EXAMEN DE SISTEMAS INFORMÁTICOS INDUSTRIALES
(TEORÍA)

DICIEMBRE 2015

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

a. Una función global amiga de una clase (friend) puede acceder a los miembros de la clase como si fuera un método de la clase. **V**
b. En C++, una variable miembro estática es común para todos los objetos de la clase (no hay una variable estática diferente para cada objeto de la clase, sino que todos los objetos comparten la misma variable estática). **V**
c. En C no es posible leer un fichero carácter a carácter. **F**
d. En C y C++ es posible reservar memoria dinámica para un array bidimensional. **V**
e. C++ permite la herencia múltiple, es decir, una clase derivada puede heredar de más de una clase base. **V**
f. Al declarar que una función recibe como parámetro un vector, no se puede dejar vacío el tamaño del vector. Por ejemplo, esta declaración daría error al compilar: void LeerVector(float v[]). **F**
g. En C++ se pueden dar valores por defecto a una función, de forma que dichos parámetros pueden ser omitidos en la llamada a la función. **V**
h. En C++ no se puede definir el operador + como una función global. **F**
i. En C no es posible abrir un fichero para escritura y lectura, sino que o bien se abre para lectura o bien para escritura. **F**
j. Para comunicar dos equipos mediante sockets, sólo se puede utilizar el protocolo TCP. **F**

(2.5 puntos)

2. Crear un programa que lea de un fichero denominado "datos.txt" un listado de números enteros, los almacene en un vector dinámico y a continuación muestre por pantalla de mayor a menor los números pares del vector. Observaciones:

a. El fichero "datos.txt" tiene la siguiente estructura: el primer elemento corresponde al número de enteros del listado, de forma que a continuación se encuentran los enteros. Por ejemplo:

b. La salida por pantalla considerando el fichero de ejemplo anterior sería:

   24 8 4

c. A continuación se muestra el pseudocódigo del algoritmo de la burbuja para ordenar un vector de números reales de menor a mayor:

```
ALGORITMO burbuja
variables enteras i,j,N
vector de reales v

para i=1 hasta N-1
  para j=N-1 hasta i
    si (v[j-1] > v[j])
       intercambiar v[j-1] y v[j]
    fsi
  fpara
fpara
FIN ALGORITMO
```

(2.5 puntos)

Solución:

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

void main (void)
{
      int *vector,num,i,j,aux;
      FILE *fich;

      fich=fopen("datos.txt","r"); // se abre el fichero
      if(fich==NULL){
            printf("Error: no existe el fichero\n");
      }
      else{
            fscanf(fich,"%d",&num);
            // Se reserva memoria para el vector
            vector=(int *)malloc(num*sizeof(int));
            if(vector==NULL){
                  printf("Error al reservar memoria");
            }
            else{
                  // Se guardan en el vector los datos del fichero
                  for(i=0;i<num;i++){
                        fscanf(fich,"%d",&vector[i]);
                  }
                  fclose(fich); // se cierra el fichero

                  // Se ordena el vector de mayor a menor
                  for(i=1;i<num;i++){
                        for(j=num-1;j>=i;j--){
                              if (vector[j-1]<vector[j]){
                                    aux = vector[j-1];
                                    vector[j-1] = vector[j];
                                    vector[j] = aux;
                              }
                        }
```

```
                }

                // Se muestran por pantalla los números pares
                for(i=0;i<num;i++){
                    if(vector[i]%2==0){
                        printf("%d ",vector[i]);
                    }
                }
                free(vector); // se libera la memoria del vector
            }
        }

}
```

3. Escribir un programa que solicite una frase (de máximo 100 caracteres) y a continuación una secuencia (de máximo 15 caracteres). El objetivo del código es buscar la secuencia en la frase indicando en qué posiciones coincide y cuántas coincidencias totales ha encontrado. Se diferenciará entre mayúsculas y minúsculas. A continuación se muestra un ejemplo de ejecución (en negrita aparece la información introducida por el usuario):

