

Olimpiada estatal de informática

Sesiones en línea

Martín Ibarra Romero

Obtener “N”

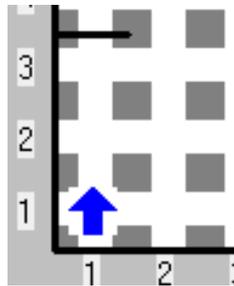
Hoy veremos porque cuando deseamos obtener un valor de “n” este se tiene cuando ya no se cumple la recursividad. Y vamos a resolver un problema utilizando recursividad y utilizando parámetros.

Recuerde que es responsabilidad de usted decidir como resuelve un problema.

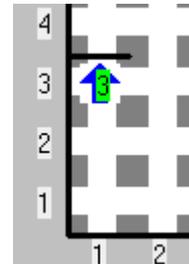
Midiendo la distancia

Tu tarea es hacer un programa que mida la distancia que hay desde donde inicia karel hasta donde esta la pared

Ejemplo



Solución



Programa recursivo

iniciar-programa

define-nueva-instruccion mide como inicio

si frente-libre entonces inicio

avanza;

mide;

fin;

deja-zumbador;

fin;

inicia-ejecucion

mide;

apagate;

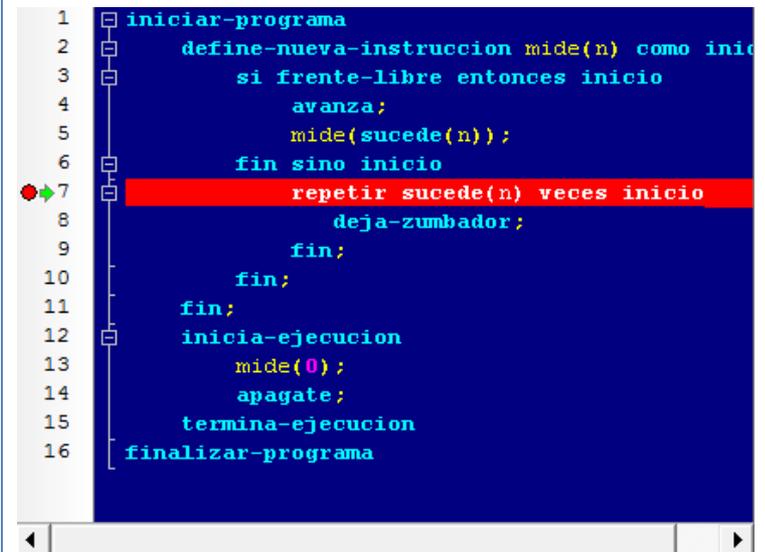
termina-ejecucion

finalizar-programa

Programa con parámetros

```
iniciar-programa
  define-nueva-instruccion mide(n) como inicio
    si frente-libre entonces inicio
      avanza;
      mide(sucede(n));
    fin sino inicio
      repetir sucede(n) veces inicio
        deja-zumbador;
      fin;
    fin;
  fin;
inicia-ejecucion
  mide(0);
  apagate;
  termina-ejecucion
finalizar-programa
```

Observe que medir se llama 3 veces pero recursivamente solo 2 por eso $n=2$



```
1  iniciar-programa
2      define-nueva-instruccion mide(n) como inicio
3          si frente-libre entonces inicio
4              avanza;
5              mide(sucede(n));
6          fin sino inicio
7          repetir sucede(n) veces inicio
8              deja-zumbador;
9          fin;
10         fin;
11     fin;
12     inicia-ejecucion
13         mide(0);
14         apagate;
15         termina-ejecucion
16     finalizar-programa
```

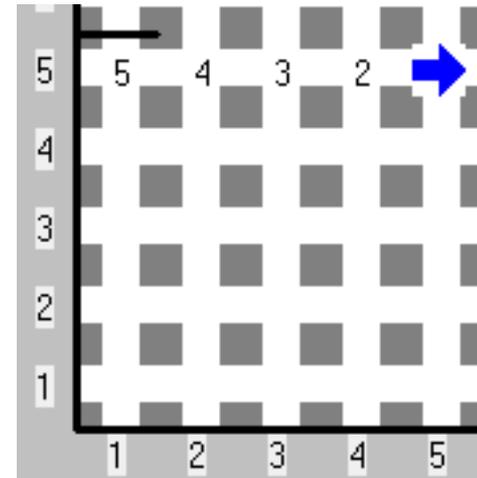
MIDE(0) -- desde línea13
MIDE(1) -- desde línea5
MIDE(2) -- desde línea5

```

iniciar-programa
  define-nueva-instruccion mide(n) como inicio
    si frente-libre entonces inicio
      avanza;
      mide(sucede(n));
      repetir sucede(n) veces inicio
        deja-zumbador;
      fin;
      avanza;
    fin sino inicio
      repetir 3 veces inicio
        gira-izquierda;
      fin;
    fin;
  fin;
  inicia-ejecucion
    mide(1);
    apagate;
  termina-ejecucion
finalizar-programa

```

Revisando el parámetro



Se modifico el programa para mostrar los regresos de Recursividad y mide se manda Llamar con valor de 1:

El 4, 3, 2, 1 representan
 Los zumbadores de Sucede(n)
 De los regresos de recursividad

Ejecución por partes 1

```
1  iniciar-programa
2  define-nueva-instruccion mide(n) como inicio
3  si frente-libre entonces inicio
4  avanza;
5  mide(sucede(n));
6  repetir sucede(n) veces inicio
7  deja-zumbador;
8  fin;
9  avanza;
10 fin sino inicio
11 repetir 3 veces inicio
12   gira-izquierda;
13   fin;
14 fin;
15 fin;
16 inicia-ejecucion
17   mide(1);
18   apagate;
19   termina-ejecucion
20 finalizar-programa
```

