,
                 bool & error);
   ```

4. Una empresa de alimentación tiene n supermercados. Definir la estructura de datos más sencilla para almacenar por día, mes, y supermercado el total de caja realizado. Definir las operaciones que indiquen:

   a) Qué supermercado ha realizado el máximo de ventas totales anual,

   b) Qué supermercado ha realizado el máximo de ventas totales mensual, indicando el mes,

c) Qué día del año se ha obtenido la mayor venta, indicando el supermercado.

5. Queremos representar un fichero de libros de una librería en una estructura estática, que almacene la siguiente información de cada libro: título, autor, código, comentarios, stock, mínimo stock. Se pretende que los libros estén siempre ordenados por el código (pero los códigos no tienen porqué ser consecutivos). Se pide:

   a) Definir la estructura de datos más adecuada para representar el fichero de libros, sabiendo que el número máximo de libros distintos que puede albergar la librería es de 1000.

   b) Definir una operación de búsqueda binaria de un libro por su código, devolviendo si está, la posición del libro en el fichero, y si no está, la del código inmediatamente posterior a él.

   c) Definir una operación que inserte un nuevo libro en la estructura de datos.

   d) Definir una operación que borre un libro obsoleto de la estructura de datos, sabiendo su código.

6. Una agencia de viajes ofrece n destinos; para cada destino se puede optar por 5 clases de viaje: *super*, *luxe*, *normal*, *turista* y *estudiante*, y además se ofrecen tres tipos de alojamiento: AD (alojamiento y desayuno), MP (media pensión) y PC (pensión completa). Cada programa de viaje se caracteriza por la información (destino, clase, alojamiento), siendo la información significativa el precio del programa y el número de viajeros.
   Se desea informatizar la gestión de esta agencia. Entre otras cosas, es preciso:

   a) La definición de la estructura de datos que soporte la información descrita (es decir, la estructura que permita almacenar todos los programas de viaje de la agencia).

   b) Escribir un algoritmo que calcule la facturación total de la agencia y el número de viajeros para cada destino. Definir la estructura de datos más adecuada para contener la salida de datos, en el caso en que sea necesario.

   c) Escribir un algoritmo que, dada una clase de viaje, permita obtener el destino con mayor número de viajeros, indicando además dicho número.

7. El Instituto Nacional de Estadística quiere guardar los datos de las últimas elecciones *locales*, *autonómicas*, *europeas* y *generales* en cada una de las 50 provincias españolas. Por cada elección se guardará el número de votos a cada uno de los 5 partidos que se presentan.

   a) Definir las estructuras de datos más adecuadas para guardar la información anterior.

   b) Implementar un algoritmo que dado un tipo de elección y una provincia devuelva el partido ganador.

   c) Implementar un algoritmo que devuelva las provincias en las que el mismo partido ha ganado todas las elecciones. Utilizar el algoritmo del apartado b). Definir la estructura de datos más adecuada para devolver el resultado.

   d) Implementar un algoritmo que, dadas una elecciones, devuelva los votos acumulados por cada partido en todas las provincias. Utilizar la estructura de datos más adecuada para devolver el resultado.

8. Queremos guardar la información sobre los canales de televisión que se han visto cada día de la semana. Por cada uno de los 10 canales existentes queremos guardar el número de veces que se ha visto cada día y el tiempo de sintonización total por día medido en horas. Supondremos la estructura inicializada con los datos de un determinado usuario.

   a) Definir las estructuras de datos más adecuadas para guardar la información anterior.

   b) Suponiendo que un día d el usuario ha visto el canal c entre las *t1* y *t2* horas, desarrollar un algoritmo que actualice la estructura con esta nueva información.

   c) Desarrollar un algoritmo que devuelva el tiempo que el usuario ha visto cada canal durante toda la semana y el canal más visto. Utilizar la estructura de datos más adecuada para devolver los resultados.

   d) Desarrollar un algoritmo que devuelva el canal que más tiempo estuvo viendo un usuario y el día en que ocurrió.

9. La empresa de construcciones *Negocietes S.A.* promueve la construcción de el edificio de viviendas de lujo *Kese-kae*. El edificio está organizado en seis escaleras: *A*, *B*, *C*, *D*, *E* y *F*. Por cada escalera hay 8 plantas, y en cada planta hay 5 puertas. Por cada una de las viviendas la constructora guarda información del número de metros cuadrados, número de

habitaciones y el precio. Si además la vivienda está vendida, se guarda la información del nombre y NIF del propietario. Se pide:

a) Definir los tipos de datos capaces de describir toda esta información, sabiendo que las claves para acceder a la información relativa a una vivienda son escalera, planta y puerta.

b) Definir una operación que teniendo como dato de entrada el NIF de un comprador verifique si realmente es un comprador y cuántas viviendas son de su propiedad.

10. Queremos implementar una clase *mistring* que se comporte como la clase string en C++. Para ello utilizamos el siguiente esquema de declaración de la clase:

```
class mistring {
  public:
    // Constructor de una cadena vacia
    mistring(void);
    // Constructor de una cadena a partir de un vector de char
    mistring(char str[]);
    int length(void);
    int compare(char str[]);
    mistring & substr(int pos, int n);
    mistring & append(char str[]);
    mistring & assign(char str[]);
    mistring & erase(int pos, int n);
    mistring & replace(int pos, int n, char str[]);
  private:
    static const MAXCAR = 256;
    char elementos[MAXCAR];
};
```

Podemos ver que el nuevo tipo se ha implementado mediante un vector de tamaño constante. Para definir el final de la cadena añadiremos el carácter `'\0'` despues de su última componente. Implementar las operaciones especificadas para que se comporten como sus equivalentes en la clase string de C++ . Hacer uso de las operaciones incluidas en la librería de cadenas clásicas de C *cstring*. La especificación de todas estas operaciones puedes encontrarla en el apéndice D de [Nyhoff'99].

11. Escribe una función *replace_all()*, de modo que la siguiente llamada:

    ```
    nuevosubstring = replace_all(str, substring, nuevosubstring);
    ```

    devolverá una copia del string *str* con cada aparición de *substring* sustituida por *nuevosubstring*.

12. Escribe una función que cuente el número de veces que una cadena aparece dentro de otra.

13. Escribe una función que, dadas las tres componentes de un nombre (nombre, primer apellido y segundo apellido), devuelva una cadena conteniendo los apellidos, seguidos de una coma y del nombre. Por ejemplo, dadas las cadenas "Pablo", "Romero", "Cifuentes", la función devolverá la cadena "Romero Cifuentes, Pablo".

14. Escribe una función que acepte una cadena conteniendo el nombre y los dos apellidos y devuelva otra cadena conteniendo los apellidos, seguidos de una coma y de la inicial del nombre seguida de un punto. Por ejemplo, dada la cadena "Pablo Romero Cifuentes", la función devolverá la cadena "Romero Cifuentes, P."

15. Una cadena es un palindromo si no cambia al invertirse el orden de sus caracteres. Por ejemplo: "a", "aba", "abba", "abcba". Escribir una función que, dada una cadena, nos diga si es o no palindroma.

16. Una cadena es un anagrama de otra si contiene sus mismos caracteres en otro orden. Por ejemplo: "mero" y "remo". Escribir una función que, dadas dos cadenas, nos diga si una es un anagrama de la otra.