Trabajo Práctico (Individual y Obligatorio)

Intérprete de Rust en Clojure

Rust es un lenguaje de programación **compilado** de propósito general. Se trata de un lenguaje multiparadigmático, ya que soporta tanto la programación declarativa (funcional) como la programación imperativa (estructurada y orientada a objetos). Rust enfatiza el rendimiento, la seguridad de tipos y la concurrencia. También refuerza la seguridad de la memoria, es decir, que todas las referencias apuntan a una memoria válida, sin requerir el uso de un *recolector de basura* o el recuento de referencias presente en otros lenguajes. Para ello, durante la compilación, Rust rastrea la vida útil de los objetos y el alcance de todas las referencias en un programa.

Desde el primer lanzamiento estable en enero de 2014, Rust ha sido adoptado por empresas como Amazon, Discord, Dropbox, Facebook (Meta), Google (Alphabet) y Microsoft. Actualmente está siendo desarrollado por la *Rust Foundation* (formada en 2021 por AWS, Huawei, Google/Alphabet, Microsoft y Mozilla) y es uno de los lenguajes de programación más usados para la programación de sistemas y a la hora de trabajar con criptomonedas y crear nodos para minar criptoactivos.

Para correr un mismo programa en diversos sistemas operativos, una práctica común consiste en compilar su código fuente en las distintas plataformas donde se lo desee ejecutar. Para no tener que hacer esto, otra opción es utilizar un **intérprete** que corra en una *máquina virtual* que se encuentre disponible para los diferentes sistemas, por ejemplo, la JVM *(Java Virtual Machine*). Precisamente, el objetivo de este trabajo práctico es **desarrollar en Clojure un intérprete de Rust que corra en la JVM**.

A los efectos de desarrollar el intérprete solicitado, se deberá partir de los siguientes materiales proporcionados por la cátedra en el aula virtual de la materia en el campus FIUBA:

- Apunte de la cátedra: *Interpretación de programas de computadora*, donde se explican la estructura y el funcionamiento de los distintos tipos de intérpretes posibles, en particular la *interpretación compilativa*, que es la estrategia a utilizar en este trabajo práctico.
- Apunte de la cátedra: *Clojure*, donde se resume el lenguaje a utilizar para el desarrollo.
- Tutorial: *Pasos para crear un proyecto en Clojure usando Leiningen*, donde se indica cómo desarrollar un proyecto dividido en código fuente y pruebas, y su despliegue como *archivo jar*.
- Código fuente de un intérprete de Rust sin terminar, para completarlo. El mismo contiene una función que debe ser terminada y 19 funciones que deben desarrollarse por completo.
- Archivos `main01.rs`, `main02.rs` … (hasta `main10.rs`) para interpretar.

Con este trabajo práctico, se espera que las/los estudiantes adquieran conocimientos profundos sobre el proceso de interpretación de programas y el funcionamiento de los intérpretes de lenguajes de programación y que, a la vez, pongan en práctica los conceptos del paradigma de *Programación Funcional* vistos durante el cuatrimestre.

Para aprobar la cursada, se deberá entregar **hasta el 30/11/2022** (por e-mail a dcorsi@fi.uba.ar o a corsi@mail.com un **proyecto** compuesto por el código fuente del intérprete de Rust en Clojure y las pruebas de las funciones desarrolladas (como *mínimo*, las pruebas correspondientes a los ejemplos que acompañan el código fuente del intérprete sin terminar proporcionado por la cátedra).

Al momento de rendir la evaluación final de la materia, se deberá modificar el intérprete presentado en este trabajo práctico, incorporándole alguna funcionalidad adicional.

Funcionamiento del intérprete de Rust al procesar los archivos main??.rs

Pantalla de inicio:

```
Interprete de RUST en Clojure
Trabajo Practico de 75.14/95.48 - Lenguajes Formales - 2022

Inspirado en: rustc 1.64.0 (2022-09-22)

Lista de comandos posibles:
AYUDA: volver a este menu
SALIR: volver al REPL de Clojure
ESCAN <archivo>: mostrar los tokens de un programa escrito en Rust
VIRTU <archivo>: mostrar la RI de un programa escrito en Rust
INTER <archivo>: interpretar la RI de un programa escrito en Rust

Rust>_
```

Rust> escan main01.rs

```
fn main ( )
{
  println! ( "- Hola, mundo!" ) ;
  print! ( "- My name is {}, James {}.\n- Hello, {}{}{}!" , "Bond" , "Bond" , - 2 + 2 , 0 , 3 + 2 * 2 ) ;
  println! ( ) ;
  println! ( "- Hasta la vista, Baby!\t\tI'll be back..." ) ;
  println! ( "{}" , if true
  {
    "- Lo dudo!\t\t\tBye!"
  }
  else
  {
    "- Obviamente!"
  }
  )
}
```

Rust> virtu main01.rs

```
0 [CAL 2]
1 HLT
2 [PUSHFI "- Hola, mundo!"]
3 [PUSHFI 1]
4 OUT
5 NL
6 [PUSHFI "- My name is {}, James {}.\n- Hello, {}{}{}!"]
7 [PUSHFI "Bond"]
8 [PUSHFI "Bond"]
9 [PUSHFI 2]
10 NEG
11 [PUSHFI 2]
12 ADD
13 [PUSHFI 0]
14 [PUSHFI 3]
15 [PUSHFI 2]
16 [PUSHFI 2]
17 MUL
18 ADD
19 [PUSHFI 6]
20 OUT
21 [PUSHFI 0]
22 OUT
23 NL
24 [PUSHFI "- Hasta la vista, Baby!\t\tI'll be back..."]
25 [PUSHFI 1]
26 OUT
27 NL
28 [PUSHFI "{}"]
29 [PUSHFI true]
30 [JC 32]
```

```
31 [JMP 34]
32 [PUSHFI "- Lo dudo!\t\t\tBye!"]
33 [JMP 35]
34 [PUSHFI "- Obviamente!"]
35 [PUSHFI 2]
36 OUT
37 NL
38 RETN
```

Rust> inter main01.rs

```
- Hola, mundo!
- My name is Bond, James Bond.
- Hello, 007!
- Hasta la vista, Baby!         I'll be back...
- Lo dudo!                      Bye!
```

Rust> escan main02.rs

```
use std :: io ;
use std :: io :: Write ;
fn main ( )
{
  println! ( "*************************************************************" ) ;
  println! ( "Se ingresan dos valores enteros, se muestra su producto." ) ;
  println! ( "Se utiliza el algoritmo de 'multiplicacion por duplicacion'." ) ;
  println! ( "(Metodo campesino ruso de multiplicacion)" ) ;
  println! ( "*************************************************************" ) ;
  print! ( "x: " ) ;
  io :: stdout ( ) . flush ( ) . expect ( "Error de escritura!" ) ;
  let mut renglon : String = String :: new ( ) ;
  io :: stdin ( ) . read_line ( & mut renglon ) . expect ( "Error de lectura!" ) ;
  let mut x : i64 = renglon . trim ( ) . parse :: < i64 > ( ) . expect ( "Se esperaba un numero entero!" ) ;
  let mut x_cambio : bool = false ;
  if x < 0
  {
    x = - x ;
    x_cambio = true ;
  }
  print! ( "y: " ) ;
  io :: stdout ( ) . flush ( ) . expect ( "Error de escritura!" ) ;
  renglon = String :: new ( ) ;
  io :: stdin ( ) . read_line ( & mut renglon ) . expect ( "Error de lectura!" ) ;
  let mut y : i64 = renglon . trim ( ) . parse :: < i64 > ( ) . expect ( "Se esperaba un numero entero!" ) ;
  let mut y_cambio : bool = false ;
  if y < 0
  {
    y = - y ;
    y_cambio = true ;
  }
  let mut prod : i64 = 0 ;
  while y > 0
  {
    if y % 2 != 0
    {
      prod += x ;
    }
    x *= 2 ;
    y /= 2 ;
  }
  if x_cambio
  {
    prod = - prod ;
  }
  if y_cambio
  {
    prod = - prod ;
  }
  println! ( "x*y={}" , prod ) ;
}
```

Rust> virtu main02.rs

```
0 [CAL 2]
1 HLT
2 [PUSHFI "***********************************************************"]
3 [PUSHFI 1]
4 OUT
5 NL
6 [PUSHFI "Se ingresan dos valores enteros, se muestra su producto."]
7 [PUSHFI 1]
8 OUT
9 NL
10 [PUSHFI "Se utiliza el algoritmo de 'multiplicacion por duplicacion'."]
11 [PUSHFI 1]
12 OUT
13 NL
14 [PUSHFI "(Metodo campesino ruso de multiplicacion)"]
15 [PUSHFI 1]
16 OUT
17 NL
18 [PUSHFI "***********************************************************"]
19 [PUSHFI 1]
20 OUT
21 NL
22 [PUSHFI "x: "]
23 [PUSHFI 1]
24 OUT
25 FLUSH
26 [PUSHFI ""]
27 [POP 0]
28 [IN 0]
29 [PUSHFM 0]
30 TOI
31 [POP 1]
32 [PUSHFI false]
33 [POP 2]
34 [PUSHFM 1]
35 [PUSHFI 0]
36 LT
37 [JC 39]
38 [JMP 44]
39 [PUSHFM 1]
40 NEG
41 [POP 1]
42 [PUSHFI true]
43 [POP 2]
44 [PUSHFI "y: "]
45 [PUSHFI 1]
46 OUT
47 FLUSH
48 [PUSHFI ""]
49 [POP 0]
50 [IN 0]
51 [PUSHFM 0]
52 TOI
53 [POP 3]
54 [PUSHFI false]
55 [POP 4]
56 [PUSHFM 3]
57 [PUSHFI 0]
58 LT
59 [JC 61]
60 [JMP 66]
61 [PUSHFM 3]
62 NEG
63 [POP 3]
64 [PUSHFI true]
65 [POP 4]
66 [PUSHFI 0]
67 [POP 5]
68 [PUSHFM 3]
69 [PUSHFI 0]
70 GT
71 [JC 73]
72 [JMP 87]
73 [PUSHFM 3]
74 [PUSHFI 2]
75 MOD
```

```
76 [PUSHFI 0]
77 NEQ
78 [JC 80]
79 [JMP 82]
80 [PUSHFM 1]
81 [POPADD 5]
82 [PUSHFI 2]
83 [POPMUL 1]
84 [PUSHFI 2]
85 [POPDIV 3]
86 [JMP 68]
87 [PUSHFM 2]
88 [JC 90]
89 [JMP 93]
90 [PUSHFM 5]
91 NEG
92 [POP 5]
93 [PUSHFM 4]
94 [JC 96]
95 [JMP 99]
96 [PUSHFM 5]
97 NEG
98 [POP 5]
99 [PUSHFI "x*y={}"]
100 [PUSHFM 5]
101 [PUSHFI 2]
102 OUT
103 NL
104 RETN
```

