EXAMEN DE SISTEMAS INFORMÁTICOS INDUSTRIALES
(SOLUCIÓN) (TEORÍA)

DICIEMBRE 2014

1. Indicar si las siguientes afirmaciones son verdaderas o falsas:

- Sea el vector de enteros *int v[5]*. Para almacenar en la última posición del vector el valor 100, es correcto poner *v[5]=100;* **F**
- Para crear una cadena de caracteres con el texto *hola* es correcto poner *char cad[4]="hola";* **F**
- En el mecanismo de herencia de C++ se permite sobrepasar un método, es decir, en la clase derivada se puede volver a definir el método que ya estaba definido en la clase base. **V**
- El operador *&* se puede aplicar a un entero, por ejemplo *&i*, para obtener la dirección donde se almacena en memoria el entero. **V**
- Para reservar memoria dinámica en C++ de un vector unidimensional se utiliza el operador *new* en combinación con el operador *sizeof*. **F**
- La función *fgets* lee una cadena almacenada en un fichero, pero sólo hasta encontrar el primer espacio en blanco o el final de línea *\n*. **F**
- En C++ se puede definir el operador + como un método de un clase o como una función global. **V**
- La ventaja de utilizar el protocolo UDP es que es un protocolo orientado a la conexión y con control de errores en la transmisión. **F**

(2 puntos)

2. Escribir una función denominada **int ContarVuelos (char *nombreFich)** con las siguientes características:

- Recibe como parámetro una cadena de caracteres que contiene el nombre de un fichero. Se considera que dicho fichero ya existe. El fichero contiene en cada línea una cadena de caracteres que indica: el código de la compañía de avión (2 primeros caracteres) y el número de vuelo (4 caracteres como máximo). Los códigos válidos son: AA, BA, IB. Ejemplo de fichero ficticio a la derecha.

```
AA049
IB2345
AA23
BA2333
IB1456
IB5681
```

- La función debe mostrar en pantalla para cada compañía el número de vuelos efectuados. En el ejemplo anterior la salida por pantalla que daría el programa sería:

```
Vuelos AA: 2
Vuelos BA: 1
Vuelos IB: 3
```

- Si ha habido un error al abrir el fichero para lectura, la función devolverá -1, en otro caso devolverá 0.

(2 puntos)

Solución:

```
#include <stdio.h>
#include <stdlib.h>

int ContarVuelos(char *nombreFich)
{
      FILE *f;
      char cadena[7];
      int aa=0,ib=0,ba=0;

      f=fopen(nombreFich,"r");
      if(f==NULL){
            printf("Error: No existe el fichero\n");
            return -1;
      }
      // Se leen los datos del fichero
      while(!feof(f)){
            fscanf(f,"%s",cadena);
            if(cadena[0]=='A'&& cadena[1]=='A') aa++;
            else if(cadena[0]=='B'&& cadena[1]=='A') ba++;
            else if(cadena[0]=='I'&& cadena[1]=='B') ib++;
      }

      printf("Vuelos AA: %d\n", aa);
      printf("Vuelos BA: %d\n", ba);
      printf("Vuelos IB: %d\n", ib);

      fclose(f); // se cierra el fichero
      return 0;
}
```

3. Se desea crear una clase en C++ denominada **CMatrizCuadrada** que permita crear matrices cuadradas. La clase tendrá las siguientes variables miembro privadas:

- `int n;` // dimensión de la matriz
- `float *pmatriz;` // vector dinámico unidimensional para almacenar los //valores de la matriz

La matriz se almacenará como un vector unidimensional. Por ejemplo, la matriz $\begin{bmatrix} 30 & 2 \\ 4 & 5 \end{bmatrix}$ se almacenaría como *pmatriz={30,2,4,5}* y *n=2*.

