UCA Universidad de Cádiz

Grado en Ingeniería Informática
Metodología de la Programación

esther.silva@uca.es

Recursividad

Guión de prácticas

Objetivos

- Aprender a resolver problemas de forma recursiva.
- Dominar los métodos de transformación de algoritmos recursivos.

En las prácticas de este tema, el alumno deberá:

- Diseñar un subalgoritmo recursivo para todos los problemas del bloque I que se presentan a continuación.
- Implementar en C los subalgoritmos recursivos del bloque I.
- Obtener, para todos los problemas del bloque II, una función recursiva final equivalente a la función recursiva que se presenta y las correspondientes soluciones iterativas, detallando todos los pasos en cada una de las transformaciones, siguiendo los métodos de transformación explicados en la teoría de la asignatura.
- Implementar en C las versiones recursivas e iterativas del paso anterior.

Bloque I - Implementación de subalgoritmos recursivos

1. Diseñe una función recursiva que calcule la i-ésima cifra de un entero n. No se debe hacer uso de un vector.

   Implemente dicha función en C.

2. Diseñe una función que localice de forma recursiva, en la misma pasada, el máximo y el mínimo de un vector dado no vacío.

   Implemente dicha función en C.

3. Diseñe una función recursiva que devuelva el producto escalar de dos vectores de n elementos enteros, $n \geq 0$.

   Implemente dicha función en C.

4. Dado un vector ordenado crecientemente $A[1..n]$, siendo $n \geq 1$, diseñe una función recursiva que calcule la longitud de la escalera más larga, es decir, la longitud de la secuencia más larga de valores consecutivos que se encuentre en A.

   Implemente dicha función en C.

5. Diseñe una función recursiva que determine si en un vector A de n enteros existen dos parejas consecutivas de elementos tales que sus sumas sean idénticas.

   Implemente dicha función en C.

6. Dado un vector A de n enteros, diseñe una función recursiva que determine si el vector cumple la siguiente propiedad:

$$1 < i \leq \left\lfloor \frac{n}{2} \right\rfloor : A[i] = \sum_{\alpha=1}^{i-1} A[\alpha] \cdot A[n - \alpha + 1]$$

   Implemente dicha función en C.

7. Diseñe una función recursiva que devuelva cuántos elementos de una matriz A de nxm elementos enteros cumplen que es mayor que el resto de los elementos de su fila y menor que el resto de elementos de su columna o viceversa, es decir, es menor que el resto de los elementos de su fila y mayor que el resto de los elementos de su columna.

   Implemente dicha función en C.

8. Dado un vector $A[1..n]$ de n enteros estrictamente positivos, siendo $n \geq 1$, diseñe una función recursiva que obtenga el número de parejas (j, k) que cumplan:

$$1 \leq j, k \leq n : \sum_{\alpha=1}^{j} A[\alpha] = \sum_{\beta=k}^{n} A[\beta]$$

   Implemente dicha función en C.

9. Diseñe en pseudocódigo un procedimiento recursivo que reciba como parámetro un valor n y genere una matriz simétrica M, de esa dimensión, cuya triangular inferior se forma de la siguiente manera:

   a) Comienza con el número 1 en la primera fila.

   b) Cada nueva fila comienza con el último elemento de la fila anterior.

   c) Los restantes elementos de esa nueva fila se generan cada uno como la suma del elemento de la columna anterior y el elemento de columna y fila anteriores.

   d) Repetir pasos 2 y 3 hasta el valor la fila n-ésima.

   Un ejemplo para $n = 5$ generaría la triangular inferior:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 \\ 2 & 3 & 5 & 0 & 0 \\ 5 & 7 & 10 & 15 & 0 \\ 15 & 20 & 27 & 37 & 52 \end{pmatrix}$$

Teniendo en cuenta que se solicita una matriz simétrica, la función devolvería la siguiente matriz:

$$\begin{pmatrix} 1 & 1 & 2 & 5 & 15 \\ 1 & 2 & 3 & 7 & 20 \\ 2 & 3 & 5 & 10 & 27 \\ 5 & 7 & 10 & 15 & 37 \\ 15 & 20 & 27 & 37 & 52 \end{pmatrix}$$

Implemente dicho procedimiento en C.