Rust> inter main02.rs

```
************************************************************
Se ingresan dos valores enteros, se muestra su producto.
Se utiliza el algoritmo de 'multiplicacion por duplicacion'.
(Metodo campesino ruso de multiplicacion)
************************************************************
x: 12
y: 4
x*y=48
```

Rust> escan main03.rs

```
use std :: io ;
use std :: io :: Write ;
use std :: process ;
fn dividir ( x : i64 , y : i64 , q : & mut i64 , r : & mut i64 )
{
  if y == 0
  {
    println! ( "ERROR: Division por cero!" ) ;
    process :: exit ( 1 ) ;
  }
  * q = 0 ;
  * r = x ;
  if * r < 0
  {
    * r = - * r ;
  }
  let v : i64 ;
  let mut w : i64 ;
  if y >= 0
  {
    v = y ;
    w = y ;
  }
  else
  {
    v = - y ;
    w = - y ;
  }
  while w <= * r
  {
    w *= 2 ;
  }
  while w > v
  {
    * q *= 2 ;
    w /= 2 ;
    if w <= * r
    {
      * r -= w ;
      * q += 1 ;
    }
  }
  if x < 0
  {
    * r = - * r ;
    * q = - * q ;
  }
  if y < 0
  {
    * q = - * q ;
  }
}
fn mostrar_salida ( cociente : i64 , resto : i64 )
{
  println! ( "Cociente: {}" , cociente ) ;
  println! ( "Resto: {}" , resto ) ;
}
fn main ( )
{
  println! ( "**************************************************************" ) ;
  println! ( "Se ingresan dos valores enteros, se muestra su cociente." ) ;
  println! ( "Se utiliza el algoritmo 'desplazar y restar' (shift-subtract)." ) ;
  println! ( "**************************************************************" ) ;
  print! ( "x: " ) ;
  io :: stdout ( ) . flush ( ) . expect ( "Error de escritura!" ) ;
  let mut renglon : String = String :: new ( ) ;
  io :: stdin ( ) . read_line ( & mut renglon ) . expect ( "Error de lectura!" ) ;
  let x : i64 = renglon . trim ( ) . parse :: < i64 > ( ) . expect ( "Se esperaba un numero entero!" ) ;
  print! ( "y: " ) ;
  io :: stdout ( ) . flush ( ) . expect ( "Error de escritura!" ) ;
  renglon = String :: new ( ) ;
  io :: stdin ( ) . read_line ( & mut renglon ) . expect ( "Error de lectura!" ) ;
  let y : i64 = renglon . trim ( ) . parse :: < i64 > ( ) . expect ( "Se esperaba un numero entero!" ) ;
  let mut q : i64 = 0 ;
  let mut r : i64 = 0 ;
  dividir ( x , y , & mut q , & mut r ) ;
  mostrar_salida ( q , r ) ;
}
```

Rust> virtu main03.rs

```
0 [CAL 105]
1 HLT
2 [POPARG 3]
3 [POPARG 2]
4 [POPARG 1]
5 [POPARG 0]
6 [PUSHFM 1]
7 [PUSHFI 0]
8 EQ
9 [JC 11]
10 [JMP 17]
11 [PUSHFI "ERROR: Division por cero!"]
12 [PUSHFI 1]
13 OUT
14 NL
15 [PUSHFI 1]
16 HLT
17 [PUSHFI 0]
18 [POPREF 2]
19 [PUSHFM 0]
20 [POPREF 3]
21 [PUSHREF 3]
22 [PUSHFI 0]
23 LT
```

```
24 [JC 26]
25 [JMP 29]
26 [PUSHREF 3]
27 NEG
28 [POPREF 3]
29 [PUSHFM 1]
30 [PUSHFI 0]
31 GTE
32 [JC 34]
33 [JMP 39]
34 [PUSHFM 1]
35 [POP 4]
36 [PUSHFM 1]
37 [POP 5]
38 [JMP 45]
39 [PUSHFM 1]
40 NEG
41 [POP 4]
42 [PUSHFM 1]
43 NEG
44 [POP 5]
45 [PUSHFM 5]
46 [PUSHREF 3]
47 LTE
48 [JC 50]
49 [JMP 53]
50 [PUSHFI 2]
51 [POPMUL 5]
52 [JMP 45]
53 [PUSHFM 5]
54 [PUSHFM 4]
55 GT
56 [JC 58]
57 [JMP 72]
58 [PUSHFI 2]
59 [POPMULREF 2]
60 [PUSHFI 2]
61 [POPDIV 5]
62 [PUSHFM 5]
63 [PUSHREF 3]
64 LTE
65 [JC 67]
66 [JMP 71]
67 [PUSHFM 5]
68 [POPSUBREF 3]
69 [PUSHFI 1]
70 [POPADDREF 2]
71 [JMP 53]
72 [PUSHFM 0]
73 [PUSHFI 0]
74 LT
75 [JC 77]
76 [JMP 83]
77 [PUSHREF 3]
78 NEG
79 [POPREF 3]
80 [PUSHREF 2]
81 NEG
82 [POPREF 2]
83 [PUSHFM 1]
84 [PUSHFI 0]
85 LT
86 [JC 88]
87 [JMP 91]
88 [PUSHREF 2]
89 NEG
90 [POPREF 2]
91 RETN
92 [POPARG 1]
93 [POPARG 0]
94 [PUSHFI "Cociente: {}"]
95 [PUSHFM 0]
96 [PUSHFI 2]
97 OUT
98 NL
99 [PUSHFI "Resto: {}"]
100 [PUSHFM 1]
```

```
101 [PUSHFI 2]
102 OUT
103 NL
104 RETN
105 [PUSHFI "****************************************************************"]
106 [PUSHFI 1]
107 OUT
108 NL
109 [PUSHFI "Se ingresan dos valores enteros, se muestra su cociente."]
110 [PUSHFI 1]
111 OUT
112 NL
113 [PUSHFI "Se utiliza el algoritmo 'desplazar y restar' (shift-subtract)."]
114 [PUSHFI 1]
115 OUT
116 NL
117 [PUSHFI "****************************************************************"]
118 [PUSHFI 1]
119 OUT
120 NL
121 [PUSHFI "x: "]
122 [PUSHFI 1]
123 OUT
124 FLUSH
125 [PUSHFI ""]
126 [POP 0]
127 [IN 0]
128 [PUSHFM 0]
129 TOI
130 [POP 1]
131 [PUSHFI "y: "]
132 [PUSHFI 1]
133 OUT
134 FLUSH
135 [PUSHFI ""]
136 [POP 0]
137 [IN 0]
138 [PUSHFM 0]
139 TOI
140 [POP 2]
141 [PUSHFI 0]
142 [POP 3]
143 [PUSHFI 0]
144 [POP 4]
145 [PUSHFM 1]
146 [PUSHFM 2]
147 [PUSHADDR 3]
148 [PUSHADDR 4]
149 [CAL 2]
150 [PUSHFM 3]
151 [PUSHFM 4]
152 [CAL 92]
153 RETN
```

Rust> inter main03.rs

```
****************************************************************
Se ingresan dos valores enteros, se muestra su cociente.
Se utiliza el algoritmo 'desplazar y restar' (shift-subtract).
****************************************************************
x: 23
y: 5
Cociente: 4
Resto: 3
```

Rust> escan main04.rs

```
use std :: io ;
use std :: io :: Write ;
use std :: process ;
fn mcd ( mut x : i64 , mut y : i64 ) -> i64
{
  if x <= 0 || y <= 0
  {
    println! ( "ERROR: El algoritmo requiere dos numeros enteros positivos!" ) ;
    process :: exit ( 1 ) ;
  }
  while x != y
  {
    if x < y
    {
      y -= x ;
    }
    if y < x
    {
      x -= y ;
    }
  }
  x
}
fn main ( )
{
  println! ( "******************************************************************************" ) ;
  println! ( "Se ingresan dos valores enteros positivos, se muestra su maximo comun divisor." ) ;
  println! ( "Se utiliza el algoritmo de Euclides." ) ;
  println! ( "******************************************************************************" ) ;
  print! ( "x: " ) ;
  io :: stdout ( ) . flush ( ) . expect ( "Error de escritura!" ) ;
  let mut renglon : String = String :: new ( ) ;
  io :: stdin ( ) . read_line ( & mut renglon ) . expect ( "Error de lectura!" ) ;
  let x : i64 = renglon . trim ( ) . parse :: < i64 > ( ) . expect ( "Se esperaba un numero entero!" ) ;
  print! ( "y: " ) ;
  io :: stdout ( ) . flush ( ) . expect ( "Error de escritura!" ) ;
  renglon = String :: new ( ) ;
  io :: stdin ( ) . read_line ( & mut renglon ) . expect ( "Error de lectura!" ) ;
  let y : i64 = renglon . trim ( ) . parse :: < i64 > ( ) . expect ( "Se esperaba un numero entero!" ) ;
  print! ( "{} es el MCD entre " , mcd ( x , y ) ) ;
  println! ( "{} y {}" , x , y ) ;
}
```

