Solingen, November 7th 2024

News from Dyalog

Stine Kromberg, CEO
Morten Kromberg, CTO

A New Face at the Helm

- Stine Kromberg
- MSc in Business Administration and Information Systems
- Grew up with APL – but not an APL'er
- Officially 4.5 years with Dyalog
- 10 months as CEO

Future of Dyalog

- [Many] New faces
- Same direction, same values!
- First support current users
- Build the tools and support for new users
- Help train the next generation of users

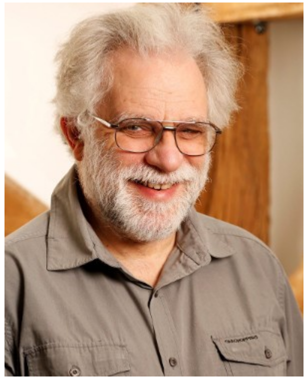
The Challenge and the Forge – for new and current developers

Road Map - 2023

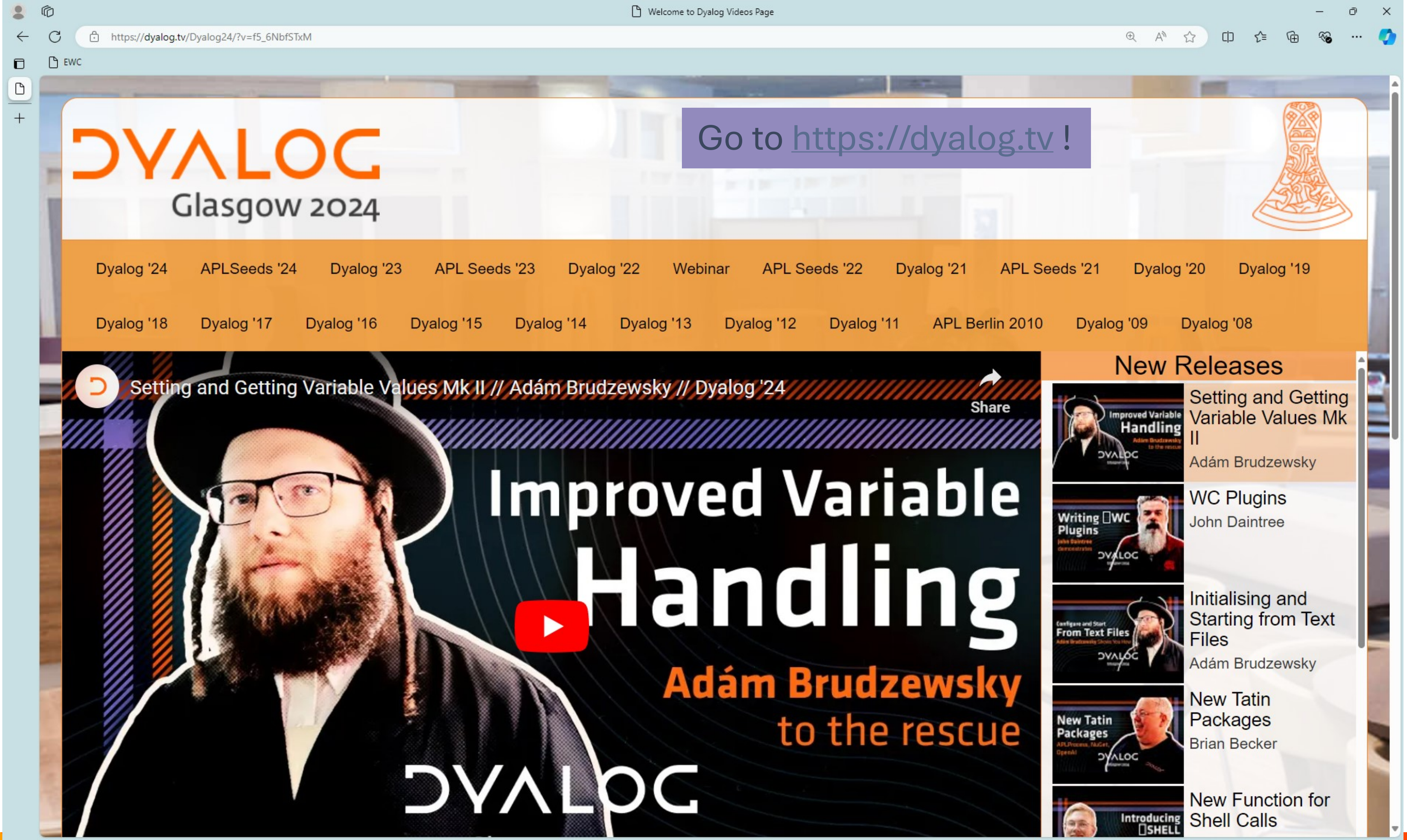
Road Map - 2024

Retired

- Geoff Streeter
- Gitte Christensen
- Peter Donnelly



U17 (29/11): Let's Put the Future Behind Us (Panel Discussion)



Welcome to Dyalog Videos Page

https://dyalog.tv/Dyalog24/?v=f5_6NbfSTxM

DYALOG Glasgow 2024

Go to https://dyalog.tv !

Dyalog '24 | APLSeeds '24 | Dyalog '23 | APL Seeds '23 | Dyalog '22 | Webinar | APL Seeds '22 | Dyalog '21 | APL Seeds '21 | Dyalog '20 | Dyalog '19

Dyalog '18 | Dyalog '17 | Dyalog '16 | Dyalog '15 | Dyalog '14 | Dyalog '13 | Dyalog '12 | Dyalog '11 | APL Berlin 2010 | Dyalog '09 | Dyalog '08

