Modularización

Objetivos

- Justificar la utilidad de encapsulamiento e interfaz.
- Mostrar el mecanismo con que esto se resuelve en el lenguaje del curso.

Ejercicio 1 Fecha

(a)

Dada la siguiente representación de fecha:

```
// Registro con dia d, mes m y año a
struct rep_fecha {
  unsigned int d;
  unsigned int m;
  unsigned int a;
};
```

Implemente una función *main()* que inicialice un arreglo de fechas (por ejemplo, con las fechas 10/10/2022, 15/5/2022 y 20/11/1992), lo ordene e imprima el resultado.

(b)

Considere ahora la siguiente representación de fecha:

```
struct rep_fecha {
  unsigned int f; // AAAAMMDD
}
```

Modifique el *main()* implementado en la parte (a) de manera que se adecúe a la nueva representación.

(c)

Considere ahora que cuenta con la siguiente interfaz del tipo de datos *TFecha*:

```
typedef struct rep_fecha* TFecha;

/* Devuelve un 'TFecha' con dia d, mes m y año a */
TFecha crearFecha (unsigned int d, unsigned int m,
                   unsigned int a);
/* Devuelve true si f1 es anterior a f2, false en otro caso */
bool esAnterior (TFecha f1, TFecha f2);
/* Devuelve el día correspondiente a fecha */
unsigned int dia (TFecha fecha);
/* Devuelve el mes correspondiente a fecha */
unsigned int mes (TFecha fecha);
/* Devuelve el año correspondiente a fecha */
unsigned int anio (TFecha fecha);
```

Implemente la interfaz utilizando la representación de fecha dada en la parte (a). Reescriba la función *main()* haciendo uso de las operaciones implementadas en la interfaz y sin acceder a la representación.

(d)

Implemente la interfaz utilizando ambas representaciones de fecha.

¿Es necesario realizar cambios en la función main()?

Ejercicio 2 Punto

Un punto en un plano cartesiano se representa como un par ordenado de números, de forma que el primer número corresponde a la coordenada horizontal x y el segundo número a la coordenada vertical y.

La siguiente es la especificación del módulo `Punto`.

```
// punto.h
#ifndef _PUNTO_H
#define _PUNTO_H

/* Representación de Punto */
typedef struct rep_punto *Punto;

/* Operaciones de Punto */

/* Devuelve un 'Punto' de coordenadas 'x' e 'y'. */
Punto crearPunto(double x, double y);


/* Devuelve la coordenada 'x' de 'punto'. */
double coordX(Punto punto);

/* Devuelve la coordenada 'y' de 'punto'. */
double coordY(Punto punto);

#endif
```

(a)

Implemente en `punto.cpp` el módulo `Punto`.

(b)

Implemente otro módulo que usando el tipo `Punto`, implemente una función que calcule la distancia entre un par de puntos, y dados tres puntos, por ejemplo, (3.0, 5.5), (0, 9.5) y (-2.0, 17.5), imprima la distancia entre cada par de ellos.

Se recuerda que la distancia euclidiana en el plano entre dos puntos (x_1, y_1) y (x_2, y_2) se calcula como:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$