Rust> virtu main04.rs

```
0 [CAL 41]
1 HLT
2 [POPARG 1]
3 [POPARG 0]
4 [PUSHFM 0]
5 [PUSHFI 0]
6 LTE
7 [PUSHFM 1]
8 [PUSHFI 0]
9 LTE
10 OR
11 [JC 13]
12 [JMP 19]
13 [PUSHFI "ERROR: El algoritmo requiere dos numeros enteros positivos!"]
14 [PUSHFI 1]
15 OUT
16 NL
17 [PUSHFI 1]
18 HLT
19 [PUSHFM 0]
20 [PUSHFM 1]
21 NEQ
22 [JC 24]
23 [JMP 39]
24 [PUSHFM 0]
25 [PUSHFM 1]
26 LT
27 [JC 29]
28 [JMP 31]
29 [PUSHFM 0]
```

```
30 [POPSUB 1]
31 [PUSHFM 1]
32 [PUSHFM 0]
33 LT
34 [JC 36]
35 [JMP 38]
36 [PUSHFM 1]
37 [POPSUB 0]
38 [JMP 19]
39 [PUSHFM 0]
40 RET
41 [PUSHFI "******************************************************************************"]
42 [PUSHFI 1]
43 OUT
44 NL
45 [PUSHFI "Se ingresan dos valores enteros positivos, se muestra su maximo comun divisor."]
46 [PUSHFI 1]
47 OUT
48 NL
49 [PUSHFI "Se utiliza el algoritmo de Euclides."]
50 [PUSHFI 1]
51 OUT
52 NL
53 [PUSHFI "******************************************************************************"]
54 [PUSHFI 1]
55 OUT
56 NL
57 [PUSHFI "x: "]
58 [PUSHFI 1]
59 OUT
60 FLUSH
61 [PUSHFI ""]
62 [POP 0]
63 [IN 0]
64 [PUSHFM 0]
65 TOI
66 [POP 1]
67 [PUSHFI "y: "]
68 [PUSHFI 1]
69 OUT
70 FLUSH
71 [PUSHFI ""]
72 [POP 0]
73 [IN 0]
74 [PUSHFM 0]
75 TOI
76 [POP 2]
77 [PUSHFI "{} es el MCD entre "]
78 [PUSHFM 1]
79 [PUSHFM 2]
80 [CAL 2]
81 [PUSHFI 2]
82 OUT
83 [PUSHFI "{} y {}"]
84 [PUSHFM 1]
85 [PUSHFM 2]
86 [PUSHFI 3]
87 OUT
88 NL
89 RETN
```

Rust> inter main04.rs

```
******************************************************************************
Se ingresan dos valores enteros positivos, se muestra su maximo comun divisor.
Se utiliza el algoritmo de Euclides.
******************************************************************************
x: 16
y: 24
8 es el MCD entre 16 y 24
```

Rust> escan main05.rs

```
use std :: io ;
use std :: io :: Write ;
const TRES : i64 = 3 ;
fn es_impar_primo ( n : i64 ) -> bool
{
  let mut es_p : bool = true ;
  let limite : i64 = f64 :: sqrt ( n as f64 ) as i64 ;
  let mut d : i64 = TRES ;
  while d <= limite && es_p
  {
    if n % d == 0
    {
      es_p = false ;
    }
    d += 2 ;
  }
  es_p
}
fn main ( )
{
  println! ( "*********************************************************************************************" ) ;
  println! ( "Se ingresa un valor entero positivo, se muestran los numeros primos menores que ese valor." ) ;
  println! ( "Se utiliza una funcion booleana para determinar si un numero impar mayor que 1 es primo o no." ) ;
  println! ( "*********************************************************************************************" ) ;
  print! ( "x: " ) ;
  io :: stdout ( ) . flush ( ) . expect ( "Error de escritura!" ) ;
  let mut renglon : String = String :: new ( ) ;
  io :: stdin ( ) . read_line ( & mut renglon ) . expect ( "Error de lectura!" ) ;
  let x : i64 = renglon . trim ( ) . parse :: < i64 > ( ) . expect ( "Se esperaba un numero entero!" ) ;
  if x <= 2
  {
    print! ( "No hay numeros primos menores que {}" , x ) ;
  }
  else
  {
    print! ( "Numeros primos menores que {}: 2" , x ) ;
    let mut n : i64 = TRES ;
    while n < x
    {
      if es_impar_primo ( n )
      {
        print! ( " {}" , n ) ;
      }
      n += 2 ;
    }
  }
  println! ( ) ;
}
```

Rust> virtu main05.rs

```
0 [CAL 33]
1 HLT
2 [POPARG 0]
3 [PUSHFI true]
4 [POP 1]
5 [PUSHFM 0]
6 TOF
7 SQRT
8 TOI
9 [POP 2]
10 [PUSHFI 3]
11 [POP 3]
12 [PUSHFM 3]
13 [PUSHFM 2]
14 LTE
15 [PUSHFM 1]
16 AND
17 [JC 19]
18 [JMP 31]
19 [PUSHFM 0]
20 [PUSHFM 3]
21 MOD
22 [PUSHFI 0]
23 EQ
```

```
24 [JC 26]
25 [JMP 28]
26 [PUSHFI false]
27 [POP 1]
28 [PUSHFI 2]
29 [POPADD 3]
30 [JMP 12]
31 [PUSHFM 1]
32 RET
33 [PUSHFI "*********************************************************************************************"]
34 [PUSHFI 1]
35 OUT
36 NL
37 [PUSHFI "Se ingresa un valor entero positivo, se muestran los numeros primos menores que ese valor."]
38 [PUSHFI 1]
39 OUT
40 NL
41 [PUSHFI "Se utiliza una funcion booleana para determinar si un numero impar mayor que 1 es primo o no."]
42 [PUSHFI 1]
43 OUT
44 NL
45 [PUSHFI "*********************************************************************************************"]
46 [PUSHFI 1]
47 OUT
48 NL
49 [PUSHFI "x: "]
50 [PUSHFI 1]
51 OUT
52 FLUSH
53 [PUSHFI ""]
54 [POP 0]
55 [IN 0]
56 [PUSHFM 0]
57 TOI
58 [POP 1]
59 [PUSHFM 1]
60 [PUSHFI 2]
61 LTE
62 [JC 64]
63 [JMP 69]
64 [PUSHFI "No hay numeros primos menores que {}"]
65 [PUSHFM 1]
66 [PUSHFI 2]
67 OUT
68 [JMP 91]
69 [PUSHFI "Numeros primos menores que {}: 2"]
70 [PUSHFM 1]
71 [PUSHFI 2]
72 OUT
73 [PUSHFI 3]
74 [POP 2]
75 [PUSHFM 2]
76 [PUSHFM 1]
77 LT
78 [JC 80]
79 [JMP 91]
80 [PUSHFM 2]
81 [CAL 2]
82 [JC 84]
83 [JMP 88]
84 [PUSHFI " {}"]
85 [PUSHFM 2]
86 [PUSHFI 2]
87 OUT
88 [PUSHFI 2]
89 [POPADD 2]
90 [JMP 75]
91 [PUSHFI 0]
92 OUT
93 NL
94 RETN
```

Rust> inter main05.rs

```
*********************************************************************************************
Se ingresa un valor entero positivo, se muestran los numeros primos menores que ese valor.
Se utiliza una funcion booleana para determinar si un numero impar mayor que 1 es primo o no.
*********************************************************************************************
x: 100
Numeros primos menores que 100: 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

Rust> escan main06.rs

```
use std :: io ;
use std :: io :: Write ;
fn raiz_cuadrada_de_positivo ( x : f64 ) -> f64
{
  if x == 1.0
  {
    return 1.0 ;
  }
  let mut izq : f64 ;
  let mut der : f64 ;
  if x < 1.0
  {
    izq = x ;
    der = 1.0 ;
  }
  else
  {
    izq = 1.0 ;
    der = x ;
  }
  let mut r : f64 = ( izq + der ) / 2.0 ;
  while f64 :: abs ( x - r * r ) > 1.0E-10
  {
    if r * r < x
    {
      izq = r ;
    }
    else
    {
      der = r ;
    }
    r = ( izq + der ) / 2.0 ;
  }
  r
}
fn main ( )
{
  println! ( "**********************************************************" ) ;
  println! ( "Se ingresa un valor numerico, se muestra su raiz cuadrada." ) ;
  println! ( "Se utiliza el algoritmo de la biseccion." ) ;
  println! ( "**********************************************************" ) ;
  print! ( "x: " ) ;
  io :: stdout ( ) . flush ( ) . expect ( "Error de escritura!" ) ;
  let mut renglon : String = String :: new ( ) ;
  io :: stdin ( ) . read_line ( & mut renglon ) . expect ( "Error de lectura!" ) ;
  let x : f64 = renglon . trim ( ) . parse :: < f64 > ( ) . expect ( "Se esperaba un numero!" ) ;
  if x == 0.0
  {
    println! ( "La raiz cuadrada de 0 es 0.00000000" ) ;
  }
  else
  {
    let rc : f64 = raiz_cuadrada_de_positivo ( f64 :: abs ( x ) ) ;
    if x < 0.0
    {
      if x == - 1.0
      {
        println! ( "Las raices cuadradas de -1 son +i y -i" ) ;
      }
      else
      {
        println! ( "Las raices cuadradas de {} son +{:.8}i y -{:.8}i" , x , rc , rc ) ;
      }
    }
    else
    {
      println! ( "Las raices cuadradas de {} son +{:.8} y -{:.8}" , x , rc , rc ) ;
    }
  }
}
```

Rust> virtu main06.rs