Bloque II - Transformación de subalgoritmos recursivos

10. Utilizando las técnicas de transformación estudiadas, obtenga (detallando todos los pasos) una versión recursiva final equivalente a la siguiente función recursiva no final, así como las correspondientes versiones iterativas a partir de cada una de las versiones recursivas:

    //Cabecera: entero sumaVect_rec(E Vect: x, E entero: n, E entero: i)
    //Precondición: $x = A[1..n] \land n > 0 \land 0 \leq i \leq n$
    //Postcondición: devuelve el valor $\sum_{\alpha=1}^{i} x[\alpha]$

    entero: **función** sumaVect_rec(E Vect: x, E entero: n, E entero: i)
    inicio
        si $i = 0$ **entonces**
            devolver 0
        si_no
            devolver $x[i]$ + sumaVect_rec$(x, n, i-1)$
        fin_si
    fin_función

11. Utilizando las técnicas de transformación estudiadas, obtenga (detallando todos los pasos) una versión recursiva final equivalente a la siguiente función recursiva no final, así como las correspondientes versiones iterativas a partir de cada una de las versiones recursivas:

    //Cabecera: entero sumVects_rec(E Vect: x, E Vect: y, E entero: n, E entero: i)
    //Precondición: $x = A[1..n] \land y = B[1..n] \land i \leq n \land i \geq 1 \land n > 0$
    //Postcondición: devuelve el valor $\sum_{\alpha=i}^{n} \frac{\alpha!}{i!}(x[\alpha] \cdot y[n - \alpha + 1])$

    entero: **función** sumVects_rec(E Vect: x, E Vect: y, E entero: n, E entero: i)
    inicio
        si $i = n$ **entonces**
            devolver $x[i] \cdot y[n - i + 1]$
        si_no
            devolver $x[i] \cdot y[n - i + 1] + (i + 1) \cdot$ sumVects_rec$(x, y, n, i + 1)$
        fin_si
    fin_función

12. Utilizando las técnicas de transformación estudiadas, obtenga (detallando todos los pasos) una versión recursiva final equivalente a la siguiente función recursiva no final, así como las correspondientes versiones iterativas a partir de cada una de las versiones recursivas:

```
//Cabecera: entero prodSum_rec(E Vect: x, E Vect: y, E entero: n, E entero: i)
//Precondición: x = A[1..n] ∧ y = B[1..n] ∧ 1 ≤ i ≤ n + 1 ∧ n > 0
//Postcondición: devuelve el valor ∏_{α=i}^{n} 6 · (x[α] + y[α])

entero: función prodSum_rec(E Vect: x, E Vect: y, E entero: n, E entero: i)
inicio
    si i > n entonces
        devolver 1
    si_no
        devolver (6 · x[i] + 6 · y[i]) · prodSum_rec(x, y, n, i + 1)
    fin_si
fin_función
```

13. Utilizando las técnicas de transformación estudiadas, obtenga (detallando todos los pasos) una versión recursiva final equivalente a la siguiente función recursiva no final, así como las correspondientes versiones iterativas a partir de cada una de las versiones recursivas:

```
//Cabecera: entero prodVects_rec(E Vect: x, E Vect: y, E entero: n, E entero: i)
//Precondición: x = A[1..n] ∧ y = B[1..n] ∧ i ≤ n + 1 ∧ i ≥ 1 ∧ n > 0
//Postcondición: devuelve el valor ∑_{α=i}^{n} 3^{α−i}(x[α] · y[n − α + 1])

entero: función prodVects_rec(E Vect: x, E Vect: y, E entero: n, E entero: i)
inicio
    si i = n + 1 entonces
        devolver 0
    si_no
        devolver x[i] · y[n − i + 1] + 3 · prodVects_rec(x, y, n, i + 1)
    fin_si
fin_función
```

NOTA.- Se supone la existencia del tipo Vect definido como:

vector[N] de entero: Vect, siendo $n \leq N$.