Tecnología de la programación
Sesión 12

Objetivos de la sesión

1. Resolver el ejercicio 9 completo (que salgan a la pizarra)
2. Introducción a punteros: ver hasta la diapositiva 18.

Guion

Solución del ejercicio 9:

1. Sumar dígitos de un entero

```
Función sumaDigitos(entero n) devuelve entero
{PRE: n>=0}
{POST: devuelve la suma de los dígitos de n}
Principio
     si n<10
          dev n
     si_no
          dev (n MOD 10 + sumaDigitos(n DIV 10))
     fsi
fin
```

2. Normalizar

```
función normalizar(entero n) devuelve entero
{PRE: n>=0}
{POST:devuelve la suma reiterada de los dígitos de n}
Principio
     si n < 10
          dev n
     si_no
          dev (normalizar(sumaDigitos(n)))
fin
```

3. Índice de normalización

```
funcion indiceNormalizacion(entero n) dev entero
{PRE: n>=0}
{POST: devuelve el índice de normalización de n}
principio
     si n < 10
          dev 1
     si_no
```

```
          dev (1 + indiceNormalizacion(sumaDigitos(n)))
fin
```

4. Normalización e índice a la vez

```
acción normalInd (ent/ entero n, ent/sal entero ind, ent/sal
entero norm)
{PRE: n>=0}
{POST: …}
Principio
     si n < 10
          ind = 1
          norm = n
     si_no
          normalInd(sumaDigitos(n), ind, norm)
          ind = 1+ind
     fsi
fin
```

Punteros:

Lo más importante es entender la idea intuitiva de punteros: son como mandos a distancia que apuntan a direcciones de memoria. Realmente son variables que contienen como valores direcciones de memoria de otras variables.

Con el operador `dirección` (escribiremos `direcc` en pseudocódigo) vamos a acceder a la dirección de memoria de una variable (sea una variable de tipo puntero o no). Con el operador `contenido` (escribiremos `cont` en pseudocódigo) vamos a acceder al contenido de una dirección de memoria. Y con el operador `destino` (escribiremos `dest` en pseudocódigo) accederemos al contenido de la dirección de memoria almacenada en el puntero.

Algunas notas sobre las transparencias:

Transparencia 15:

- Las direcciones de memoria (el `20d`, el `5d`, `28d`) son inventadas.
- Se aconseja dibujar línea a línea lo que va pasando para entenderlo mejor. Por ejemplo, cuando pone `real x = 3.5`, conviene dibujar la cajita de la variable `x`, darle una dirección de memoria y ponerle a la variable el valor 3.5.
- Cuando pone `p=q`, lo que se hace es dibujar una flecha desde `p` hacia lo mismo que apuntaba `q`.
- Respuesta a la pregunta de ¿Qué ocurre si modificamos el contenido de la dirección `28d`? ¿Cuál es el valor de `dest(p)` en cada caso?
    - Caso 1: `dest(p)` será igual al nuevo valor.

- Caso 2: `dest(p)` seguirá valiendo 4.2. De hecho, la variable y tendrá el nuevo valor, y `dest(q)` también será el nuevo valor (porque apunta a `y`).

Transparencia 16:

- Si nos damos cuenta, podemos simplificar la notación porque por lo general no nos hace falta saber la dirección de memoria.

Transparencia 17:

- ¿Qué se mostrará por pantalla?

```
entero x
puntero a entero p
p = direcc(x)
dest(p) = 7
x = dest(p)+1
escribir(dest(p))
```

Solución: un 8. Realmente se puede entender en este caso sustituyendo donde pone `dest(p)` por `x` en el código, porque `p` apunta a `x` (y por tanto, `dest(p)` es precisamente el valor de la variable `x`).