```
0 [CAL 61]
1 HLT
2 [POPARG 0]
3 [PUSHFM 0]
4 [PUSHFI 1.0]
5 EQ
6 [JC 8]
7 [JMP 10]
8 [PUSHFI 1.0]
9 RET
10 [PUSHFM 0]
11 [PUSHFI 1.0]
12 LT
13 [JC 15]
14 [JMP 20]
15 [PUSHFM 0]
16 [POP 1]
17 [PUSHFI 1.0]
18 [POP 2]
19 [JMP 24]
20 [PUSHFI 1.0]
21 [POP 1]
22 [PUSHFM 0]
23 [POP 2]
24 [PUSHFM 1]
25 [PUSHFM 2]
26 ADD
27 [PUSHFI 2.0]
28 DIV
29 [POP 3]
30 [PUSHFM 0]
31 [PUSHFM 3]
32 [PUSHFM 3]
33 MUL
34 SUB
35 ABS
36 [PUSHFI 1.0E-10]
37 GT
38 [JC 40]
39 [JMP 59]
40 [PUSHFM 3]
41 [PUSHFM 3]
42 MUL
43 [PUSHFM 0]
44 LT
45 [JC 47]
46 [JMP 50]
47 [PUSHFM 3]
48 [POP 1]
49 [JMP 52]
50 [PUSHFM 3]
51 [POP 2]
52 [PUSHFM 1]
53 [PUSHFM 2]
54 ADD
55 [PUSHFI 2.0]
56 DIV
57 [POP 3]
58 [JMP 30]
59 [PUSHFM 3]
60 RET
61 [PUSHFI "**********************************************************"]
62 [PUSHFI 1]
63 OUT
64 NL
```

```
65 [PUSHFI "Se ingresa un valor numerico, se muestra su raiz cuadrada."]
66 [PUSHFI 1]
67 OUT
68 NL
69 [PUSHFI "Se utiliza el algoritmo de la biseccion."]
70 [PUSHFI 1]
71 OUT
72 NL
73 [PUSHFI "**********************************************************"]
74 [PUSHFI 1]
75 OUT
76 NL
77 [PUSHFI "x: "]
78 [PUSHFI 1]
79 OUT
80 FLUSH
81 [PUSHFI ""]
82 [POP 0]
83 [IN 0]
84 [PUSHFM 0]
85 TOF
86 [POP 1]
87 [PUSHFM 1]
88 [PUSHFI 0.0]
89 EQ
90 [JC 92]
91 [JMP 97]
92 [PUSHFI "La raiz cuadrada de 0 es 0.00000000"]
93 [PUSHFI 1]
94 OUT
95 NL
96 [JMP 132]
97 [PUSHFM 1]
98 ABS
99 [CAL 2]
100 [POP 2]
101 [PUSHFM 1]
102 [PUSHFI 0.0]
103 LT
104 [JC 106]
105 [JMP 125]
106 [PUSHFM 1]
107 [PUSHFI 1.0]
108 NEG
109 EQ
110 [JC 112]
111 [JMP 117]
112 [PUSHFI "Las raices cuadradas de -1 son +i y -i"]
113 [PUSHFI 1]
114 OUT
115 NL
116 [JMP 124]
117 [PUSHFI "Las raices cuadradas de {} son +{:.8}i y -{:.8}i"]
118 [PUSHFM 1]
119 [PUSHFM 2]
120 [PUSHFM 2]
121 [PUSHFI 4]
122 OUT
123 NL
124 [JMP 132]
125 [PUSHFI "Las raices cuadradas de {} son +{:.8} y -{:.8}"]
126 [PUSHFM 1]
127 [PUSHFM 2]
128 [PUSHFM 2]
129 [PUSHFI 4]
130 OUT
131 NL
132 RETN
```

Rust> inter main06.rs

```
**********************************************************
Se ingresa un valor numerico, se muestra su raiz cuadrada.
Se utiliza el algoritmo de la biseccion.
**********************************************************
x: 80
Las raices cuadradas de 80 son +8,94427191 y -8,94427191
```

Rust> escan main07.rs

```
fn mostrar_espacios ( n : i64 )
{
  let mut i : i64 = 0 ;
  while i < n
  {
    print! ( " " ) ;
    i += 1 ;
  }
}
fn main ( )
{
  println! ( "*******************************************************" ) ;
  println! ( "Se muestra una sinusoide graficada mediante asteriscos." ) ;
  println! ( "El programa no utiliza 'use'." ) ;
  println! ( "*******************************************************" ) ;
  println! ( "x\t sin(x)" ) ;
  let mut a : f64 = 0.0 ;
  let lim : f64 = 8.0 * f64 :: atan ( 1.0 ) ;
  while a < lim
  {
    let s : f64 = f64 :: sin ( a ) ;
    if s >= 0.0
    {
      print! ( "{:.2}\t {:.5}" , a , s ) ;
    }
    else
    {
      print! ( "{:.2}\t{:.5}" , a , s ) ;
    }
    mostrar_espacios ( ( 25.0 + 24.0 * s ) as i64 ) ;
    println! ( "*" ) ;
    a += 0.1 ;
  }
  print! ( "{:.2}\t {:.5}" , lim , 0.0 ) ;
  mostrar_espacios ( 25 ) ;
  println! ( "*" ) ;
}
```

Rust> virtu main07.rs

```
0 [CAL 17]
1 HLT
2 [POPARG 0]
3 [PUSHFI 0]
4 [POP 1]
5 [PUSHFM 1]
6 [PUSHFM 0]
7 LT
8 [JC 10]
9 [JMP 16]
10 [PUSHFI " "]
11 [PUSHFI 1]
12 OUT
13 [PUSHFI 1]
14 [POPADD 1]
15 [JMP 5]
16 RETN
17 [PUSHFI "*******************************************************"]
18 [PUSHFI 1]
19 OUT
```

```
20 NL
21 [PUSHFI "Se muestra una sinusoide graficada mediante asteriscos."]
22 [PUSHFI 1]
23 OUT
24 NL
25 [PUSHFI "El programa no utiliza 'use'."]
26 [PUSHFI 1]
27 OUT
28 NL
29 [PUSHFI "*******************************************************"]
30 [PUSHFI 1]
31 OUT
32 NL
33 [PUSHFI "x\t sin(x)"]
34 [PUSHFI 1]
35 OUT
36 NL
37 [PUSHFI 0.0]
38 [POP 0]
39 [PUSHFI 8.0]
40 [PUSHFI 1.0]
41 ATAN
42 MUL
43 [POP 1]
44 [PUSHFM 0]
45 [PUSHFM 1]
46 LT
47 [JC 49]
48 [JMP 82]
49 [PUSHFM 0]
50 SIN
51 [POP 2]
52 [PUSHFM 2]
53 [PUSHFI 0.0]
54 GTE
55 [JC 57]
56 [JMP 63]
57 [PUSHFI "{:.2}\t {:.5}"]
58 [PUSHFM 0]
59 [PUSHFM 2]
60 [PUSHFI 3]
61 OUT
62 [JMP 68]
63 [PUSHFI "{:.2}\t{:.5}"]
64 [PUSHFM 0]
65 [PUSHFM 2]
66 [PUSHFI 3]
67 OUT
68 [PUSHFI 25.0]
69 [PUSHFI 24.0]
70 [PUSHFM 2]
71 MUL
72 ADD
73 TOI
74 [CAL 2]
75 [PUSHFI "*"]
76 [PUSHFI 1]
77 OUT
78 NL
79 [PUSHFI 0.1]
80 [POPADD 0]
81 [JMP 44]
82 [PUSHFI "{:.2}\t {:.5}"]
83 [PUSHFM 1]
84 [PUSHFI 0.0]
85 [PUSHFI 3]
86 OUT
87 [PUSHFI 25]
88 [CAL 2]
89 [PUSHFI "*"]
90 [PUSHFI 1]
91 OUT
92 NL
93 RETN
```

```
Rust> inter main07.rs
```

```
*******************************************************
Se muestra una sinusoide graficada mediante asteriscos.
El programa no utiliza 'use'.
*******************************************************
x        sin(x)
0,00     0,00000                         *
0,10     0,09983                           *
0,20     0,19867                             *
0,30     0,29552                                *
0,40     0,38942                                  *
0,50     0,47943                                    *
0,60     0,56464                                      *
0,70     0,64422                                        *
0,80     0,71736                                          *
0,90     0,78333                                           *
1,00     0,84147                                             *
1,10     0,89121                                              *
1,20     0,93204                                               *
1,30     0,96356                                                *
1,40     0,98545                                                *
1,50     0,99749                                                *
1,60     0,99957                                                *
1,70     0,99166                                                *
1,80     0,97385                                                *
1,90     0,94630                                               *
2,00     0,90930                                              *
2,10     0,86321                                             *
2,20     0,80850                                            *
2,30     0,74571                                          *
2,40     0,67546                                         *
2,50     0,59847                                       *
2,60     0,51550                                     *
2,70     0,42738                                   *
2,80     0,33499                                 *
2,90     0,23925                              *
3,00     0,14112                            *
3,10     0,04158                         *
3,20    -0,05837                       *
3,30    -0,15775                     *
3,40    -0,25554                  *
3,50    -0,35078                *
3,60    -0,44252              *
3,70    -0,52984            *
3,80    -0,61186          *
3,90    -0,68777        *
4,00    -0,75680      *
4,10    -0,81828     *
4,20    -0,87158    *
4,30    -0,91617   *
4,40    -0,95160  *
4,50    -0,97753 *
4,60    -0,99369 *
4,70    -0,99992 *
4,80    -0,99616 *
4,90    -0,98245 *
5,00    -0,95892 *
5,10    -0,92581  *
5,20    -0,88345   *
5,30    -0,83227     *
5,40    -0,77276      *
5,50    -0,70554        *
5,60    -0,63127         *
5,70    -0,55069           *
5,80    -0,46460             *
5,90    -0,37388                *
6,00    -0,27942                  *
6,10    -0,18216                    *
6,20    -0,08309                       *
6,28     0,00000                         *
```

Rust> escan main08.rs