Setting and Getting Variable Values Mk II // Adám Brudzewsky // Dyalog '24

New Releases

Setting and Getting Variable Values Mk II
Adám Brudzewsky

WC Plugins
John Daintree

Initialising and Starting from Text Files
Adám Brudzewsky

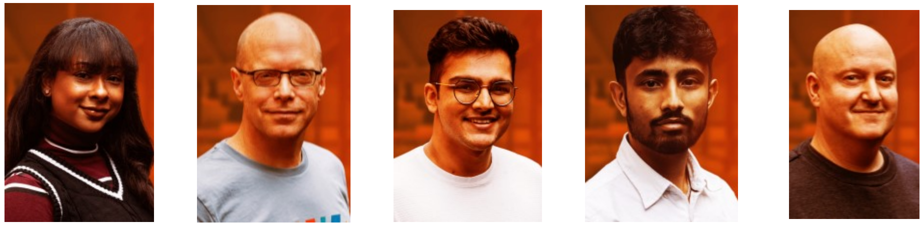
New Tatin Packages
Brian Becker

New Function for Shell Calls

Dyalog – The Next Generation

2010-2021

2022



2023

U18 (29/11): The New Breed Plugs In (Panel Discussion)

Dyalog – The Next Generation

2024 Summer Interns

U03 next week: raylib-apl (Brian Ellingsgaard)

U07 tomorrow: Climbing Trees and Catching Bugs (Asher Harvey-Smith)

Karl Holt

(February, Århus, DK)

- Karl is a member of the APL Tools Group and an APL Consultant
- Karl is working on the emulator for APL+Win GUI (⎕WI)

Tomorrow, Here: Migrating APL+Win GUI

Brandon Wilson

(July, 昭和村 / Shōwa, Japan)

- Brandon is heading up our Static Analysis project
- Working with Aaron Hsu to enhance the co-dfns compiler to support the project

D06 (22/11): Static Analysis of APL in APL

D14 (22/11): Data Parallel Proof Verification in APL

Martina Crippa

(August, Copenhagen)

- Martina is a member of the APL Tools group and a Consultant
- Martina also works in C++
- Her first projects are a Kafka interface (C++) and the Dyalog File Server (APL)

Kafka is an Event Streaming Platform

(message queue)

Neil Kirsopp
(Gunzenhausen, Bavaria)

Tomorrow, Here: Migrating GUI to the Cloud

- Neil is a JavaScript developer (amongst many things)
- Also an APL enthusiast
- He will work on enhancing EWC, and also take over RIDE development
- … and look at a VS Code / Emacs plugin

Big Things Heading Your Way

- Array Notation
- Set & Get Variables
- Token-by-Token Debugging
- Everywhere WC
- Reverse Compose

Array Notation

```
z←,⊂'Three'
z,←⊂'Blind'
z,←⊂'Mice'
```

≡

```
('Three'
 'Blind'
 'Mice')
```

```
z←⍉⍪0 6 1 8
z⍪← 1 4 1 4
z⍪← 2 7 1 8
z⍪← 3 1 4 2
```

≡

```
[0 6 1 8
 1 4 1 4
 2 7 1 8
 3 1 4 2]
```

```
z←⍪10
z⍪←20
z⍪←30
z⍪←40
```

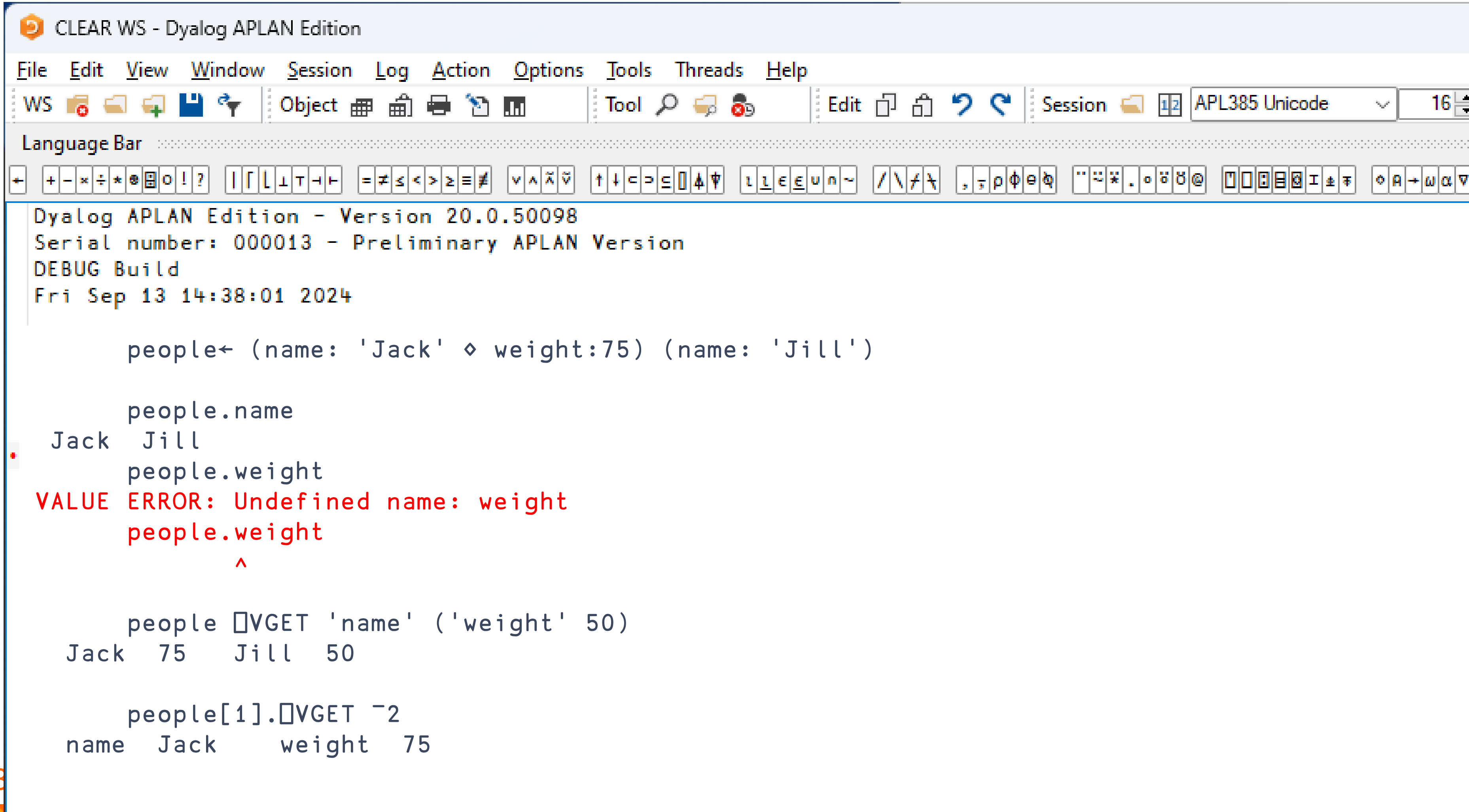
≡

```
[10
 20
 30
 40]
```

Namespace Notation

```
      person←(first: 'Max' ⋄ last: 'Mustermann')

      person.last,', ',person.first
Mustermann, Max

      person.(first,' ',last)
Max Mustermann
```

