# EXAMEN DE SISTEMAS INFORMÁTICOS INDUSTRIALES
## (SOLUCIÓN) (TEORÍA)

**JUNIO 2014**

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

(a) En el lenguaje C, el tipo de datos *int* ocupa la misma memoria independientemente del sistema en el que se use. **F**
(b) Considerando: *int a[10]; int *pa;* Se puede conseguir que el puntero *pa* apunte al vector *a* de cualquiera de estas formas: 1) *pa=a;* 2) *pa=&a[0]* **V**
(c) En C no es posible crear un vector de estructuras, ya que sólo se pueden crear vectores de tipos de datos simples **F**
(d) Una función virtual pura se debe definir obligatoriamente en la clase base, aunque se puede volver a definir en las clases derivadas. **F**
(e) No puede utilizarse *printf* para mostrar una cadena de caracteres por pantalla si ésta contiene espacios en blanco. **F**
(f) Una función global en C++ puede acceder a los datos privados de un objeto si la función se declara como amiga (*friend*) dentro de la clase. **V**
(g) El operador + en una clase se puede definir como un método de la clase o como una función global. **V**
(h) Para comunicar dos equipos mediante sockets, sólo se puede utilizar el protocolo TCP. **F**

*(2 puntos)*

**2.** Crear un programa en C con las siguientes características:

- Pida al usuario por teclado el numero de personas que quiere introducir. Para cada persona solicitará el nombre y la edad. Se considerará que el nombre de la persona tiene siempre menos de 100 caracteres.
- Tenga una función denominada *GenerarFichero* encargada de generar un fichero denominado "datos.txt" en el que se escriba:
  - En la primera línea, el nombre y la edad de la persona mayor.
  - En la segunda línea, el nombre y la edad de la persona menor.
  - En el resto de líneas, se pondrán el nombre y la edad del resto de personas en el orden en el que fueron introducidas.

  Esta función devolverá -1 si no se ha podido generar el fichero o 0 en caso contrario.
- Observaciones: Se considerará que siempre se introducen al menos los datos de 2 personas y que la edad mayor y menor no se repiten.
- Ejemplo de ejecución (en negrita aparecen los datos introducidos por el usuario):

*(3 puntos)*

```
Introduce numero de personas: 4
Introduce nombre: Brian Kernighan
Introduce edad: 50
Introduce nombre: Dennis Ritchie
Introduce edad: 23
Introduce nombre: Harvey Deitel
Introduce edad: 64
Introduce nombre: Bjarne Stroustup
Introduce edad: 75
```

A partir de estos datos, se generaría el siguiente fichero "datos.txt":

```
Mayor: Bjarne Stroustup 75
Menor: Dennis Ritchie 23
Brian Kernighan 50
Harvey Deitel 64
```

---

*Solución:*

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

typedef struct{
      char nombre[100];
      int edad;
}persona;

int GenerarFichero(persona *vector, int tam, int mayor, int
menor);

void main(void)
{
      persona *personas;
      int num,i,iMayor,iMenor,mayor=0,menor=200,res;

      printf("Introduzca numero de personas: ");
      scanf("%d",&num);

      personas=(persona *)malloc(num*sizeof(persona));
      if(personas!=NULL){
            for(i=0;i<num;i++){
              printf("Introduzca el nombre:");
              gets(personas[i].nombre);
              printf("Introduzca la edad: ");
              scanf("%d",&personas[i].edad);
              if(personas[i].edad>mayor){
                  mayor=personas[i].edad;
                  iMayor=i;
              }
              if(personas[i].edad<menor){
                  menor=personas[i].edad;
                  iMenor=i;
              }
            }
            res=GenerarFichero(personas,num,iMayor,iMenor);
            if(res==-1){
                  printf("No se ha podido generar el fichero);
            }
```

```
            free(personas);
      }
}

int GenerarFichero(persona *vector, int tam, int mayor, int
menor)
{
      int i;
      FILE *pFich;
      fich=fopen("datos.txt","w");
      if(pFich==NULL){
            printf("Error al abrir el fichero para escribir\n");
            return(-1);
      }
      fprintf(pFich,"Mayor: %s %d\n",vector[mayor].nombre,
               vector[mayor].edad);
      fprintf(pFich,"Menor: %s %d\n",vector[menor].nombre,
               vector[menor].edad);
      for(i=0;i<tam;i++){
       if(i!=mayor && i!=menor){
        fprintf(pFich,"%s %d\n",vector[i].nombre,vector[i].edad);
       }
      }
      fclose(pFich);
      return(0);
}
```

**3.** Considérese que se ha creado una clase denominada *CConjunto* para trabajar con conjuntos de números enteros. La declaración de esta clase es la siguiente:

```
const int MAX=100;
class CConjunto
{
private:
      int tam;
      int numeros[MAX];
public:
      CConjunto(void);
      CConjunto(const CConjunto &con);
      ~CConjunto(void);
      CConjunto& operator=(const CConjunto &con);
      int IntroducirNumero(int numero);
      int EliminarNumero(int numero);
};
```

donde:

- La variable miembro *numeros* es un vector que representa los números actuales en el conjunto (recuérdese que en un conjunto no puede haber datos repetidos) y *tam* representa el número actual de datos en el conjunto. Por ejemplo, si en un momento dato el vector de números es *{5,15,43,6}*, el valor de *tam* sería 4.

- La constante *MAX* indica el número máximo de elementos que puede tener un conjunto.
- El método *IntroducirNumero* añade un número al conjunto si no estaba dentro previamente (devolviendo en tal caso un 1) o no lo añade si ya estaba dentro o si el número de elementos del vector coincide con MAX (devolviendo en tal caso un valor de 0).
- El método *EliminarNumero* elimina un número del conjunto si estaba dentro (devolviendo en tal caso un 1). Si no estaba dentro el número devuelve 0.

Considerando que puede hacerse uso de cualquier método de la clase, se pide:

(a) Definir el operador + de forma que realice la unión de dos conjuntos *(1 punto)*

(b) Definir el operador * de forma que realice la intersección de dos conjuntos *(2 puntos)*

---

*Solución:*

(a)

Opción 1: Definir el operador como un método de la clase

```
CConjunto CConjunto::operator+(const CConjunto &con)
{
      CConjunto union(*this);
      for(int i=0;i<con.tam;i++){
            union.IntroducirNumero(con.numeros[i]);
      }
      return(union);
}
```

En este caso habría que incluir el prototipo del operador dentro de la declaración de la clase: `CConjunto operator+(const CConjunto &con);`

Opción 2: Definir el operador como una función global

```
CConjunto operator+(const CConjunto &con1, const CConjunto
&con2)
{
      CConjunto union(con1);
      for(int i=0;i<con2.tam;i++){
            union.IntroducirNumero(con2.numeros[i]);
      }
      return(union);
}
```

En este caso habría que definir la función como amiga de la clase, de forma que dentro de la declaración de la clase se añadiría:

```
friend CConjunto operator+(const CConjunto &con1, const
CConjunto &con2);
```

(b)

Opción 1: Definir el operador como un método de la clase

```
CConjunto CConjunto::operator*(const CConjunto &con)
{
      CConjunto interseccion;
      for(int i=0;i<tam;i++){
         for(int j=0;j<con.tam;j++){
            if(numeros[i]==con.numeros[j]){
                interseccion.IntroducirNumero(numeros[i]);
            }
         }
      }
      return(interseccion);
}
```

En este caso habría que incluir el prototipo del operador dentro de la declaración de la clase: `CConjunto operator*(const CConjunto &con);`

Opción 2: Definir el operador como una función global

```
CConjunto operator*(const CConjunto &con1, const CConjunto
&con2)
{
      CConjunto interseccion;
      for(int i=0;i<con1.tam;i++){
         for(int j=0;j<con2.tam;j++){
            if(con1.numeros[i]==con2.numeros[j]){
                interseccion.IntroducirNumero(con1.numeros[i]);
            }
         }
      }
      return(interseccion);
}
```

En este caso habría que definir la función como amiga de la clase, de forma que dentro de la declaración de la clase se añadiría:

```
friend CConjunto operator*(const CConjunto &con1, const
CConjunto &con2);
```

**4.** Escribir un programa que pida al usuario que introduzca una cadena de caracteres por teclado, almacene la cadena de caracteres en un array y la convierta en una cadena cifrada aplicando estos cambios: la 'a' se convierta en 'e', la 'e' en 'i', la 'i' en 'o', la 'o' en 'u', y la 'u' en 'a' (el mismo cifrado se aplicaría para el caso de las vocales mayúsculas). Para almacenar la cadena de caracteres utilizar un array de caracteres de tamaño 100. Ejemplo de ejecución (en negrita aparece la cadena introducida por el usuario):

*(2 puntos)*

```
Introduce cadena: Esto Es Una Cadena De Prueba
Cadena cifrada: Istu Is Ane Cedine Di Praibe
```

---

*Solución:*

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

void main (void)
{
      char cadena[100];
      int i;
      printf("Introduce cadena:");
      gets(cadena);
      for(i=0;cadena[i]!='\0';i++){
            if(cadena[i]=='a') cadena[i]='e';
            else if (cadena[i]=='e') cadena[i]='i';
            else if (cadena[i]=='i') cadena[i]='o';
            else if (cadena[i]=='o') cadena[i]='u';
            else if (cadena[i]=='u') cadena[i]='a';
            else if (cadena[i]=='A') cadena[i]='E';
            else if (cadena[i]=='E') cadena[i]='I';
            else if (cadena[i]=='I') cadena[i]='O';
            else if (cadena[i]=='O') cadena[i]='U';
            else if (cadena[i]=='U') cadena[i]='A';
      }
      printf("Cadena cifrada: %s\n", cadena);
}
```