```
use std :: io ;
use std :: io :: Write ;
fn factorial ( n : i64 ) -> i64
{
  if n < 2
  {
    1
  }
  else
  {
    n * factorial ( n - 1 )
  }
}
fn main ( )
{
  println! ( "****************************************************" ) ;
  println! ( "Se ingresa un valor entero, se muestra su factorial." ) ;
  println! ( "Se utiliza un algoritmo recursivo." ) ;
  println! ( "****************************************************" ) ;
  print! ( "n: " ) ;
  io :: stdout ( ) . flush ( ) . expect ( "Error de escritura!" ) ;
  let mut renglon : String = String :: new ( ) ;
  io :: stdin ( ) . read_line ( & mut renglon ) . expect ( "Error de lectura!" ) ;
  let n : i64 = renglon . trim ( ) . parse :: < i64 > ( ) . expect ( "Se esperaba un numero entero!" ) ;
  if n < 0
  {
    println! ( "ERROR: El algoritmo requiere un numero entero no negativo!" ) ;
  }
  else
  {
    println! ( "{}! es {}" , n , factorial ( n ) ) ;
  }
}
```

Rust> virtu main08.rs

```
0 [CAL 17]
1 HLT
2 [POPARG 0]
3 [PUSHFM 0]
4 [PUSHFI 2]
5 LT
6 [JC 8]
7 [JMP 10]
8 [PUSHFI 1]
9 [JMP 16]
10 [PUSHFM 0]
11 [PUSHFM 0]
12 [PUSHFI 1]
13 SUB
14 [CAL 2]
15 MUL
16 RET
17 [PUSHFI "****************************************************"]
18 [PUSHFI 1]
19 OUT
20 NL
21 [PUSHFI "Se ingresa un valor entero, se muestra su factorial."]
22 [PUSHFI 1]
23 OUT
24 NL
25 [PUSHFI "Se utiliza un algoritmo recursivo."]
26 [PUSHFI 1]
27 OUT
28 NL
29 [PUSHFI "****************************************************"]
30 [PUSHFI 1]
31 OUT
32 NL
33 [PUSHFI "n: "]
```

```
34 [PUSHFI 1]
35 OUT
36 FLUSH
37 [PUSHFI ""]
38 [POP 0]
39 [IN 0]
40 [PUSHFM 0]
41 TOI
42 [POP 1]
43 [PUSHFM 1]
44 [PUSHFI 0]
45 LT
46 [JC 48]
47 [JMP 53]
48 [PUSHFI "ERROR: El algoritmo requiere un numero entero no negativo!"]
49 [PUSHFI 1]
50 OUT
51 NL
52 [JMP 60]
53 [PUSHFI "{}! es {}"]
54 [PUSHFM 1]
55 [PUSHFM 1]
56 [CAL 2]
57 [PUSHFI 3]
58 OUT
59 NL
60 RETN
```

Rust> inter main08.rs

```
****************************************************
Se ingresa un valor entero, se muestra su factorial.
Se utiliza un algoritmo recursivo.
****************************************************
n: 5
5! es 120
```

Rust> escan main09.rs

```
use std :: io ;
use std :: io :: Write ;
fn potencia_rec ( x : f64 , n : i64 ) -> f64
{
  if n == 0
  {
    return 1.0 ;
  }
  if n % 2 == 0
  {
    let m : f64 = potencia_rec ( x , n / 2 ) ;
    m * m
  }
  else
  {
    x * potencia_rec ( x , n - 1 )
  }
}
fn potencia ( x : f64 , n : i64 ) -> f64
{
  let p : f64 ;
  if n < 0
  {
    p = potencia_rec ( x , - n ) ;
    1.0 / p
  }
  else
  {
    potencia_rec ( x , n )
  }
}
```

```
fn main ( )
{
  println! ( "*******************************************************************************************" ) ;
  println! ( "Se ingresan el valor de una base y el valor entero de un exponente, se muestra la potencia." ) ;
  println! ( "Se utiliza un algoritmo recursivo." ) ;
  println! ( "*******************************************************************************************" ) ;
  print! ( "b: " ) ;
  io :: stdout ( ) . flush ( ) . expect ( "Error de escritura!" ) ;
  let mut renglon : String = String :: new ( ) ;
  io :: stdin ( ) . read_line ( & mut renglon ) . expect ( "Error de lectura!" ) ;
  let b : f64 = renglon . trim ( ) . parse :: < f64 > ( ) . expect ( "Se esperaba un numero!" ) ;
  print! ( "e: " ) ;
  io :: stdout ( ) . flush ( ) . expect ( "Error de escritura!" ) ;
  renglon = String :: new ( ) ;
  io :: stdin ( ) . read_line ( & mut renglon ) . expect ( "Error de lectura!" ) ;
  let e : i64 = renglon . trim ( ) . parse :: < i64 > ( ) . expect ( "Se esperaba un numero entero!" ) ;
  println! ( "{} elevado a la {} es {}" , b , e , potencia ( b , e ) ) ;
}
```

Rust> virtu main09.rs

```
0 [CAL 56]
1 HLT
2 [POPARG 1]
3 [POPARG 0]
4 [PUSHFM 1]
5 [PUSHFI 0]
6 EQ
7 [JC 9]
8 [JMP 11]
9 [PUSHFI 1.0]
10 RET
11 [PUSHFM 1]
12 [PUSHFI 2]
13 MOD
14 [PUSHFI 0]
15 EQ
16 [JC 18]
17 [JMP 28]
18 [PUSHFM 0]
19 [PUSHFM 1]
20 [PUSHFI 2]
21 DIV
22 [CAL 2]
23 [POP 2]
24 [PUSHFM 2]
25 [PUSHFM 2]
26 MUL
27 [JMP 35]
28 [PUSHFM 0]
29 [PUSHFM 0]
30 [PUSHFM 1]
31 [PUSHFI 1]
32 SUB
33 [CAL 2]
34 MUL
35 RET
36 [POPARG 1]
37 [POPARG 0]
38 [PUSHFM 1]
39 [PUSHFI 0]
40 LT
41 [JC 43]
42 [JMP 52]
43 [PUSHFM 0]
44 [PUSHFM 1]
45 NEG
46 [CAL 2]
47 [POP 2]
48 [PUSHFI 1.0]
49 [PUSHFM 2]
50 DIV
51 [JMP 55]
52 [PUSHFM 0]
53 [PUSHFM 1]
54 [CAL 2]
55 RET
```

```
56 [PUSHFI "******************************************************************************************"]
57 [PUSHFI 1]
58 OUT
59 NL
60 [PUSHFI "Se ingresan el valor de una base y el valor entero de un exponente, se muestra la potencia."]
61 [PUSHFI 1]
62 OUT
63 NL
64 [PUSHFI "Se utiliza un algoritmo recursivo."]
65 [PUSHFI 1]
66 OUT
67 NL
68 [PUSHFI "******************************************************************************************"]
69 [PUSHFI 1]
70 OUT
71 NL
72 [PUSHFI "b: "]
73 [PUSHFI 1]
74 OUT
75 FLUSH
76 [PUSHFI ""]
77 [POP 0]
78 [IN 0]
79 [PUSHFM 0]
80 TOF
81 [POP 1]
82 [PUSHFI "e: "]
83 [PUSHFI 1]
84 OUT
85 FLUSH
86 [PUSHFI ""]
87 [POP 0]
88 [IN 0]
89 [PUSHFM 0]
90 TOI
91 [POP 2]
92 [PUSHFI "{} elevado a la {} es {}"]
93 [PUSHFM 1]
94 [PUSHFM 2]
95 [PUSHFM 1]
96 [PUSHFM 2]
97 [CAL 36]
98 [PUSHFI 4]
99 OUT
100 NL
101 RETN
```

Rust> inter main09.rs

```
******************************************************************************************
Se ingresan el valor de una base y el valor entero de un exponente, se muestra la potencia.
Se utiliza un algoritmo recursivo.
******************************************************************************************
b: 2.5
e: 5
2,500000 elevado a la 5 es 97,656250
```

Rust> escan main10.rs

```
use std :: io ;
use std :: io :: Write ;
fn entero_a_hexa ( n : i64 ) -> String
{
  let mut hexa : String = String :: from ( "0" ) ;
  if n != 0
  {
    hexa = String :: new ( ) ;
  }
  let digitos : String = String :: from ( "0123456789ABCDEF" ) ;
  let mut cociente : i64 = n ;
  while cociente != 0
  {
    let resto : i64 = cociente % 16 ;
    let ch : char = digitos . as_str ( ) . chars ( ) . nth ( resto as usize ) . unwrap ( ) ;
    hexa = format! ( "{}{}" , ch , hexa ) ;
    cociente /= 16 ;
  }
  hexa
}
fn main ( )
{
  println! ( "*************************************************************************************************" ) ;
  println! ( "Se muestran 16 numeros en decimal y en hexadecimal, al presionar Enter se muestran 16 mas." ) ;
  println! ( "Se sale del programa escribiendo 'salir' y presionando Enter." ) ;
  println! ( "*************************************************************************************************" ) ;
  let mut opcion : String = String :: new ( ) ;
  let mut num : i64 = 0 ;
  while opcion != "salir" && opcion != "SALIR"
  {
    println! ( "\nDEC\tHEX" ) ;
    println! ( "---\t---" ) ;
    let mut cont : i64 = 0 ;
    while cont < 16
    {
      println! ( "{}\t{}" , num , entero_a_hexa ( num ) ) ;
      num += 1 ;
      cont += 1 ;
    }
    print! ( "Presione Enter o escriba 'salir' y presione Enter: " ) ;
    io :: stdout ( ) . flush ( ) . expect ( "Error de escritura!" ) ;
    opcion = String :: new ( ) ;
    io :: stdin ( ) . read_line ( & mut opcion ) . expect ( "Error de lectura!" ) ;
    opcion = opcion . trim ( ) . to_string ( ) ;
  }
}
```

Rust> virtu main10.rs