D03: Array Notation: A Journey of Discovery (John Daintree)


CLEAR WS - Dyalog APLAN Edition
File Edit View Window Session Log Action Options Tools Threads Help
WS Object Tool Edit Session APL385 Unicode 16
Language Bar

```
Dyalog APLAN Edition - Version 20.0.50098
Serial number: 000013 - Preliminary APLAN Version
DEBUG Build
Fri Sep 13 14:38:01 2024

      people← (name: 'Jack' ⋄ weight:75) (name: 'Jill')

      people.name
 Jack  Jill
      people.weight
VALUE ERROR: Undefined name: weight
      people.weight
            ∧

      people ⎕VGET 'name' ('weight' 50)
  Jack  75   Jill  50

      people[1].⎕VGET ¯2
  name  Jack    weight  75
```

Set and Get Variables

Get:

```
    [source] ⎕VGET 'Name' ('Height' ¯1)  ⍝ Values w/optional defaults
```

D04: Setting and Getting Variables (Adam Brudzewsky)

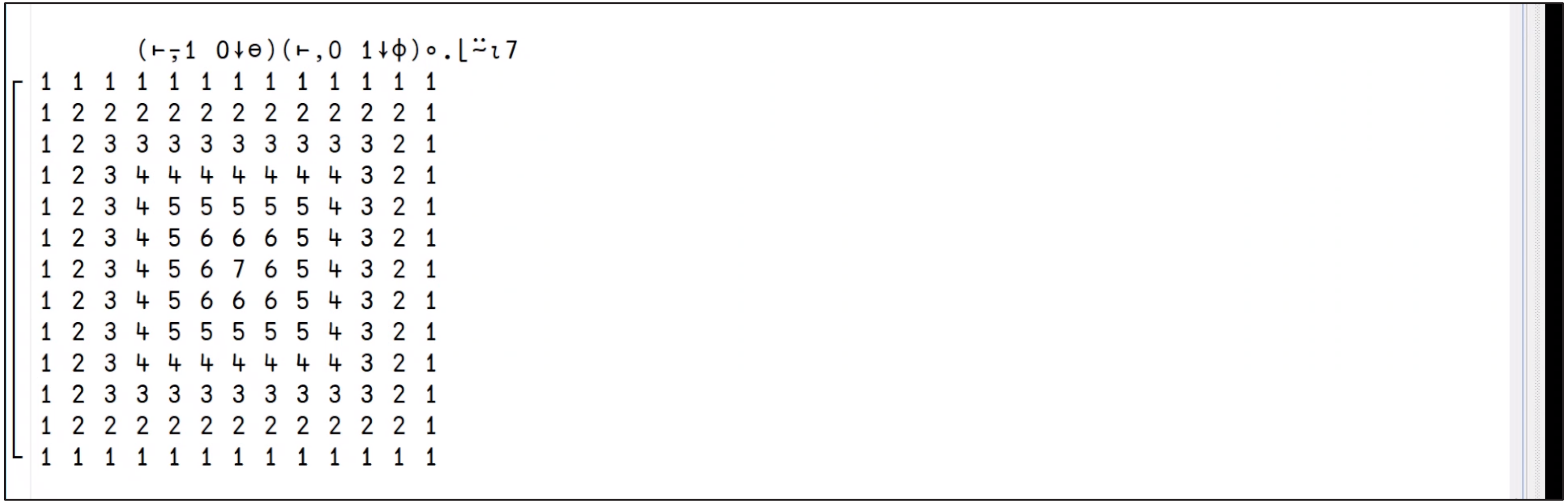
Name List with Values:

```
     (names values)←[source] ⎕VGET 2       ⍝ Name Matrix and values
              pairs←[source] ⎕VGET ¯2      ⍝ (Name Value) Pairs
```

