Exercici L6

Recursivitat

Fonaments de Programació II

Objectiu

L'objectiu d'aquesta pràctica és introduir-vos en la programació recursiva mitjançant la implementació de quatre funcions matemàtiques bàsiques que es poden resoldre recursivament. Aquesta pràctica us ajudarà a comprendre com es divideix un problema en subproblemes més petits i com es poden estructurar solucions recursives.

Descripció de l'exercici

Aquesta pràctica es compon de quatre exercicis progressius en dificultat. Implementareu cadascuna de les funcions seguint una estratègia recursiva. Per cada funció, haureu d'identificar el cas base (condició de parada) i el cas recursiu (com es descompon el problema en una versió més petita d'ell mateix). El primer exercici, el del factorial, està resolt pas a pas perquè pogueu veure com es dissenya una funció recursiva. La resta, l'heu de dissenyar i programar vosaltres.

Per resoldre un problema recursivament, seguiu aquestes etapes:

1. **Identifica un patró repetitiu al problema**

   Busca com es pot expressar el problema en termes d'una versió més petita del mateix problema.

2. **Defineix el cas en el qual aturem el procés (cas base)**

   Troba el cas més senzill en què podem donar una resposta directa sense recórrer a més crides recursives.

3. **Redueix el problema**

   Expressa la solució del problema en funció d'un subproblema més petit.

4. **Confia en la recursió!**

   Assumeix que la funció funciona correctament per als subproblemes més petits i centra't en com utilitzar aquest resultat per resoldre el problema més gran.

1. Factorial d'un nombre (pas a pas)

El factorial d'un nombre enter positiu `n` (`n!`) es defineix com:

$$n! = \prod_{i=1}^{n} i$$

Per exemple:

```
5! = 5 * 4 * 3 * 2 * 1 = 120
```

Anem a aplicar els passos anteriors per resoldre aquest problema.

Identificació del patró

- Si calculem el factorial manualment, veiem un patró:
  `5! = 5 × 4 × 3 × 2 × 1`, que és `5 * 4!`
  `4! = 4 × 3 × 2 × 1`, que és `4 * 3!`
  `3! = 3 × 2 × 1`, que és `3 * 2!`
- Això ens mostra que `n!` sempre es pot expressar en funció d'un subproblema més petit:
  `n! = n × (n - 1)!`
- Per exemple, el factorial de `5` es pot descompondre en:

```
5! = 5 × 4!
4! = 4 × 3!
3! = 3 × 2!
2! = 2 × 1!
1! = 1 × 0!
0! = 1  (cas base)
```

Identificació del cas base (o cas directe)

Quan es treballa amb recursió, sempre cal trobar una condició que ens permeti aturar la crida recursiva.

- En aquest cas, observem que el factorial de `0` està definit com `1`. No cal fer cap càlcul per esbrinar aquest valor.
- Això ens diu que quan `n` sigui `0`, la funció ha de retornar directament `1` sense fer més crides recursives.

Reducció del problema

Ara que sabem com expressar el factorial d'un nombre en termes d'un subproblema més petit, el següent pas és transformar aquesta relació en un procés recursiu.

- Per calcular `n!`, podem calcular `(n-1)!` i multiplicar el resultat per `n`.
- Cada pas redueix `n` en `1` fins a arribar al cas base `n = 0`, on ja sabem la resposta (`1`).
- Això vol dir que, en lloc de calcular tot el factorial directament, deleguem la feina a una versió més petita del mateix problema.
- Quan `n=5`, no calculem `5!` directament, sinó que esperem que `4!` es calculi recursivament. Això es repeteix fins que arribem a `0!`, que sabem que val `1`, i a partir d'aquí els resultats es van acumulant cap enrere.

Resum

Pas	Descripció
Identificació del patró	A l'exemple veiem com `5!` es pot expressar com `5 * 4!` Es a dir, cada nombre `n!` es pot expressar com `n * (n-1)!`
Cas base	Quan `n = 0`, retornem `1` directament.
Reducció del problema	Transformem el càlcul de `n!` en el càlcul del factorial del nombre immediatament anterior, reduint el problema de `n` a `n-1` fins a arribar a `0!`.

Mans a la feina: disseny de la funció factorial

- **Què ha de rebre?**
  Un nombre enter `n`.
- **Què retorna?**
  El factorial de `n`, calculat de manera recursiva.
- **Codi per al cas directe:**
  Si `n == 0`, retornem `1`.
- **Codi per al cas recursiu:**
  La funció es crida a si mateixa amb `n - 1` i multipliquem el resultat per `n`.

Tasques a fer

- Implementa la funció factorial, i escriu un programa principal que la cridi i que comprovi que funciona correctament.
- Per aquest exercici, contesta les preguntes següents (repassa la teoria si cal):
  - És recursivitat simple o múltiple?
  - És recursivitat final o no-final?
  - És recursivitat directa o indirecta?
  - És recursivitat subtractiva o divisiva?

2. Suma dels dígits d'un nombre

Donat un nombre enter positiu `n`, calcula la suma dels seus dígits de manera recursiva.

Per exemple:

```
suma_digits(1234) = 1 + 2 + 3 + 4 = 10
```

Especificació de la funció:

- Rep un nombre enter com a paràmetre (no una taula de caràcters!)
- Retorna la suma dels seus dígits, calculats recursivament.