```
0 [CAL 41]
1 HLT
2 [POPARG 0]
3 [PUSHFI "0"]
4 [POP 1]
5 [PUSHFM 0]
6 [PUSHFI 0]
7 NEQ
8 [JC 10]
9 [JMP 12]
10 [PUSHFI ""]
11 [POP 1]
12 [PUSHFI "0123456789ABCDEF"]
13 [POP 2]
14 [PUSHFM 0]
15 [POP 3]
16 [PUSHFM 3]
17 [PUSHFI 0]
18 NEQ
19 [JC 21]
20 [JMP 39]
21 [PUSHFM 3]
22 [PUSHFI 16]
23 MOD
24 [POP 4]
25 [PUSHFM 2]
26 [PUSHFM 4]
27 TOI
28 CHR
29 [POP 5]
30 [PUSHFI "{}{}"]
31 [PUSHFM 5]
32 [PUSHFM 1]
33 [PUSHFI 3]
34 FMT
35 [POP 1]
36 [PUSHFI 16]
```

```
37 [POPDIV 3]
38 [JMP 16]
39 [PUSHFM 1]
40 RET
41 [PUSHFI "*********************************************************************************************"]
42 [PUSHFI 1]
43 OUT
44 NL
45 [PUSHFI "Se muestran 16 numeros en decimal y en hexadecimal, al presionar Enter se muestran 16 mas."]
46 [PUSHFI 1]
47 OUT
48 NL
49 [PUSHFI "Se sale del programa escribiendo 'salir' y presionando Enter."]
50 [PUSHFI 1]
51 OUT
52 NL
53 [PUSHFI "*********************************************************************************************"]
54 [PUSHFI 1]
55 OUT
56 NL
57 [PUSHFI ""]
58 [POP 0]
59 [PUSHFI 0]
60 [POP 1]
61 [PUSHFM 0]
62 [PUSHFI "salir"]
63 NEQ
64 [PUSHFM 0]
65 [PUSHFI "SALIR"]
66 NEQ
67 AND
68 [JC 70]
69 [JMP 107]
70 [PUSHFI "\nDEC\tHEX"]
71 [PUSHFI 1]
72 OUT
73 NL
74 [PUSHFI "---\t---"]
75 [PUSHFI 1]
76 OUT
77 NL
78 [PUSHFI 0]
79 [POP 2]
80 [PUSHFM 2]
81 [PUSHFI 16]
82 LT
83 [JC 85]
84 [JMP 97]
85 [PUSHFI "{}\t{}"]
86 [PUSHFM 1]
87 [PUSHFM 1]
88 [CAL 2]
89 [PUSHFI 3]
90 OUT
91 NL
92 [PUSHFI 1]
93 [POPADD 1]
94 [PUSHFI 1]
95 [POPADD 2]
96 [JMP 80]
97 [PUSHFI "Presione Enter o escriba 'salir' y presione Enter: "]
98 [PUSHFI 1]
99 OUT
100 FLUSH
101 [PUSHFI ""]
102 [POP 0]
103 [IN 0]
104 [PUSHFM 0]
105 [POP 0]
106 [JMP 61]
107 RETN
```

Rust> inter main10.rs

```
**********************************************************************************************
Se muestran 16 numeros en decimal y en hexadecimal, al presionar Enter se muestran 16 mas.
Se sale del programa escribiendo 'salir' y presionando Enter.
**********************************************************************************************

DEC     HEX
---     ---
0       0
1       1
2       2
3       3
4       4
5       5
6       6
7       7
8       8
9       9
10      A
11      B
12      C
13      D
14      E
15      F
Presione Enter o escriba 'salir' y presione Enter:

DEC     HEX
---     ---
16      10
17      11
18      12
19      13
20      14
21      15
22      16
23      17
24      18
25      19
26      1A
27      1B
28      1C
29      1D
30      1E
31      1F
Presione Enter o escriba 'salir' y presione Enter: salir
```

Ejemplo de interpretación: main03.rs

```
use std::io;
use std::io::Write;
use std::process;

fn dividir(x: i64, y: i64, q: &mut i64, r: &mut i64) {
    if y == 0 {
        println!("ERROR: Division por cero!");
        process::exit(1);
    }

    *q = 0;
    *r = x;
    if *r < 0 {
        *r = -*r;
    }

    let v: i64;
    let mut w: i64;
    if y >= 0 {
        v = y;
        w = y;
    } else {
        v = -y;
        w = -y;
    }

    while w <= *r {
        w *= 2;
    }

    while w > v {
        *q *= 2;
        w /= 2;
        if w <= *r {
            *r -= w;
            *q += 1;
        }
    }

    if x < 0 {
        *r = -*r;
        *q = -*q;
    }

    if y < 0 {
        *q = -*q;
    }
}

fn mostrar_salida(cociente: i64, resto: i64) {
    println!("Cociente: {}", cociente);
    println!("Resto: {}", resto);
}

fn main() {
    println!("***************************************************************");
    println!("Se ingresan dos valores enteros, se muestra su cociente.");
    println!("Se utiliza el algoritmo 'desplazar y restar' (shift-subtract).");
    println!("***************************************************************");

    print!("x: ");
    io::stdout().flush().expect("Error de escritura!");

    let mut renglon: String = String::new();
    io::stdin()
        .read_line(&mut renglon)
        .expect("Error de lectura!");
    let x: i64 = renglon
        .trim()
        .parse::<i64>()
        .expect("Se esperaba un numero entero!");

    print!("y: ");
    io::stdout().flush().expect("Error de escritura!");

    renglon = String::new();
    io::stdin()
        .read_line(&mut renglon)
        .expect("Error de lectura!");
    let y: i64 = renglon
        .trim()
        .parse::<i64>()
        .expect("Se esperaba un numero entero!");

    let mut q: i64 = 0;
    let mut r: i64 = 0;

    dividir(x, y, &mut q, &mut r);

    mostrar_salida(q, r);
}
```

```
Rust> inter main03.rs

Ambiente del interprete:
cod: [[CAL 105] HLT [POPARG 3] [POPARG 2] [POPARG 1] [POPARG 0] [PUSHFM 1] [PUSHFI 0] EQ [JC 11] [JMP 17] [PUSHFI "ERROR:
Division por cero!"] [PUSHFI 1] OUT NL [PUSHFI 1] HLT [PUSHFI 0] [POPREF 2] [PUSHFM 0] [POPREF 3] [PUSHREF 3] [PUSHFI 0] LT [JC
26] [JMP 29] [PUSHREF 3] NEG [POPREF 3] [PUSHFM 1] [PUSHFI 0] GTE [JC 34] [JMP 39] [PUSHFM 1] [POP 4] [PUSHFM 1] [POP 5] [JMP
45] [PUSHFM 1] NEG [POP 4] [PUSHFM 1] NEG [POP 5] [PUSHFM 5] [PUSHREF 3] LTE [JC 50] [JMP 53] [PUSHFI 2] [POPMUL 5] [JMP 45]
[PUSHFM 5] [PUSHFM 4] GT [JC 58] [JMP 72] [PUSHFI 2] [POPMULREF 2] [PUSHFI 2] [POPDIV 5] [PUSHFM 5] [PUSHREF 3] LTE [JC 67] [JMP
71] [PUSHFM 5] [POPSUBREF 3] [PUSHFI 1] [POPADDREF 2] [JMP 53] [PUSHFM 0] [PUSHFI 0] LT [JC 77] [JMP 83] [PUSHREF 3] NEG [POPREF
3] [PUSHREF 2] NEG [POPREF 2] [PUSHFM 1] [PUSHFI 0] LT [JC 88] [JMP 91] [PUSHREF 2] NEG [POPREF 2] RETN [POPARG 1] [POPARG 0]
[PUSHFI "Cociente: {}"] [PUSHFM 0] [PUSHFI 2] OUT NL [PUSHFI "Resto: {}"] [PUSHFM 1] [PUSHFI 2] OUT NL RETN [PUSHFI
"**************************************************************"] [PUSHFI 1] OUT NL [PUSHFI "Se ingresan dos valores enteros, se
muestra su cociente."] [PUSHFI 1] OUT NL [PUSHFI "Se utiliza el algoritmo 'desplazar y restar' (shift-subtract)."] [PUSHFI 1]
OUT NL [PUSHFI "**************************************************************"] [PUSHFI 1] OUT NL [PUSHFI "x: "] [PUSHFI 1] OUT
FLUSH [PUSHFI ""] [POP 0] [IN 0] [PUSHFM 0] TOI [POP 1] [PUSHFI "y: "] [PUSHFI 1] OUT FLUSH [PUSHFI ""] [POP 0] [IN 0] [PUSHFM
0] TOI [POP 2] [PUSHFI 0] [POP 3] [PUSHFI 0] [POP 4] [PUSHFM 1] [PUSHFM 2] [PUSHADDR 3] [PUSHADDR 4] [CAL 2] [PUSHFM 3] [PUSHFM
4] [CAL 92] RETN]

0 [CAL 105]
1 HLT
2 [POPARG 3]
3 [POPARG 2]
4 [POPARG 1]
5 [POPARG 0]
6 [PUSHFM 1]
7 [PUSHFI 0]
8 EQ
9 [JC 11]
10 [JMP 17]
11 [PUSHFI "ERROR: Division por cero!"]
12 [PUSHFI 1]
13 OUT
14 NL
15 [PUSHFI 1]
16 HLT
17 [PUSHFI 0]
18 [POPREF 2]
19 [PUSHFM 0]
20 [POPREF 3]
21 [PUSHREF 3]
22 [PUSHFI 0]
23 LT
24 [JC 26]
25 [JMP 29]
26 [PUSHREF 3]
27 NEG
28 [POPREF 3]
29 [PUSHFM 1]
30 [PUSHFI 0]
31 GTE
32 [JC 34]
33 [JMP 39]
34 [PUSHFM 1]
35 [POP 4]
36 [PUSHFM 1]
37 [POP 5]
38 [JMP 45]
39 [PUSHFM 1]
40 NEG
41 [POP 4]
42 [PUSHFM 1]
43 NEG
44 [POP 5]
45 [PUSHFM 5]
46 [PUSHREF 3]
47 LTE
48 [JC 50]
49 [JMP 53]
50 [PUSHFI 2]
51 [POPMUL 5]
52 [JMP 45]
53 [PUSHFM 5]
54 [PUSHFM 4]
55 GT
56 [JC 58]
57 [JMP 72]
58 [PUSHFI 2]
59 [POPMULREF 2]
60 [PUSHFI 2]
61 [POPDIV 5]
62 [PUSHFM 5]
63 [PUSHREF 3]
64 LTE
65 [JC 67]
66 [JMP 71]
67 [PUSHFM 5]
68 [POPSUBREF 3]
69 [PUSHFI 1]
```