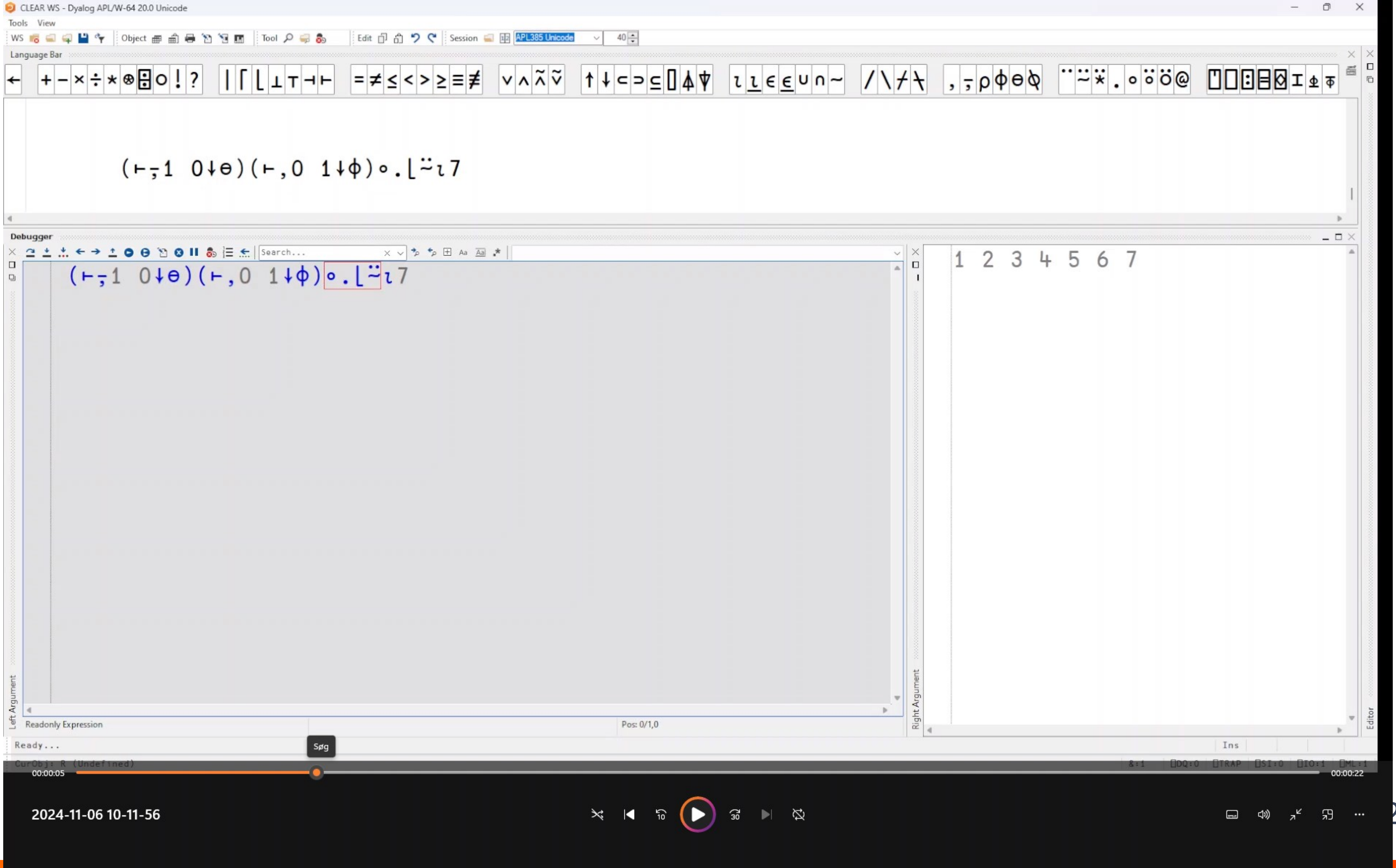
Set:

```
     ref←[target] ⎕VSET ('First' 'Lieschen')('Last' 'Müller')
     ref←[target] ⎕VSET (2 5⍴'FirstLast ')('Max' 'Munstermann')
```

Token by Token Debugging



```
      (⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
┌ 1 1 1 1 1 1 1 1 1 1 1 1 1
│ 1 2 2 2 2 2 2 2 2 2 2 2 1
│ 1 2 3 3 3 3 3 3 3 3 3 2 1
│ 1 2 3 4 4 4 4 4 4 4 3 2 1
│ 1 2 3 4 5 5 5 5 5 4 3 2 1
│ 1 2 3 4 5 6 6 6 5 4 3 2 1
│ 1 2 3 4 5 6 7 6 5 4 3 2 1
│ 1 2 3 4 5 6 6 6 5 4 3 2 1
│ 1 2 3 4 5 5 5 5 5 4 3 2 1
│ 1 2 3 4 4 4 4 4 4 4 3 2 1
│ 1 2 3 3 3 3 3 3 3 3 3 2 1
│ 1 2 2 2 2 2 2 2 2 2 2 2 1
└ 1 1 1 1 1 1 1 1 1 1 1 1 1
```

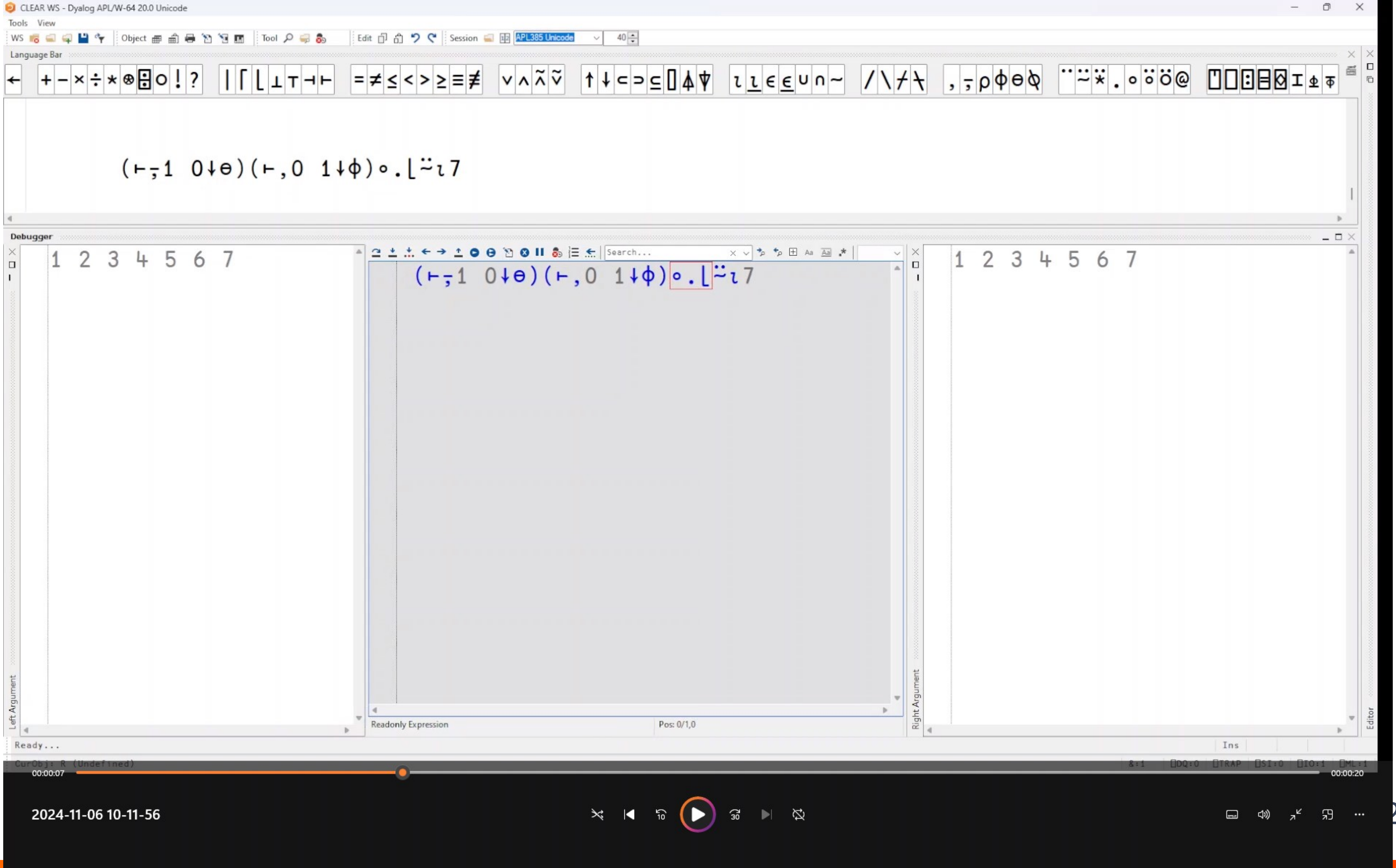


```
(⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
```

```
1 2 3 4 5 6 7
```

```
Readonly Expression                    Pos: 0/1,0
Ready...                                                Ins
CurObj: R (Undefined)            &:1  ⎕DQ:0  ⎕TRAP  ⎕SI:0  ⎕IO:1  ⎕ML:1
```

