Ejercicios del Tema 4

Estructuras

1. Distancia entre puntos

Escribe un programa que calcule la distancia entre dos puntos introducidos por el usuario, y que decida el cuadrante en el que está localizado cada punto. Este programa debe emplear la estructura `punto` capaz de almacenar dos coordenadas `x` e `y` de tipo `float`, y debe utilizar dos funciones, `cuadrante` y `distancia`. La función `cuadrante` devuelve un número entero indicando el cuadrante del punto, y la función `distancia` devuelve un número real con la distancia entre dos puntos.

```
int cuadrante(punto p);
float distancia(punto p1, punto p2);
```

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

typedef struct
{
  float x;
  float y;
} punto;

int cuadrante(punto p);
float distancia(punto p1, punto p2);

int main()
{
  punto a, b;
  float d;

  printf("Escribe las coordenadas del punto 1:\n");
  scanf("%f %f", &a.x, &a.y);

  printf("Escribe las coordenadas del punto 2:\n");
  scanf("%f %f", &b.x, &b.y);

  printf("El punto 1 está en el cuadrante %i.\n",
        cuadrante(a));

  printf("El punto 2 está en el cuadrante %i.\n",
        cuadrante(b));

  d = distancia(a, b);

  printf("La distancia entre los puntos es %f.\n", d);

}

int cuadrante(punto p)
{
  if (p.x > 0 && p.y > 0)
```

```
    return 1;
  else if (p.x < 0 && p.y > 0)
    return 2;
  else if (p.x < 0 && p.y < 0)
    return 3;
  else if (p.x > 0 && p.y < 0)
    return 4;
  else return 0; // Punto en un eje
}

float distancia(punto p1, punto p2)
{
  return sqrt(pow(p1.x - p2.x, 2) + pow(p1.y - p2.y, 2));
}
```

2. Vector de puntos

Realiza un programa que pida al usuario un número `n` de puntos indicando sus coordenadas (estructura `punto` definida en el ejercicio anterior) y los introduzca en un vector. El programa debe indicar el total de puntos que hay en cada uno de los cuadrantes.

Ejemplo de funcionamiento:

```
Indique el numero de puntos a introducir:
3
Introduzca x1, y1:
3 8
Introduzca x2, y2:
-1 8
Introduzca x3, y3:
-3 3
El total de puntos por cuadrante es:
   Cuadrante 1 = 1
   Cuadrante 2 = 2
   Cuadrante 3 = 0
   Cuadrante 4 = 0
```

```
#include <stdio.h>

typedef struct
{
  float x;
  float y;
} punto;

int cuadrante(punto p);

int main()
{
  int i, n;
  int iCuad; // indice auxiliar para cuadrantes
  // Vector contador de puntos por cuadrante. Elemento 5 para
  // puntos en ejes y origen
  int vCuad[5] = {0, 0, 0, 0, 0};
  punto puntos[100]; //Vector suficientemente grande. Es un vector de
                     //estructuras, cuyos componentes son de tipo
                     //punto.


  printf("Dime el número de puntos (< 100).\t");
  scanf("%i", &n);
  // Pido datos al usuario
  for (i = 0; i < n; i++)
    {
      printf("Punto %d:\n", i + 1);
      // Relleno el vector de puntos
      scanf("%f %f", &puntos[i].x, &puntos[i].y);
```

```
  }

 // Recorro el vector
 for (i = 0; i < n; i++)
  {
    // Con la funcion determino el cuadrante del punto
    iCuad = cuadrante(puntos[i]);
    // y actualizo el vector contador
    ++vCuad[iCuad - 1];
  }

 // Finalmente muestro los resultados
 printf("Puntos por cuadrante:\n");
 for (i = 0; i < 5; i++)
    printf("\t Cuadrante %d: %d \n",
          i + 1, vCuad[i]);
 };


int cuadrante(punto p)
{
 if (p.x > 0 && p.y > 0)
   return 1;
 else if (p.x < 0 && p.y > 0)
   return 2;
 else if (p.x < 0 && p.y < 0)
   return 3;
 else if (p.x > 0 && p.y < 0)
   return 4;
 else return 5; // Puntos sobre el eje
}
```

3. Distancia entre dos instantes temporales

Escribe un programa que calcule la distancia temporal entre dos instantes. Este programa debe emplear la estructura `tiempo` (horas, minutos, segundos) para almacenar la información de cada instante. Asimismo debe emplear la función `dTiempo` que admite como argumentos dos estructuras `tiempo` y devuelve una estructura `tiempo`. Por ejemplo, si la estructura `t1` representa el instante `3:45:15` y la estructura `t2` representa el instante `9:44:03`, la función `dTiempo` debe devolver la estructura `5:58:48`. Es importante tener en cuenta la posibilidad de que entre `t1` y `t2` se encuentre la medianoche.

```
tiempo dTiempo(tiempo t1, tiempo t2);
```

4. Contacto

Define la estructura `contacto` con los campos nombre, primer apellido, segundo apellido, y fecha de nacimiento. Esta estructura emplea internamente la estructura `fecha` para almacenar la fecha de nacimiento (día, mes y año).

Escribe un programa que lea los datos de dos contactos, y los almacene en un vector de estructuras. A continuación debe imprimir los datos del que sea más mayor, o ambos si tienen la misma edad.

El programa debe estar construido en base a dos funciones: `printContacto` debe imprimir el contenido de una estructura `contacto`; `compFecha` compara dos estructuras `fecha` y devuelve 1 cuando la primera es mayor, -1 cuando es menor, 0 cuando son iguales.

```
void printContacto(contacto persona);
int compFecha(fecha f1, fecha f2);
```