Pistes per resoldre'l:

- Com podem obtenir l'últim dígit d'un nombre?

  (Pensa en una operació matemàtica que permeti extreure l'últim dígit d'un nombre enter).

- Com podem eliminar l'últim dígit d'un nombre?

  (Quina operació ens permet reduir un nombre eliminant-ne l'últim dígit?).

- Com podem expressar el problema en termes d'una versió més petita del mateix problema?

  (Intenta veure si la suma dels dígits de n es pot escriure en funció de la suma dels dígits d'un nombre més petit).

Tasques a fer

- Dissenya la funció, seguint els passos:
  - Identificar un patró
  - Identificar el cas base
  - Reduir el problema
- Implementa la funció, i escriu un programa principal que la cridi i que comprovi que funciona correctament.
- Per aquest exercici, contesta les preguntes següents (repassa la teoria si cal):
  - És recursivitat simple o múltiple?
  - És recursivitat final o no-final?
  - És recursivitat directa o indirecta?
  - És recursivitat subtractiva o divisiva?

3. Càlcul d'una potència

Donats una base `x` i un exponent enter no negatiu `n`, calcula `x^n` recursivament.

Per exemple:

```
2^4 = 2 * 2 * 2 * 2 = 16
```

Especificació de la funció:

- Rep dos nombres enters:
    - `x`: la base.
    - `n`: l'exponent (sempre un enter no negatiu).
- Retorna el valor de `x^n`, calculat de manera recursiva.

Pistes per resoldre'l:

1. Quin és el valor de qualsevol nombre elevat a 0?
   (Pensa en el cas especial x^0 que es pot resoldre sense recursió).
2. Com podem expressar x^n en funció d'una potència més petita?
   (Intenta trobar una relació entre x^n i x^(n-1)).
3. Quina operació necessitem per reduir el problema a un exponent més petit?
   (Pensa en com podem escriure x^n a partir de x^(n-1)).

Tasques a fer

- Dissenya la funció, seguint els passos:
    - Identificar un patró
    - Identificar el cas base
    - Reduir el problema
- Implementa la funció, i escriu un programa principal que la cridi i que comprovi que funciona correctament.
- Per aquest exercici, contesta les preguntes següents (repassa la teoria si cal):
    - És recursivitat simple o múltiple?
    - És recursivitat final o no-final?
    - És recursivitat directa o indirecta?
    - És recursivitat subtractiva o divisiva?

4. Seqüència de Fibonacci

La seqüència de Fibonacci és una successió de nombres on cada element es calcula a partir dels anteriors. Els primers valors de la seqüència són:

```
0, 1, 1, 2, 3, 5, 8, 13, 21, ...
```

Cada valor es calcula com la suma dels dos anteriors, excepte els dos primers, que s'han de definir manualment: `F(0) = 0` i `F(1) = 1`.

La resta de valors es calculen com:

```
F(n) = F(n - 1) + F(n - 2) per a n > 1
```

Especificació de la funció Fibonacci:

- Rep un nombre enter `n`, que indica la posició en la seqüència de Fibonacci.
- Retorna el valor de `fibonacci(n)`, calculat de manera recursiva.

Especificació del programa principal:

- La funció `fibonacci(n)` només retorna el valor de la seqüència de Fibonacci de l'enèsima posició. Per obtenir tota la seqüència des de `0` fins a `n`, serà necessari fer un bucle al programa principal.

Pistes per resoldre'l:

1. Quins són els primers dos valors de la seqüència de Fibonacci? Cal fer algun càlcul per obtenir aquests dos valors?
2. Com es pot calcular F(n) a partir de valors de F(n-1) i F(n-2)? Veus algun patró aquí? F(5) = F(4) + F(3) = (F(3) + F(2)) + (F(2) + F(1)) = 5

Tasques a fer

- Dissenya la funció, seguint els passos:
  - Identificar un patró
  - Identificar el cas base
  - Reduir el problema
- Implementa la funció, i escriu un programa principal que la cridi i que comprovi que funciona correctament.
- Per aquest exercici, contesta les preguntes següents (repassa la teoria si cal):
  - És recursivitat simple o múltiple?
  - És recursivitat final o no-final?
  - És recursivitat directa o indirecta?
  - És recursivitat subtractiva o divisiva?

Exercicis addicionals

1. Implementa una funció que donat un vector d'enters, en computi la suma de manera recursiva.

2. Implementa la sèrie de Collatz, que es defineix com:
   - Si **n** és parell, es divideix per 2.
   - Si **n** és imparell, es multiplica per 3 i se suma 1.
   - Es repeteix el procés fins que **n** arriba a 1.

3. Implementa la sèrie de Padovan, que es defineix així:

$$P(n) = \begin{cases} 1, & \text{si } n = 0, 1, 2 \\ P(n-2) + P(n-3), & \text{si } n > 2 \end{cases}$$

4. Donada la funcio `suma_digits`, transforma-la de recursiva **no-final** a recursiva **final**. Fes servir l'exemple de teoria del factorial com a referència.

5. (**difícil*) Modifica la funció de `suma_digits` **original** de manera que si el resultat de la suma dels dígits té més de dues xifres (major que 10), també en computi la suma.
   Exemple:
   `1998 = 1 + 9 + 9 + 5 = 24 → 2 + 4  = 6`