24



```
(⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
```

Left Argument:

```
1 2 3 4 5 6 7
```

Expression:

```
(⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
```

Right Argument:

```
1 2 3 4 5 6 7
```

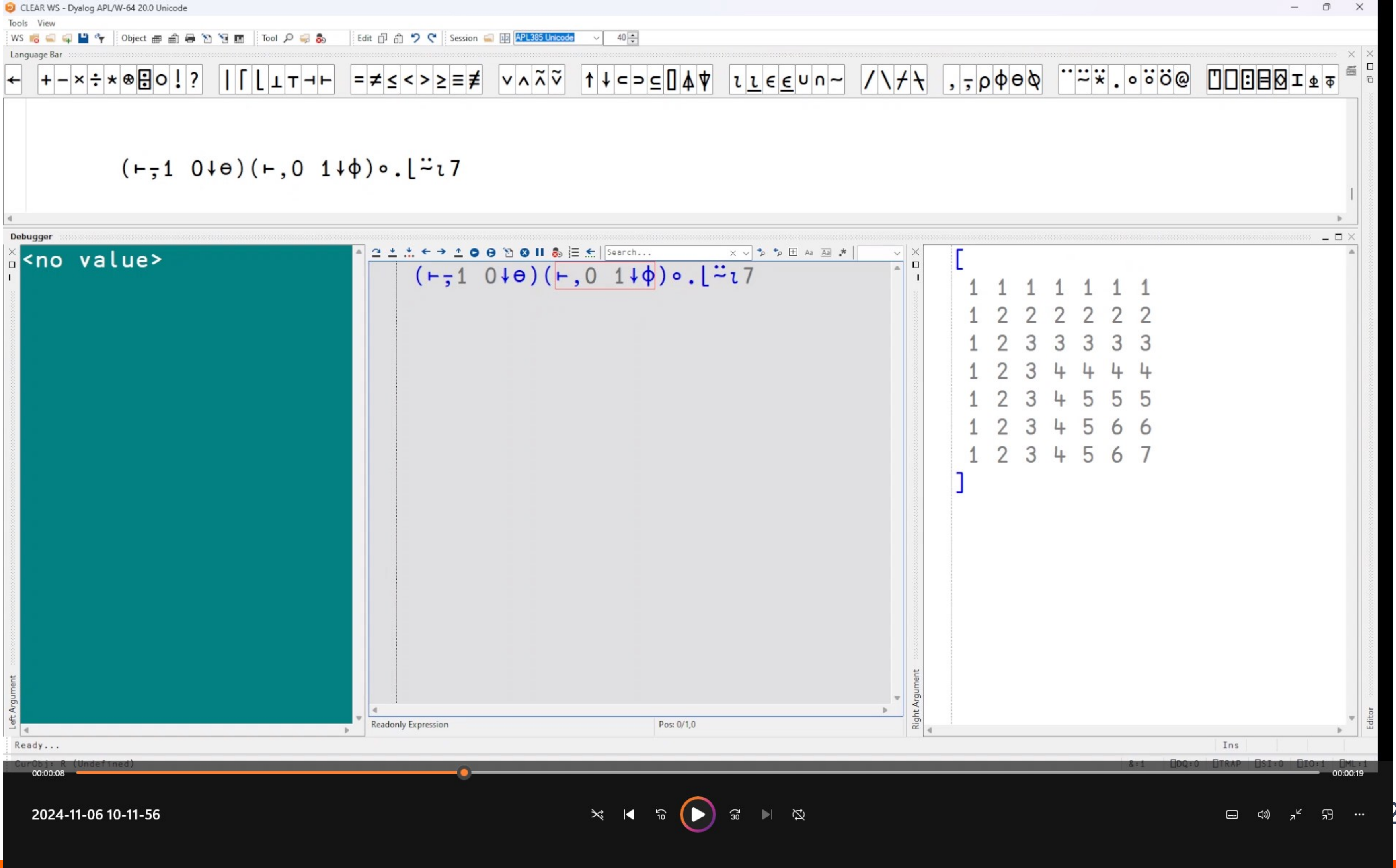
Readonly Expression Pos: 0/1,0

Ready...