Se pide:

(a) Implementar el constructor que recibe como parámetro la dimensión de la matriz. *(0.5 puntos)*

(b) Implementar el destructor. *(0.5 puntos)*

(c) Implementar el operador = *(0.75 puntos)*

(d) Implementar el método Diagonal que debe devolver una nueva matriz con todos los elementos a 0, excepto los elementos que hay en la diagonal *(1.25 punto)*

Solución:

(a)

```
CMatrizCuadrada::CMatrizCuadrada(int  t)
```

```
{
      n=t;
      pmatriz= new float [n*n];
}
```

(b)

```
CMatrizCuadrada::~CMatrizCuadrada (void)
{
      delete []pmatriz;
}
```

(c)

```
CMatrizCuadrada& CMatrizCuadrada::operator=
                                  (const CMatrizCuadrada  &m)
{
      n=m.n;
      delete []pmatriz;
      pmatriz=new floaf [n*n];
      for(int i=0;i<n*n;i++) pmatriz[i]=m.pmatriz[i];
      return (*this);
}
```

(d)

```
CMatrizCuadrada CMatrizCuadrada::Diagonal()
{
      CMatrizCuadrada diag(n);
      int i;

      for(i=0;i<n*n;i++) diag.pmatriz[i]=0;
      for(i=0;i<n*n;i+=n+1) diag.pmatriz[i]=pmatriz[i];

      return(diag);
}
```

4. Realizar un programa que se encargue de leer por teclado la hora en formato europeo y la muestre en formato americano. En el formado europeo la hora será introducida como una cadena con este formato: *hh:mm*, donde *hh* puede ser un valor entre 00 y 23, y *mm* puede ser un valor entre 00 y 59. Se considerará que el usuario introduce correctamente la hora (no es necesario realizar hacer comprobaciones de formato). La hora será mostrada en formato americano de esta forma: *hh:mm am* ó *hh:mm pm*, donde *hh* será un valor entre 0 y 12, *mm* será un valor entre 0 y 59, y a continuación se mostrará *am* si la hora en formato europeo era menor a 12:00, y se mostrará *pm* si la hora en formato era mayor a 12:00 y menor o igual a 23:59. Se puede hacer uso de la función **`atoi(s)`**, que conviente la cadena **`s`** en un número entero. Ejemplo de ejecución (en negrita aparece la hora introducida por el usuario):

(2 puntos)

```
Introduce hora: 11:34
Hora en formato americano: 11:34 am

Introduce hora: 14:01
Hora en formato americano: 2:01 pm

Introduce hora: 12:59
Fecha en formato americano: 12:59 pm
```

Solución:

```
#include <stdio.h>
#include <stlib.h>

void main (void)
{
      char cadena[6], hora[3], min[3];
      int h,m;
      printf("Introduce hora:");
      gets(cadena);

      for(i=0;cad[i]!=':';i++) hora[i]=cadena[i];
      hora[i]='\0';
      for(i++,j=0;cad[i]!='\n';i++,j++) min[j]=cadena[i];
      min[j]='\0';

      h=atoi(hora);
      m=atoi(min);

      if(h<=11) printf("%s am\n",cadena);
      else if(h==12) printf ("%s pm\n",cadena);
      else printf("%d:%d pm\n",h-12,m);

}
```

5. Explicar las diferentes formas de pasar parámetros a un método o una función en C++. Poner un ejemplo de cada uno

(1 pto)

Solución:

a) Paso por valor: Se pasa una copia del valor de la variable, de forma que aunque se modifique el valor dentro de la función, la variable original no se modifica. Ejemplo:

```
int a=10;
void funcion(int valor){
     valor=valor+5;
}
Llamada a la función: función(a);
```

```
Después de ejecutarse la función: a=10;
```

b) Paso por dirección: Se pasa la dirección de memoria en la que se almacena la variable, de forma que si se modifica su valor dentro de la función, la variable original también se modifica. Requiere el uso de punteros.
Ejemplo:

```
int a=10;
void funcion(int *valor){
     *valor=*valor+5;
}
Llamada a la función: función(&a);
Después de ejecutarse la función: a=15;
```

c) Paso por referencia: Se pasa una referencia de la variable, de forma que si se modifica el valor de la variable dentro de la función, la variable original también se modifica. No requiere el uso de punteros.
Ejemplo:

```
int a=10;
void funcion(int &valor){
     valor=valor+5;
}
Llamada a la función: función(a);
Después de ejecutarse la función: a=15;
```