

# Procedural and Object-oriented Programming 6

## Trace (Debug)

### Business Programming 1

### Business Programming 2



Basics,  
Parsing

Commands,  
APIs

Window-  
Automatisation,  
Web-Scripting

Security,  
Debugging

Graphical User  
Interfaces (GUI),  
Sockets,  
...

# TRACE – How Does Your Code Execute?



- Understanding what your Rexx program does
  - Sometimes coding errors cannot be understood easily
  - Keyword statement **TRACE** and built-in function (BIF) **TRACE()**
    - Allows to temporarily have Rexx show in detail how it executes your statements
    - Allows to enter interactive mode that allows to inspect a specific location in your code
  - Most important options
    - **N** normal trace setting, only traces failures in commands
    - **A** show all statements that get executed
    - **R** show all statements and their final results
    - **I** show in detail all intermediate steps when executing a statement
  - Prefixing an option with a question mark (?) causes **TRACE** to become interactive and to trace statements step by step at each press of the `<enter>` key
    - Enter "`trace n`" to return to normal execution and stop interactive trace



# TRACE N – Example, 1



- Simple program, run twice with different output, default option **N**
  - Values change per run because of using the REXX `random()`-BIF

```
/* default: "trace normal" in effect */  
a=100+20  
b=100+random(10,99)  
say "a*b:" a*b
```

Output of run # 1:

```
a*b: 16800
```

Output of run # 2:

```
a*b: 17400
```



# TRACE A – Example, 2



- Running the program with **TRACE A**
  - Option **A** traces all statements (clauses) before execution

```
trace a      /* show all statements */  
a=100+20  
b=100+random(10,99)  
say "a*b:" a*b
```

Output:

```
2 *-* a=100+20  
3 *-* b=100+random(10,99)  
4 *-* say "a*b:" a*b  
a*b: 17040
```



# TRACE R – Example, 3



- Running the program with **TRACE R**
  - Option **R** traces all statements (clauses) and their final results

```
trace r      /* show results */  
a=100+20  
b=100+random(10,99)  
say "a*b:" a*b
```

Output:

```
2 *-* a=100+20  
  >>> "120"  
3 *-* b=100+random(10,99)  
  >>> "113"  
4 *-* say "a*b:" a*b  
  >>> "a*b: 13560"  
a*b: 13560
```



# TRACE I – Example, 4

- Running the program with **TRACE I**
  - Option **I** traces all intermediate results

```
trace i      /* show intermediates */
a=100+20
b=100+random(10,99)
say "a*b:" a*b
```

→  
Output:

```
2 *-* a=100+20
>L> "100"
>L> "20"
>O> "+" => "120"
>>> "120"
>=> A <= "120"
3 *-* b=100+random(10,99)
>L> "100"
>L> "10"
>A> "10"
>L> "99"
>A> "99"
>F> RANDOM => "98"
>O> "+" => "198"
>>> "198"
>=> B <= "198"
4 *-* say "a*b:" a*b
>L> "a*b:"
>V> A => "120"
>V> B => "198"
>O> "*" => "23760"
>O> " " => "a*b: 23760"
>>> "a*b: 23760"
a*b: 23760
```

# TRACE I, TRACE N – Example, 5



- Running the program with **TRACE I** for tracing a single statement
  - **TRACE N** gets used to reset tracing to the default

```
a=100+20
trace i      /* show intermediates */
b=100+random(10,99)
trace n      /* reset to normal */
say "a*b:" a*b
```

Output:

```
3 *** b=100+random(10,99)
  >L>  "100"
  >L>  "10"
  >A>  "10"
  >L>  "99"
  >A>  "99"
  >F>  RANDOM => "80"
  >O>  "+" => "180"
  >>>  "180"
  >=>  B <= "180"
4 *** trace n      /* reset to normal */
a*b: 21600
```



# TRACE ?A – Example, 6



- Using **TRACE ?A** for tracing all statements in interactive mode
  - Interactive tracing allows for inspecting, changing values, tracing step by step

```
trace ?a      /* show all statements, enter interactive mode */
a=100+20
b=100+random(10,99)
say "a*b:" a*b
```

Output:

```
+++ "WindowsNT COMMAND E:\tmp\06_ooRexx\19_trace_r_interactive.rex"
2 ** a=100+20
+++ Interactive trace. "Trace Off" to end debug, ENTER to continue. +++
say a
120
a=100; trace r
3 **      b=100+random(10,99)
      >>>      "198"
trace off
a*b: 19800
```

Execution stops, manually adding a statement to display the current value of variable **a**, pressing <ENTER> continues.

Execution stops, manually adding two statements: changing variable **a** to 100 and changing tracing to trace results from now on, pressing <ENTER> continues.

Execution stops, manually adding a single statement: set interactive tracing off, pressing <ENTER> continues.