CurObj: R (Undefined)

00:00:07 00:00:20

2024-11-06 10-11-56



```
      (⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
```

```
<no value>
```

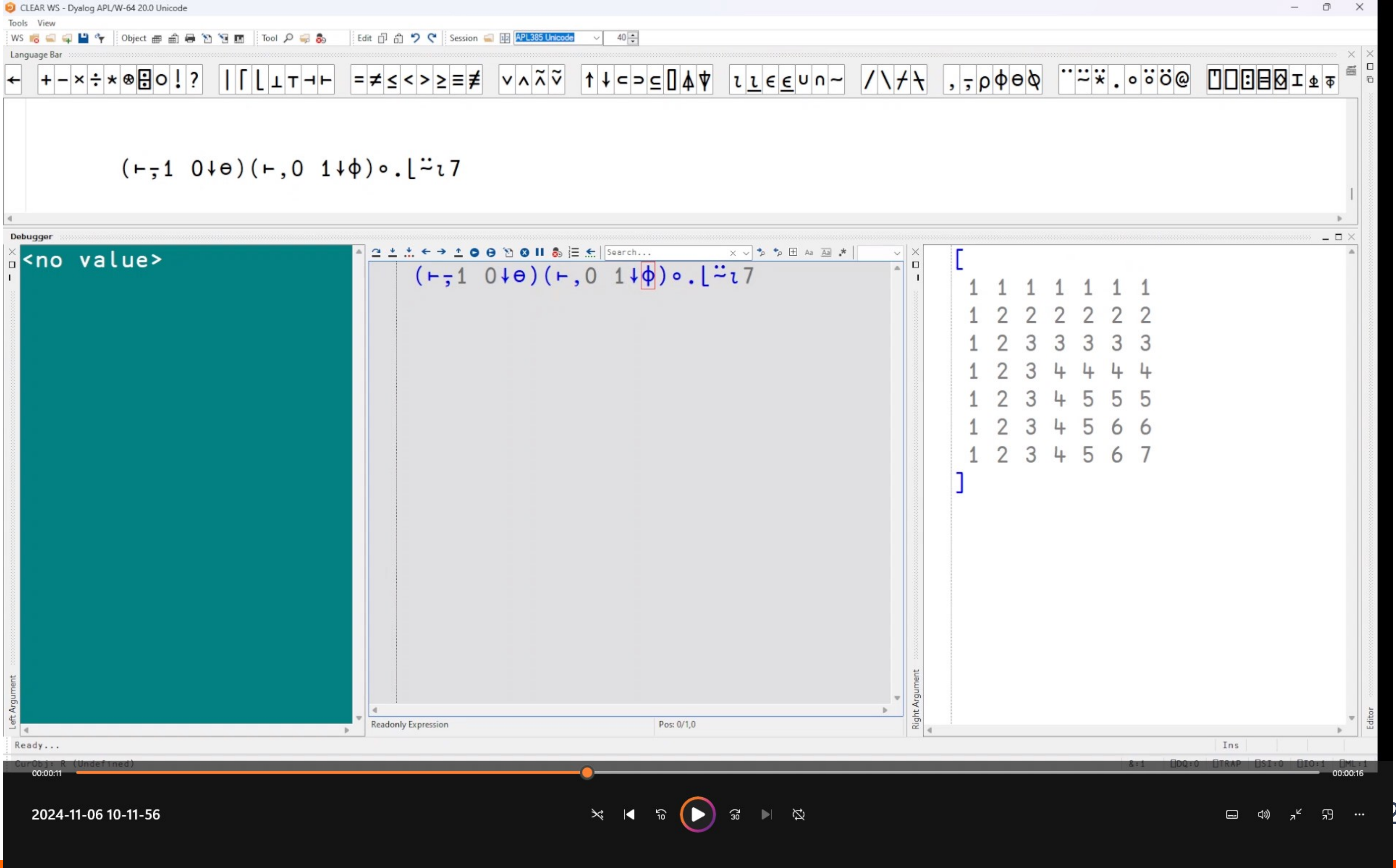
```
(⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
```

```
[
 1 1 1 1 1 1 1
 1 2 2 2 2 2 2
 1 2 3 3 3 3 3
 1 2 3 4 4 4 4
 1 2 3 4 5 5 5
 1 2 3 4 5 6 6
 1 2 3 4 5 6 7
]
```

Readonly Expression   Pos: 0/1,0

Ready...   Ins

CurObj: R (Undefined)   &:1 ⎕DQ:0 ⎕TRAP ⎕SI:0 ⎕IO:1 ⎕ML:1


CLEAR WS - Dyalog APL/W-64 20.0 Unicode
Tools View
WS Object Tool Edit Session APL385 Unicode 40
Language Bar
Debugger
Search...
Left Argument
Right Argument
Editor

```
(⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
```

```
<no value>
```

```
(⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
```

```
[
 1 1 1 1 1 1 1
 1 2 2 2 2 2 2
 1 2 3 3 3 3 3
 1 2 3 4 4 4 4
 1 2 3 4 5 5 5
 1 2 3 4 5 6 6
 1 2 3 4 5 6 7
]
```