```
70 [POPADDREF 2]
71 [JMP 53]
72 [PUSHFM 0]
73 [PUSHFI 0]
74 LT
75 [JC 77]
76 [JMP 83]
77 [PUSHREF 3]
78 NEG
79 [POPREF 3]
80 [PUSHREF 2]
81 NEG
82 [POPREF 2]
83 [PUSHFM 1]
84 [PUSHFI 0]
85 LT
86 [JC 88]
87 [JMP 91]
88 [PUSHREF 2]
89 NEG
90 [POPREF 2]
91 RETN
92 [POPARG 1]
93 [POPARG 0]
94 [PUSHFI "Cociente: {}"]
95 [PUSHFM 0]
96 [PUSHFI 2]
97 OUT
98 NL
99 [PUSHFI "Resto: {}"]
100 [PUSHFM 1]
101 [PUSHFI 2]
102 OUT
103 NL
104 RETN
105 [PUSHFI "**************************************************************"]
106 [PUSHFI 1]
107 OUT
108 NL
109 [PUSHFI "Se ingresan dos valores enteros, se muestra su cociente."]
110 [PUSHFI 1]
111 OUT
112 NL
113 [PUSHFI "Se utiliza el algoritmo 'desplazar y restar' (shift-subtract)."]
114 [PUSHFI 1]
115 OUT
116 NL
117 [PUSHFI "**************************************************************"]
118 [PUSHFI 1]
119 OUT
120 NL
121 [PUSHFI "x: "]
122 [PUSHFI 1]
123 OUT
124 FLUSH
125 [PUSHFI ""]
126 [POP 0]
127 [IN 0]
128 [PUSHFM 0]
129 TOI
130 [POP 1]
131 [PUSHFI "y: "]
132 [PUSHFI 1]
133 OUT
134 FLUSH
135 [PUSHFI ""]
136 [POP 0]
137 [IN 0]
138 [PUSHFM 0]
139 TOI
140 [POP 2]
141 [PUSHFI 0]
142 [POP 3]
143 [PUSHFI 0]
144 [POP 4]
145 [PUSHFM 1]
146 [PUSHFM 2]
147 [PUSHADDR 3]
148 [PUSHADDR 4]
149 [CAL 2]
150 [PUSHFM 3]
151 [PUSHFM 4]
152 [CAL 92]
153 RETN
```

mapa-regs: **{2 [[i64 nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]], 92 [[i64 nil] [i64 nil]], 105 [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]}**

```
Ambiente del interprete:
regs-de-act: []
reg-actual: nil
cont-prg: 0
fetched: [CAL 105]
pila: []

Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 105
fetched: [PUSHFI "**************************************************************"]
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 106
fetched: [PUSHFI 1]
pila: [1 "**************************************************************"]

Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 107
fetched: OUT
pila: [1 "**************************************************************" 1]

**************************************************************
Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 108
fetched: NL
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 109
fetched: [PUSHFI "Se ingresan dos valores enteros, se muestra su cociente."]
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 110
fetched: [PUSHFI 1]
pila: [1 "Se ingresan dos valores enteros, se muestra su cociente."]

Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 111
fetched: OUT
pila: [1 "Se ingresan dos valores enteros, se muestra su cociente." 1]

Se ingresan dos valores enteros, se muestra su cociente.
Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 112
fetched: NL
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 113
fetched: [PUSHFI "Se utiliza el algoritmo 'desplazar y restar' (shift-subtract)."]
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 114
fetched: [PUSHFI 1]
pila: [1 "Se utiliza el algoritmo 'desplazar y restar' (shift-subtract)."]

Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 115
fetched: OUT
pila: [1 "Se utiliza el algoritmo 'desplazar y restar' (shift-subtract)." 1]

Se utiliza el algoritmo 'desplazar y restar' (shift-subtract).
Ambiente del interprete:
```

```
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 116
fetched: NL
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 117
fetched: [PUSHFI "**************************************************************"]
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 118
fetched: [PUSHFI 1]
pila: [1 "**************************************************************"]

Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 119
fetched: OUT
pila: [1 "**************************************************************" 1]

**************************************************************
Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 120
fetched: NL
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 121
fetched: [PUSHFI "x: "]
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 122
fetched: [PUSHFI 1]
pila: [1 "x: "]

Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 123
fetched: OUT
pila: [1 "x: " 1]

x:
Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 124
fetched: FLUSH
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 125
fetched: [PUSHFI ""]
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 126
fetched: [POP 0]
pila: [1 ""]

Ambiente del interprete:
regs-de-act: [[[String ""] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String ""] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 127
fetched: [IN 0]
pila: [1]

23

Ambiente del interprete:
```

```
regs-de-act: [[[String "23"] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String "23"] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 128
fetched: [PUSHFM 0]
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String "23"] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String "23"] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 129
fetched: TOI
pila: [1 "23"]

Ambiente del interprete:
regs-de-act: [[[String "23"] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String "23"] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 130
fetched: [POP 1]
pila: [1 23]

Ambiente del interprete:
regs-de-act: [[[String "23"] [i64 23] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String "23"] [i64 23] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 131
fetched: [PUSHFI "y: "]
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String "23"] [i64 23] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String "23"] [i64 23] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 132
fetched: [PUSHFI 1]
pila: [1 "y: "]

Ambiente del interprete:
regs-de-act: [[[String "23"] [i64 23] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String "23"] [i64 23] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 133
fetched: OUT
pila: [1 "y: " 1]

y:
Ambiente del interprete:
regs-de-act: [[[String "23"] [i64 23] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String "23"] [i64 23] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 134
fetched: FLUSH
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String "23"] [i64 23] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String "23"] [i64 23] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 135
fetched: [PUSHFI ""]
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String "23"] [i64 23] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String "23"] [i64 23] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 136
fetched: [POP 0]
pila: [1 ""]

Ambiente del interprete:
regs-de-act: [[[String ""] [i64 23] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String ""] [i64 23] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 137
fetched: [IN 0]
pila: [1]

5

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String "5"] [i64 23] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 138
fetched: [PUSHFM 0]
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[String "5"] [i64 23] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 139
fetched: TOI
pila: [1 "5"]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 nil] [i64 nil] [i64 nil]]]
```

```
reg-actual: [[String "5"] [i64 23] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 140
fetched: [POP 2]
pila: [1 5]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 nil] [i64 nil]]]
reg-actual: [[String "5"] [i64 23] [i64 5] [i64 nil] [i64 nil]]
cont-prg: 141
fetched: [PUSHFI 0]
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 nil] [i64 nil]]]
reg-actual: [[String "5"] [i64 23] [i64 5] [i64 nil] [i64 nil]]
cont-prg: 142
fetched: [POP 3]
pila: [1 0]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 nil]]]
reg-actual: [[String "5"] [i64 23] [i64 5] [i64 0] [i64 nil]]
cont-prg: 143
fetched: [PUSHFI 0]
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 nil]]]
reg-actual: [[String "5"] [i64 23] [i64 5] [i64 0] [i64 nil]]
cont-prg: 144
fetched: [POP 4]
pila: [1 0]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]]]
reg-actual: [[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]]
cont-prg: 145
fetched: [PUSHFM 1]
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]]]
reg-actual: [[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]]
cont-prg: 146
fetched: [PUSHFM 2]
pila: [1 23]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]]]
reg-actual: [[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]]
cont-prg: 147
fetched: [PUSHADDR 3]
pila: [1 23 5]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]]]
reg-actual: [[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]]
cont-prg: 148
fetched: [PUSHADDR 4]
pila: [1 23 5 [0 3]]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]]]
reg-actual: [[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]]
cont-prg: 149
fetched: [CAL 2]
pila: [1 23 5 [0 3] [0 4]]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]] [[i64 nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]]
reg-actual: [[i64 nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil] [i64 nil]]
cont-prg: 2
fetched: [POPARG 3]
pila: [1 23 5 [0 3] [0 4] 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]] [[i64 nil] [i64 nil] [i64 nil] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 nil] [i64 nil] [i64 nil] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 3
fetched: [POPARG 2]
pila: [1 23 5 [0 3] 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]] [[i64 nil] [i64 nil] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 nil] [i64 nil] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 4
fetched: [POPARG 1]
pila: [1 23 5 150]
```

```
Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]] [[i64 nil] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 nil] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 5
fetched: [POPARG 0]
pila: [1 23 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 6
fetched: [PUSHFM 1]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 7
fetched: [PUSHFI 0]
pila: [1 150 5]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 8
fetched: EQ
pila: [1 150 5 0]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 9
fetched: [JC 11]
pila: [1 150 false]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 10
fetched: [JMP 17]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 17
fetched: [PUSHFI 0]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 18
fetched: [POPREF 2]
pila: [1 150 0]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 19
fetched: [PUSHFM 0]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 0]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 20
fetched: [POPREF 3]
pila: [1 150 23]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 21
fetched: [PUSHREF 3]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 22
fetched: [PUSHFI 0]
pila: [1 150 23]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
```