```
Introduzca una frase: Es evidente que desde el viaje de ayer deseemos
aprender a decir la verdad.
Introduzca una secuencia para buscar dentro de la frase: de
Coincidencia de la secuencia de encontrada en la posición: 7
Coincidencia de la secuencia de encontrada en la posición: 17
Coincidencia de la secuencia de encontrada en la posición: 20
Coincidencia de la secuencia de encontrada en la posición: 32
Coincidencia de la secuencia de encontrada en la posición: 40
Coincidencia de la secuencia de encontrada en la posición: 54
Coincidencia de la secuencia de encontrada en la posición: 60
Número total de coincidencias encontradas 7
Presione una tecla para continuar . . .
```

(2.5 puntos)

Solución:

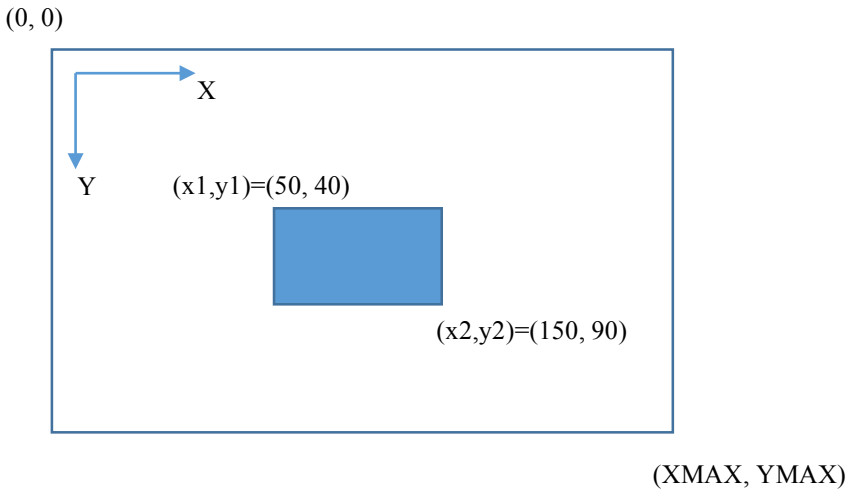
```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{

char frase[100], palabra[15];
int len_frase, len_palabra, total_encontradas=0, i,j,iguales;

printf("Introduzca una frase: ");
gets(frase);
printf("Introduzca una secuencia para buscar dentro de la frase: ");
scanf("%s",&palabra);

len_frase=strlen(frase);
len_palabra=strlen(palabra);
```

```
for (i=0;i<len_frase-len_palabra+1;i++) {
    iguales=1;
 for (j=0;j<len_palabra;j++) {
     if (frase[i+j]!=palabra[j]) {
            iguales=0;
            break;
        }
    }
    if (iguales==1) {
     printf("Coincidencia de la secuencia %s encontrada en la
posicion: %d\n",palabra,i+1);
     total_encontradas++;
    }
}
printf("Numero total de coincidencias encontradas:
%d\n",total_encontradas);
  system("PAUSE");
}
```

4. Para implementar un programa de diseño asistido por computador, se desea crear una clase en C++ denominada **CRectangulo** que permita definir rectángulos a partir de las coordenadas, en píxeles, de la esquina superior izquierda y de la esquina inferior derecha del rectángulo. La clase tendrá las siguientes variables miembro privadas:

- `int x1,y1;` //coordenadas (x,y) de la esquina superior izquierda
- `int x2,y2;` //coordenadas (x,y) de la esquina inferior derecha

Se considera que las coordenadas de los rectángulos deben estar comprendidas dentro de los límites del área de dibujo. A continuación se muestra el área de dibujo (donde XMAX e YMAX son constantes que no es necesario definir) y un ejemplo de rectángulo:



Se pide:

(a) Implementar el constructor que recibe cuatro parámetros de tipo entero, correspondiente a las coordenadas **x** e **y** de la esquina superior izquierda e inferior derecha del rectángulo. Si el rectángulo no se puede crear por que las coordenadas no son correctas (se excede el área de dibujo o las coordenadas de la esquina superior izquierda no son inferiores a las de la esquina inferior derecha), se mostrará un mensaje de error por la pantalla. Ejemplo:
`CRectangulo r1(50,40,150,90);`
En este caso (x1, y1) = (50, 40), y (x2, y2) = (150, 90)
(0.75 puntos)

(b) Implementar el destructor. *(0.25 puntos)*

(c) Implementar el método `Perimetro` que debe devolver el perímetro del rectángulo. Por ejemplo, si se ejecutara `r1.Perimetro()` considerando el objeto `r1` creado anteriormente, se obtendría el valor 300. *(0.75 puntos)*

(d) Implementar el método `Area` que debe devolver el área del rectángulo. Por ejemplo, si se ejecutara `r1.Area()` considerando el objeto `r1` creado anteriormente, se obtendría el valor 5000. *(0.75 puntos)*

Solución:

(a)

```
CRectangulo::CRectangulo(int a1, int b1, int a2, int b2)
{
      if(a1<0 || a2<0 || a1>XMAX || a2>XMAX || b1<0 || b2<0 ||
b1>YMAX || b2>YMAX || a1>=a2 || b1>=b2){
            cout<<"Error en las coordenadas\n";
      }
      else{
            x1=a1;
            y1=b1;
            x2=a2;
            y2=b2;
      }
}
```

(b)

```
CRectangulo::~CRectangulo(void)
{
}
```

(c)

```
int CRectangulo::Perimetro(void)
{
      int ancho, alto;
      ancho=x2-x1;
```

```
      alto=y2-y1;
      return(2*ancho+2*alto);
}

(d)
long CRectangulo::Area(void)
{
      int ancho, alto;
      ancho=x2-x1;
      alto=y2-y1;
      return(ancho*alto);
}
```