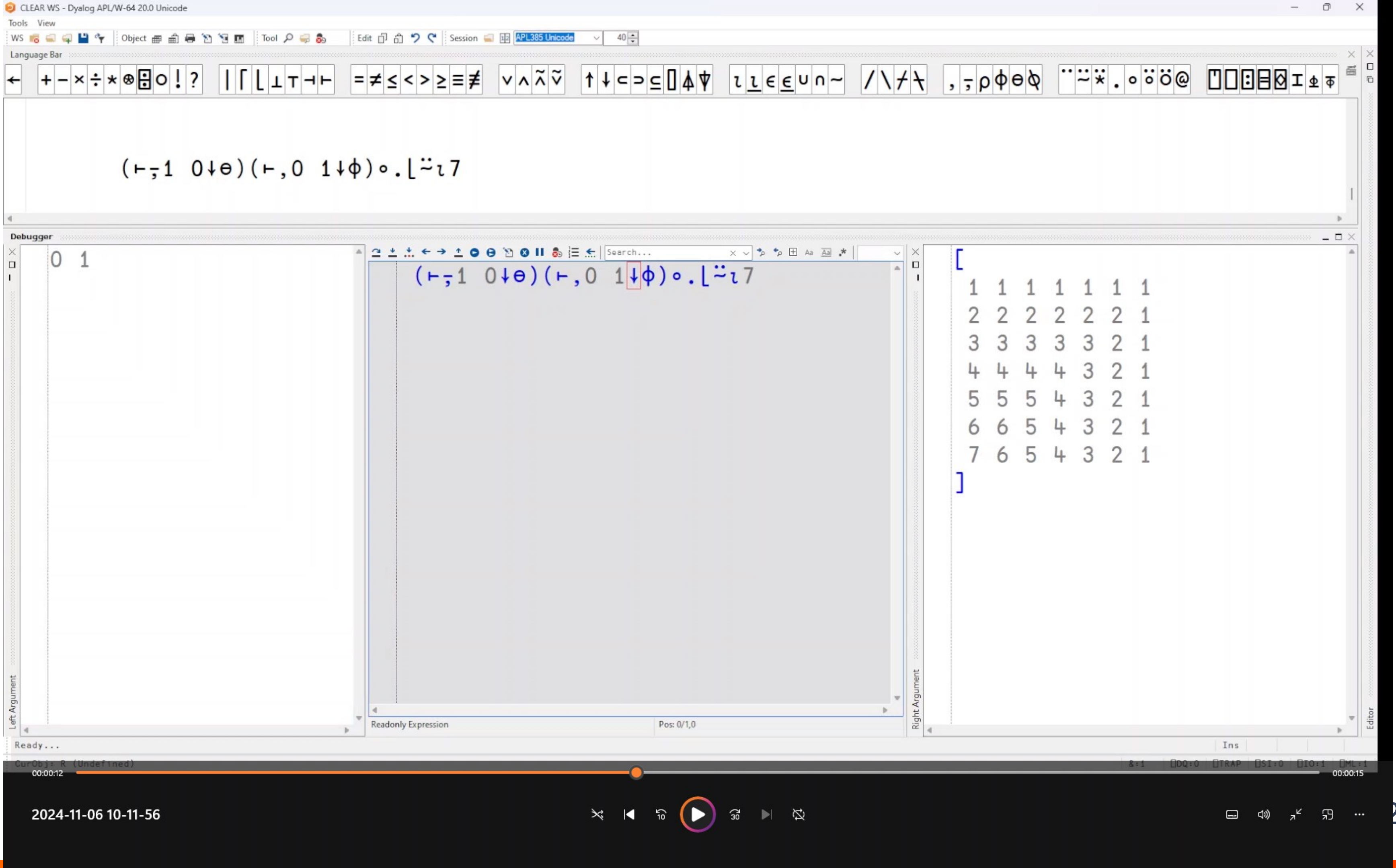
Readonly Expression   Pos: 0/1,0

Ready...   Ins

CurObj: R (Undefined)   &:1 ⎕DQ:0 ⎕TRAP ⎕SI:0 ⎕IO:1 ⎕ML:1

00:00:11   00:00:16

2024-11-06 10-11-56



```
(⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
```

Left argument:

```
0 1
```

Expression:

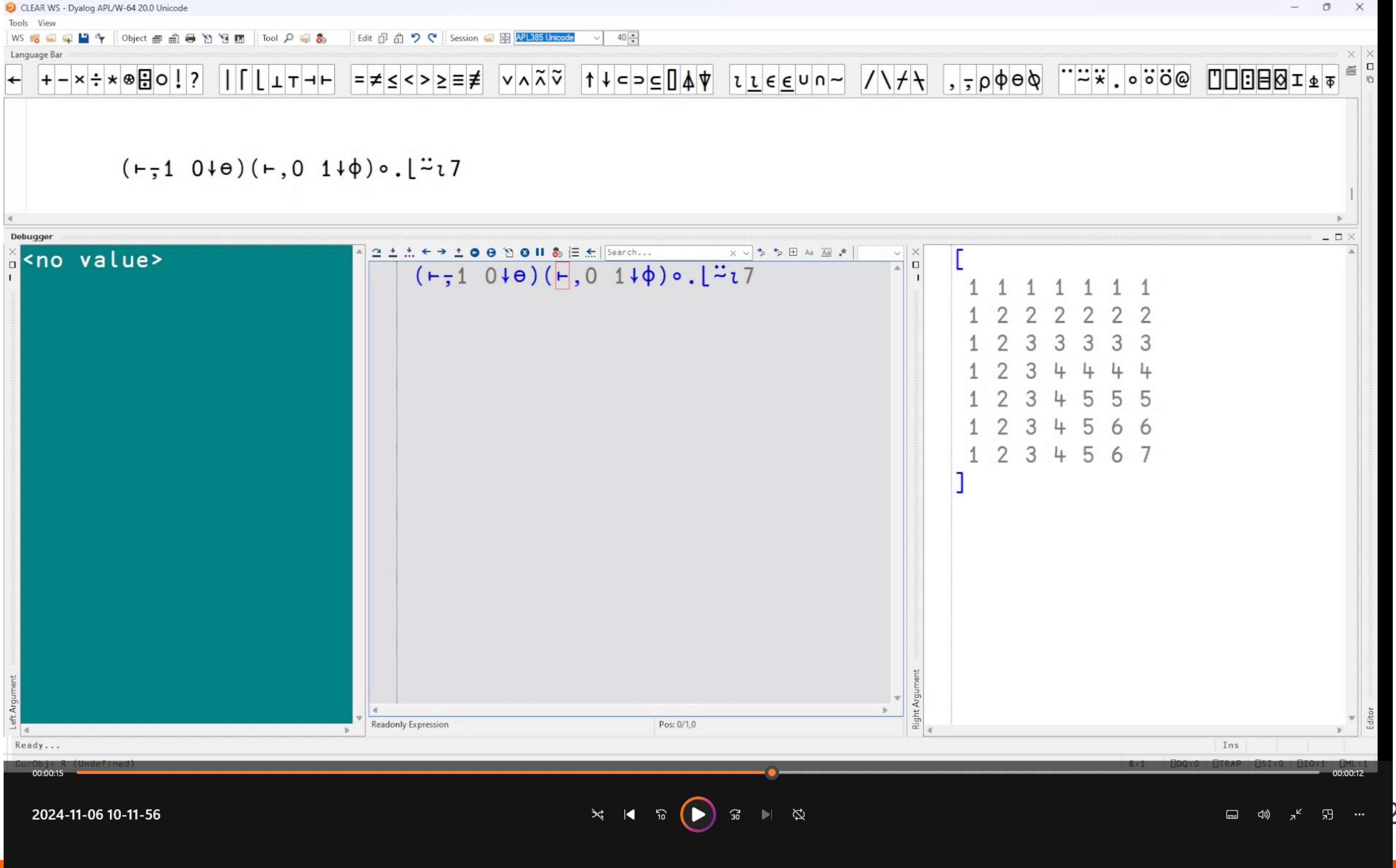
```
(⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
```