MIDE(1) -- desde línea17
MIDE(2) -- desde línea5
MIDE(3) -- desde línea5
MIDE(4) -- desde línea5
MIDE(5) -- desde línea5

Detenemos el programa cuando se termina la recursividad

Como se puede ver mide tiene el valor 5 (4 de los sucedes(n) recursivos, Más 1 del valor con que se invoco)

Ejecución por partes 2

Observe que después del regreso de recursividad el parámetro disminuye de 5 a 4

The image shows a programming environment with a code editor on the left and a grid on the right. The code editor has a dark blue background with white text. The code is as follows:

```
1  iniciar-programa
2      define-nueva-instruccion mide(n) como inicio
3          si frente-libre entonces inicio
4              avanza;
5              mide(sucede(n));
6          repetir sucede(n) veces inicio
7              deja-zumbador;
8          fin;
9  avanza;
10 fin sino inicio
11 repetir 3 veces inicio
12     gira-izquierda;
13     fin;
14 fin;
15 fin;
16 inicia-ejecucion
17     mide(1);
18     apagate;
19 termina-ejecucion
20 finalizar-programa
```

Red highlights are present on lines 9 and 11. A red arrow points to line 9, and a red dot is on line 11. A vertical line on the left side of the code editor indicates the current execution point, with a red dot on line 9 and a red arrow pointing to line 11.

Below the code editor, there is a console window with the following output:

```
MIDE(1) -- desde línea17
MIDE(2) -- desde línea5
MIDE(3) -- desde línea5
MIDE(4) -- desde línea5
```

On the right side, there is a grid with 15 rows and 5 columns. The rows are numbered 1 to 15 on the left. A blue arrow points to the cell at row 5, column 5.

Ejecución por partes 3

Observe que después del regreso de recursividad el parámetro disminuye de 4 a 3

The screenshot displays a code editor on the left and a grid visualization on the right. The code editor shows a program with 20 lines. Lines 9 and 11 are highlighted in red. A red arrow points to line 9, and a red dot is on line 11. The grid visualization shows a 15x3 grid with a blue arrow pointing right from the cell at row 5, column 2. The grid is labeled with numbers 1 to 15 on the y-axis and 1 to 3 on the x-axis.

```
1  iniciar-programa
2      define-nueva-instruccion mide(n) como inicio
3      si frente-libre entonces inicio
4          avanza;
5          mide(sucede(n));
6      repetir sucede(n) veces inicio
7          deja-zumbador;
8      fin;
9  avanza;
10     fin sino inicio
11     repetir 3 veces inicio
12         gira-izquierda;
13     fin;
14     fin;
15     fin;
16     inicia-ejecucion
17         mide(1);
18         apagate;
19     termina-ejecucion
20     finalizar-programa
```

MIDE(1) -- desde línea17
MIDE(2) -- desde línea5
MIDE(3) -- desde línea5

Ejecución por partes 4

Observe que después del regreso de recursividad el parámetro disminuye de 3 a 2

The screenshot displays a programming environment with a code editor on the left and a grid on the right. The code editor shows a program with 20 lines. Lines 9 and 11 are highlighted in red. A green arrow points to line 9, and a red dot is on line 11. The grid on the right has a vertical axis from 1 to 15 and a horizontal axis from 1 to 5. A blue arrow points to the cell at row 5, column 3.

```
1  iniciar-programa
2      define-nueva-instruccion mide(n) como inicio
3      si frente-libre entonces inicio
4          avanza;
5          mide(sucede(n));
6      repetir sucede(n) veces inicio
7          deja-zumbador;
8      fin;
9  avanza;
10 fin sino inicio
11 repetir 3 veces inicio
12     gira-izquierda;
13     fin;
14 fin;
15 fin;
16 inicia-ejecucion
17     mide(1);
18     apagate;
19 termina-ejecucion
20 finalizar-programa
```

MIDE(1) -- desde línea17
MIDE(2) -- desde línea5

Ejecución por partes 5

Después del regreso de recursividad el parámetro disminuye de 2 a 1

The image shows a programming environment with a code editor on the left and a grid on the right. The code editor has a blue background and contains the following code:

```
1  iniciar-programa
2      define-nueva-instruccion mide(n) como inicio
3          si frente-libre entonces inicio
4              avanza;
5              mide(sucede(n));
6          repetir sucede(n) veces inicio
7              deja-zumbador;
8          fin;
9  avanza;
10 fin sino inicio
11 repetir 3 veces inicio
12     gira-izquierda;
13     fin;
14 fin;
15 fin;
16 inicia-ejecucion
17     mide(1);
18     apagate;
19 termina-ejecucion
20 finalizar-programa
```

Red arrows point to lines 9 and 11. The grid on the right is a 15x5 grid of gray squares. The vertical axis is labeled 2 to 15. The horizontal axis is labeled 5, 4, 3, 2. A blue arrow points to the square at row 5, column 2.

MIDE(1) -- desde línea17

Conclusión

Si nos interesa un valor ya sea de zumbadores, de medir pasos, etc. este valor se tiene cuando no se cumple la condición de realizar los llamados recursivos

- Cuando requerimos tener un valor los parámetros son la base
- La recursividad es la base de los procesos de manipulación de números
- Los parámetros sirven como un auxiliar en la elaboración de programas
- Es importante conocer los parámetros y su uso para poder saber cuando lo utilizamos