```
cont-prg: 23
fetched: LT
pila: [1 150 23 0]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 24
fetched: [JC 26]
pila: [1 150 false]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 25
fetched: [JMP 29]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 29
fetched: [PUSHFM 1]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 30
fetched: [PUSHFI 0]
pila: [1 150 5]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 31
fetched: GTE
pila: [1 150 5 0]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 32
fetched: [JC 34]
pila: [1 150 true]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 34
fetched: [PUSHFM 1]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 nil] [i64 nil]]
cont-prg: 35
fetched: [POP 4]
pila: [1 150 5]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 nil]]
cont-prg: 36
fetched: [PUSHFM 1]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 nil]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 nil]]
cont-prg: 37
fetched: [POP 5]
pila: [1 150 5]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 38
fetched: [JMP 45]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 45
fetched: [PUSHFM 5]
pila: [1 150]
```

```
Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 46
fetched: [PUSHREF 3]
pila: [1 150 5]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 47
fetched: LTE
pila: [1 150 5 23]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 48
fetched: [JC 50]
pila: [1 150 true]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 50
fetched: [PUSHFI 2]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 51
fetched: [POPMUL 5]
pila: [1 150 2]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 52
fetched: [JMP 45]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 45
fetched: [PUSHFM 5]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 46
fetched: [PUSHREF 3]
pila: [1 150 10]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 47
fetched: LTE
pila: [1 150 10 23]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 48
fetched: [JC 50]
pila: [1 150 true]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 50
fetched: [PUSHFI 2]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 51
fetched: [POPMUL 5]
pila: [1 150 2]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 52
```

```
fetched: [JMP 45]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 45
fetched: [PUSHFM 5]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 46
fetched: [PUSHREF 3]
pila: [1 150 20]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 47
fetched: LTE
pila: [1 150 20 23]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 48
fetched: [JC 50]
pila: [1 150 true]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 50
fetched: [PUSHFI 2]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 51
fetched: [POPMUL 5]
pila: [1 150 2]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]
cont-prg: 52
fetched: [JMP 45]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]
cont-prg: 45
fetched: [PUSHFM 5]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]
cont-prg: 46
fetched: [PUSHREF 3]
pila: [1 150 40]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]
cont-prg: 47
fetched: LTE
pila: [1 150 40 23]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]
cont-prg: 48
fetched: [JC 50]
pila: [1 150 false]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]
cont-prg: 49
fetched: [JMP 53]
pila: [1 150]

Ambiente del interprete:
```

```
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]
cont-prg: 53
fetched: [PUSHFM 5]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]
cont-prg: 54
fetched: [PUSHFM 4]
pila: [1 150 40]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]
cont-prg: 55
fetched: GT
pila: [1 150 40 5]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]
cont-prg: 56
fetched: [JC 58]
pila: [1 150 true]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]
cont-prg: 58
fetched: [PUSHFI 2]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]
cont-prg: 59
fetched: [POPMULREF 2]
pila: [1 150 2]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]
cont-prg: 60
fetched: [PUSHFI 2]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 40]]
cont-prg: 61
fetched: [POPDIV 5]
pila: [1 150 2]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 62
fetched: [PUSHFM 5]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] NL [i64 5] [i64 20]]
cont-prg: 63
fetched: [PUSHREF 3]
pila: [1 150 20]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 64
fetched: LTE
pila: [1 150 20 23]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 65
fetched: [JC 67]
pila: [1 150 true]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 67
fetched: [PUSHFM 5]
```

```
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 23]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 68
fetched: [POPSUBREF 3]
pila: [1 150 20]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 69
fetched: [PUSHFI 1]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 0] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 70
fetched: [POPADDREF 2]
pila: [1 150 1]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 1] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 71
fetched: [JMP 53]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 1] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 53
fetched: [PUSHFM 5]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 1] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 54
fetched: [PUSHFM 4]
pila: [1 150 20]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 1] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 55
fetched: GT
pila: [1 150 20 5]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 1] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 56
fetched: [JC 58]
pila: [1 150 true]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 1] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 58
fetched: [PUSHFI 2]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 1] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 59
fetched: [POPMULREF 2]
pila: [1 150 2]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 2] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 60
fetched: [PUSHFI 2]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 2] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 20]]
cont-prg: 61
fetched: [POPDIV 5]
pila: [1 150 2]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 2] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
```

```
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 62
fetched: [PUSHFM 5]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 2] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 63
fetched: [PUSHREF 3]
pila: [1 150 10]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 2] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 64
fetched: LTE
pila: [1 150 10 3]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 2] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 65
fetched: [JC 67]
pila: [1 150 false]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 2] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 66
fetched: [JMP 71]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 2] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 71
fetched: [JMP 53]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 2] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 53
fetched: [PUSHFM 5]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 2] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 54
fetched: [PUSHFM 4]
pila: [1 150 10]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 2] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 55
fetched: GT
pila: [1 150 10 5]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 2] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 56
fetched: [JC 58]
pila: [1 150 true]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 2] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 58
fetched: [PUSHFI 2]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 2] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 59
fetched: [POPMULREF 2]
pila: [1 150 2]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 60
fetched: [PUSHFI 2]
pila: [1 150]
```

```
Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 10]]
cont-prg: 61
fetched: [POPDIV 5]
pila: [1 150 2]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 62
fetched: [PUSHFM 5]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 63
fetched: [PUSHREF 3]
pila: [1 150 5]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 64
fetched: LTE
pila: [1 150 5 3]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 65
fetched: [JC 67]
pila: [1 150 false]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 66
fetched: [JMP 71]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 71
fetched: [JMP 53]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 53
fetched: [PUSHFM 5]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 54
fetched: [PUSHFM 4]
pila: [1 150 5]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 55
fetched: GT
pila: [1 150 5 5]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 56
fetched: [JC 58]
pila: [1 150 false]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 57
fetched: [JMP 72]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
```

```
cont-prg: 72
fetched: [PUSHFM 0]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 73
fetched: [PUSHFI 0]
pila: [1 150 23]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 74
fetched: LT
pila: [1 150 23 0]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 75
fetched: [JC 77]
pila: [1 150 false]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 76
fetched: [JMP 83]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 83
fetched: [PUSHFM 1]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 84
fetched: [PUSHFI 0]
pila: [1 150 5]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 85
fetched: LT
pila: [1 150 5 0]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 86
fetched: [JC 88]
pila: [1 150 false]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 87
fetched: [JMP 91]
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]]
reg-actual: [[i64 23] [i64 5] [i64 [0 3]] [i64 [0 4]] [i64 5] [i64 5]]
cont-prg: 91
fetched: RETN
pila: [1 150]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]]]
reg-actual: [[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]]
cont-prg: 150
fetched: [PUSHFM 3]
pila: [1]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]]]
reg-actual: [[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]]
cont-prg: 151
fetched: [PUSHFM 4]
pila: [1 4]
```

```
Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]]]
reg-actual: [[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]]
cont-prg: 152
fetched: [CAL 92]
pila: [1 4 3]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 nil] [i64 nil]]]
reg-actual: [[i64 nil] [i64 nil]]
cont-prg: 92
fetched: [POPARG 1]
pila: [1 4 3 153]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 nil] [i64 3]]]
reg-actual: [[i64 nil] [i64 3]]
cont-prg: 93
fetched: [POPARG 0]
pila: [1 4 153]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 4] [i64 3]]]
reg-actual: [[i64 4] [i64 3]]
cont-prg: 94
fetched: [PUSHFI "Cociente: {}"]
pila: [1 153]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 4] [i64 3]]]
reg-actual: [[i64 4] [i64 3]]
cont-prg: 95
fetched: [PUSHFM 0]
pila: [1 153 "Cociente: {}"]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 4] [i64 3]]]
reg-actual: [[i64 4] [i64 3]]
cont-prg: 96
fetched: [PUSHFI 2]
pila: [1 153 "Cociente: {}" 4]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 4] [i64 3]]]
reg-actual: [[i64 4] [i64 3]]
cont-prg: 97
fetched: OUT
pila: [1 153 "Cociente: {}" 4 2]

Cociente: 4
Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 4] [i64 3]]]
reg-actual: [[i64 4] [i64 3]]
cont-prg: 98
fetched: NL
pila: [1 153]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 4] [i64 3]]]
reg-actual: [[i64 4] [i64 3]]
cont-prg: 99
fetched: [PUSHFI "Resto: {}"]
pila: [1 153]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 4] [i64 3]]]
reg-actual: [[i64 4] [i64 3]]
cont-prg: 100
fetched: [PUSHFM 1]
pila: [1 153 "Resto: {}"]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 4] [i64 3]]]
reg-actual: [[i64 4] [i64 3]]
cont-prg: 101
fetched: [PUSHFI 2]
pila: [1 153 "Resto: {}" 3]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 4] [i64 3]]]
reg-actual: [[i64 4] [i64 3]]
cont-prg: 102
fetched: OUT
pila: [1 153 "Resto: {}" 3 2]

Resto: 3
Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 4] [i64 3]]]
```

```
reg-actual: [[i64 4] [i64 3]]
cont-prg: 103
fetched: NL
pila: [1 153]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]] [[i64 4] [i64 3]]]
reg-actual: [[i64 4] [i64 3]]
cont-prg: 104
fetched: RETN
pila: [1 153]

Ambiente del interprete:
regs-de-act: [[[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]]]
reg-actual: [[String "5"] [i64 23] [i64 5] [i64 4] [i64 3]]
cont-prg: 153
fetched: RETN
pila: [1]

Ambiente del interprete:
regs-de-act: []
reg-actual: nil
cont-prg: 1
fetched: HLT
pila: []

Rust>
```

Enlaces de interés

Diagramas de sintaxis de la versión de Rust implementada en el TP
http://diegocorsi.com.ar/rust

Herramienta **Railroad Diagram Generator**
https://www.bottlecaps.de/rr/ui

Página oficial del lenguaje **Rust**
https://www.rust-lang.org/es

El libro **The Rust Programming Language**
https://doc.rust-lang.org/book

Colección de ejemplos **Rust by Example**
https://doc.rust-lang.org/stable/rust-by-example

Página para probar Rust en línea
https://play.rust-lang.org