**UNIVERSIDAD DE VALLADOLID** **Departamento de Informática**
Informática Licenciatura en Matemáticas
Parcial Febrero de 2009

---

Apellidos : Nombre :

---

**AVISOS:**

- Es necesario entregar todas las hojas del enunciado del examen al finalizar, se hayan respondido o no a las preguntas.
- Es obligatorio firmar todas las hojas de respuesta del examen.
- En breve estará en la página web de la asignatura una copia del enunciado del examen y la solución del mismo.
- **La solución de cada problema debe entregarse en una hoja separada**.
- **La duración de esta parte del examen es de 1 hora y 30 minutos.**

# Problemas

1. (**3 puntos**) **Realiza un algoritmo Principal y su equivalente programa en C.**

   Crea un programa que solicite al usuario dos números enteros positivos, P y Q, y que a continuación le muestre al usuario aquellos números que estén comprendidos entre P y Q y que además sean múltiplos de 3, de 7 ó de ambos. En el caso de que no haya ninguno, también debe indicarlo. Además, sólo deberá hacer las comprobaciones si Q >P.

   El programa debe repetir esta operación hasta que el usuario decida terminar.

   Debes indicar de forma razonada **¿qué tipo de esquema repetitivo estás empleando y por qué lo has elegido?**

```
PBR:Solicitar al usuario los número P y Q, y si describen un intervalo válido, [P, Q],
recorro los enteros entre P y Q, ambos inclusive, comprobando aquellos que cumplan la
condición y llevando la cuenta de cuántos lo cumplen.

Necesitaremos, además de P y Q, una variable contador, I, y un acumulador para llevar
la cuenta del número de ocurrencias, veces. Además, para comprobar si el usuario quiere
terminar o no, utilizamos la variable lógica terminar.

Existe una secuencia externa, formada por distintos valores de (P, Q y terminar)
terminar tomará el valor falso, hasta que llegue al valor cierto y terminemos.
Se ejecutará al menos una vez la solicitud de los datos --> esquema repetir-hasta que

Internamente, para cada valor válido de P y Q, tendremos la secuencia formada por
la pareja: (I, veces)

P <= I <= Q
0 <= veces <= Q - P + 1

Este bucle se ejecutará siempre Q- P + 1 veces, ya que hay que comprobar todos los
casos --> es un esquema para-hacer-fin-para.
```

```
Algoritmo Principal es
P, Q, I, veces: numérico;
terminar: lógico;
Inicio
terminar := falso;
repetir
  escribir "Dame dos enteros positivos que formen un intervalo válido:";
  leer P, Q;
  si P > 0 and Q > 0 and Q > P entonces
    veces := 0;
    para I desde P hasta Q hacer
    si I mod 3 == 0 or I mod 7 == 0 entonces
          veces := veces + 1;
          escribir I, " es divisor de 3 o de 7 o de ambos";
       fin si;
    fin para;
    si veces = 0 entonces
        escribir "No había ningún caso";
    sino
        escribir "Hubo ", veces, " casos";
    fin si;
  sino
    escribir "No forman un intervalo valido";
  fin si
  escribir "Quiere terminar (cierto -> si, falso -> no):";
  leer terminar;
hasta que terminar;
Fin
```

Programa principal equivalente en C:

```
#include <stdio.h>

main(){
int P, Q, i, veces, terminar=0;
do {
 printf("\nDame dos numeros enteros positivos que formen un intervalo valido:");
 scanf("%d %d", &P, &Q);
 if (P > 0 && Q > 0 && Q > P){
  veces = 0;
  for (i=P; i<= Q; i++) {
    if (i % 3 == 0  || i % 7 == 0) {
      printf("\n%d es divisor de 3, de 7 o de ambos",i);
      veces++;
    }
  };
  if (veces == 0) printf("\nNo hubo ningun caso");
  else printf ("\nHubo %d casos", veces);
 }
 else
  printf("\n%d y %d no forman un intervalor valido\n", P, Q);

 printf("\nQuieres terminar: 0 es no, 1 es si:");
 scanf("%d", &terminar);
} while (terminar == 0);
}
```

2. **(1,5 puntos) Función en C.**

Disponemos de un vector de números reales, con un máximo de 100 elementos. Queremos hacer una función que calcule cuántas veces aparece un número real X dentro de ese vector, y si aparece al menos una vez que muestre por pantalla las posiciones en las que aparece.

La función debe responder a las siguientes especificaciones en código algorítmico:

```
Algoritmo cuenta_apariciones (X, datos, N, cuantas, existe) es
X: numérico; {real; parámetro dato; dato a buscar en el vector}
Datos: vector[1..100] de numérico; {reales; parámetro dato;
  vector con los datos donde hay que buscar las ocurrencias de X}
N: numérico; {parámetro dato; entero; posiciones ocupadas del vector datos,
  1 <= N <= 100}
Cuantas: numérico; {entero; parámetro resultado; número de veces que aparece el
  elemento X en V; 0 <= cuantas <= N}
Existe: lógico; {parámetro resultado; será cierto si X aparece al menos una vez
  en Datos; falso en caso contrario}
```

Debes indicar de forma razonada:

**¿cómo se transforma esta cabecera en una función en C: qué tipo de función es, cómo se especifican los parámetros en C?** y **¿qué tipo de esquema repetitivo estás empleando y por qué lo haces?**

```
Hay dos parámetros resultado: cuantas y existe. Podemos elegir cualquiera como tipo de la
función. Elegimos existe para responder cierto/falso sobre el funcionamiento de la operación.
Como en C no existen lógicos, se utilizará tipo int, que devolverá 1 para cierto y 0 para
falso.
Para guardar el valor de existe, utilizaremos una variable local, int existe.

PBR: Recorremos todas los posiciones del vector, entre 1 y N en código algorítmico,
entre 0 y N-1 en C. Para ello utilizaremos una variable indice, i.

Para cada elemento de la secuencia (i, Datos(i), cuantas, existe, X) comprobamos si
Datos(i)== X. En ese caso aumentamos en uno el valor de cuantas y ponemos el valor
de existe a cierto.
```

```
/* Codigo de la funcion C */
int cuenta_apariciones (float X, float *Datos, int N,  int *cuantas)
{
  int existe=0;
  int i;
  *cuantas=0;
  for (i=0; i < N; i++)
    if (Datos[i] == X) {
      ++*cuantas;
      printf("\nAparece %f en la posicion %d", Datos[i], i+1);
      existe = 1;
      };
  if (cuantas == 0) printf("\nNo aparece ninguna vez.");
  return existe;
}
```

```
/* Un ejemplo de programa principal para probarlo. No era materia de examen */
#include <stdio.h>
```

```
int cuenta_apariciones();
main(){
  float D[100];
  int N, i, veces;
  float B;

  printf("\nCuantos datos quieres usar (0, termino): \n");
  scanf("%d", &N);
  while(N) {
   printf("\nDame los datos del vector:\n ");
   for (i=0; i < N; i++) scanf("%f", &D[i]);
   printf("\nDame el dato a buscar: ");
   scanf("%f", &B);
   if (cuenta_apariciones(B, D, N, &veces))
     printf("\n%f esta en el vector %d veces", B, veces);
   else
     printf("\n%f no esta en el vector", B);
   printf("\nCuantos datos quieres usar (0, termino): ");
   scanf("%d", &N);
  };
}
```