Instrucciones de control de programa

Para poder realizar la mayoría de los programas que representan métodos numerícos de resolución de problemas matemáticos, se deben emplear sentencias de control.
Este tipo de sentencias permiten realizar fundamentalmente operaciones de forma cíclica. La sustitución regresiva de un sistema triangular superior consta de n etapas. Todas ellas siguen una ley de recurrecia común, basta con indicarla una vez y pedir que se realice n veces. Los métodos iterativos, necesitan de un número indeterminado, a priori, de ciclos para conseguir una buena aproximación a la solución del problema.
Todos estos problemas se resuelven con sentencias de control.

Sentencia Do

Su estructura es la siguiente: Do[expresión,{i,imin, imax,di}]
Realiza la expresión que se indica dentro del comando desde imin hasta imax con un incremento di.
Si no se indica el incremento di se considera di=1

```
sum = 0;
```

```
Do[sum = sum + i, {i, 1000}];
```

```
sum
```

```
500 500
```

En el ejemplo que se ha realizado se calcula el sumatorio de i desde i = 1 hasta 1000,

como ya es sabido $\sum_{i=1}^{n} i = \frac{n^2}{2} + \frac{n}{2}$

```
1000^2/2 + 1000/2
```

```
500 500
```

```
Sum[i, {i, 1000}]
```

```
500 500
```

Sentencia While

Su estructura es la siguiente: While[test,ejecución]
Realiza todo lo indicado en la parte de ejecución, tantas sentencias como se desee, mientras se cumpla la

condición del test.
Si no se indica el incremento di se considera di=1

```
sum = 0;
i = 0;
```

```
While[i ≤ 1000, sum = sum + i; i = i + 1;]
```

```
sum
```

```
500 500
```

Se puede terminar el ciclo antes de que deje de cumplirse la condición del test con el comando Break[ ]

```
sum = 0;
i = 0;
```

```
While[i ≤ 1000, sum = sum + i; If[sum > 10 000, Break[]]; i = i + 1;]
```

```
sum
```

```
10 011
```

```
i
```

```
141
```

En este ejemplo se controla que el sumatorio supere 10 000, y en ese momento se sale del ciclo while, almacenando el valor de i así como la suma realizada. Por tanto $\sum_{i=1}^{141} i = 10\,011$

Sentencia For

Su estructura es la siguiente: For[inicio,test,incremento,ejecución]
Se trata de un ciclo que tiene un inicio, se realizan todas las instrucciones del cuerpo ejecución mientras se cumpla el test y se puede incrementar la variable inicio una cantidad en el apartado incremento. En ejecucion se pueden realizar todas las intrucciones que se desee mientras estén separados por ; Se puede salir del ciclo con la sentencia Break[ ]
El incremento es opcional.

```
sum = 0;
```

```
For[i = 1, i ≤ 1000, i++, sum = sum + i]
```

```
sum
```

```
500 500
```

Se puede eliminar la parte de incremento y posteriormente incrementar la variable de control dentro del cuerpo de ejecución

```
sum = 0;
```

```
For[i = 1, i ≤ 1000, sum = sum + i; i = i + 1;]
```

```
sum
```

```
500 500
```

Ejemplo: Sustitución regresiva

El proceso de sustitución regresiva se puede programar mediante una sentencia de tipo Do, While, For, (sentencias ciclicas).

La ley de recurrencia es : $x_k = \frac{b_k - \sum_{j=k+1}^{n} a_{kj} * x_j}{a_{kk}}$, válido desde k = n ... ... 1.

Implementaremos este algoritmo con un ejemplo.

```
a = {{1, 1, 2}, {0, 1, -1}, {0, 0, -4}}
```

```
{{1, 1, 2}, {0, 1, -1}, {0, 0, -4}}
```

```
a // TableForm
```

```
1   1   2
0   1   -1
0   0   -4
```

```
b = {1, 0, -1}
```

```
{1, 0, -1}
```

```
n = Length[b]
```

```
3
```

```
x = Table[0, {i, 1, n}]
```

```
{0, 0, 0}
```

Con el comando For:

```
For[k = n, k ≥ 1, k--, x[[k]] = (b[[k]] - ∑_{j=k+1}^{n} a[[k, j]] * x[[j]]) / a[[k, k]];]
```

```
x
```

$$\left\{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right\}$$

```
a.x
```

```
{1, 0, -1}
```

Con el comando While

No es necesario utilizar un vector solución x, se pueden ir almacenandio los resultados en el propio vector de términos independientes b,así a medida que se va resolviendo el problema pasamos de tener el vector b a tener la solución del problema. Esto permite disminuir los requerimientos de memoria del problema, dado que el sistema de ecuaciones puede ser de grán dimensión.

```
a = {{1, 1, 2}, {0, 1, -1}, {0, 0, -4}}
```

```
{{1, 1, 2}, {0, 1, -1}, {0, 0, -4}}
```

```
b = {1, 0, -1}
```

```
{1, 0, -1}
```

```
n = Length[b]
```

```
3
```

```
k = n
```

```
3
```

```
While[k ≥ 1, b[[k]] = (b[[k]] - ∑_{j=k+1}^{n} a[[k, j]] * b[[j]]) / a[[k, k]]; k = k - 1;]
```

```
b
```

$$\left\{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right\}$$

Con el comando Do

```
a = {{1, 1, 2}, {0, 1, -1}, {0, 0, -4}}
```

```
{{1, 1, 2}, {0, 1, -1}, {0, 0, -4}}
```

```
b = {1, 0, -1}
```

```
{1, 0, -1}
```

```
n = Length[b]
```

```
3
```

```
Do[b[[k]] = (b[[k]] - ∑_{j=k+1}^{n} a[[k, j]] * b[[j]]) / a[[k, k]], {k, n, 1, -1}]
```

```
b
```

$$\left\{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right\}$$