Readonly Expression   Pos: 0/1,0

Right argument:

```
[
 1 1 1 1 1 1 1
 2 2 2 2 2 2 1
 3 3 3 3 3 2 1
 4 4 4 4 3 2 1
 5 5 5 4 3 2 1
 6 6 5 4 3 2 1
 7 6 5 4 3 2 1
]
```

Ready...



```
(⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
```

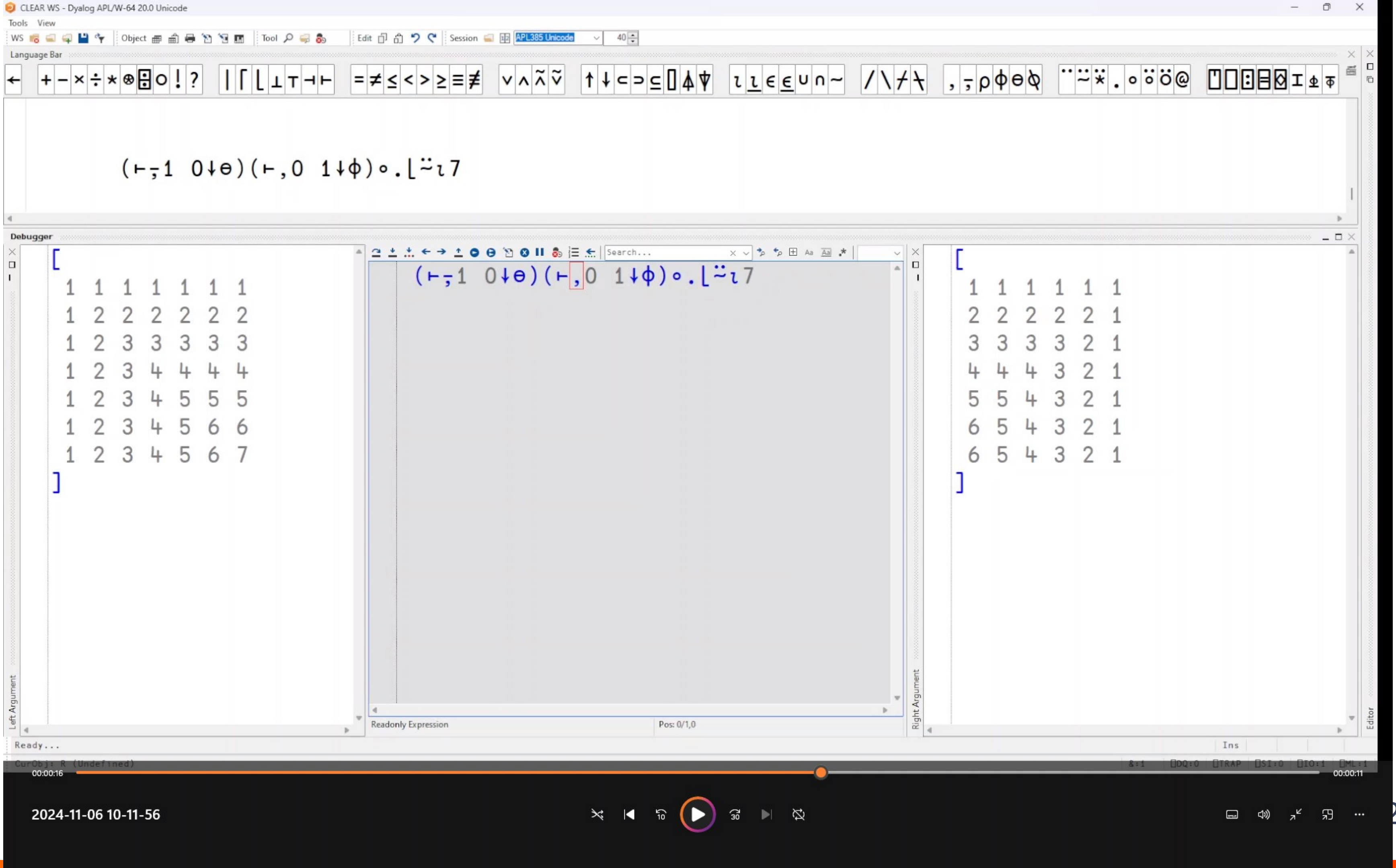
```
<no value>
```

```
(⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
```

```
[
 1 1 1 1 1 1 1
 1 2 2 2 2 2 2
 1 2 3 3 3 3 3
 1 2 3 4 4 4 4
 1 2 3 4 5 5 5
 1 2 3 4 5 6 6
 1 2 3 4 5 6 7
]
```

```
Readonly Expression          Pos: 0/1,0
Ready...                     Ins
CurObj: R (Undefined)        &:1  ⎕DQ:0  ⎕TRAP  ⎕SI:0  ⎕IO:1  ⎕ML:1
```

24



```
(⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
```

Left Argument:

```
[
 1 1 1 1 1 1 1
 1 2 2 2 2 2 2
 1 2 3 3 3 3 3
 1 2 3 4 4 4 4
 1 2 3 4 5 5 5
 1 2 3 4 5 6 6
 1 2 3 4 5 6 7
]
```

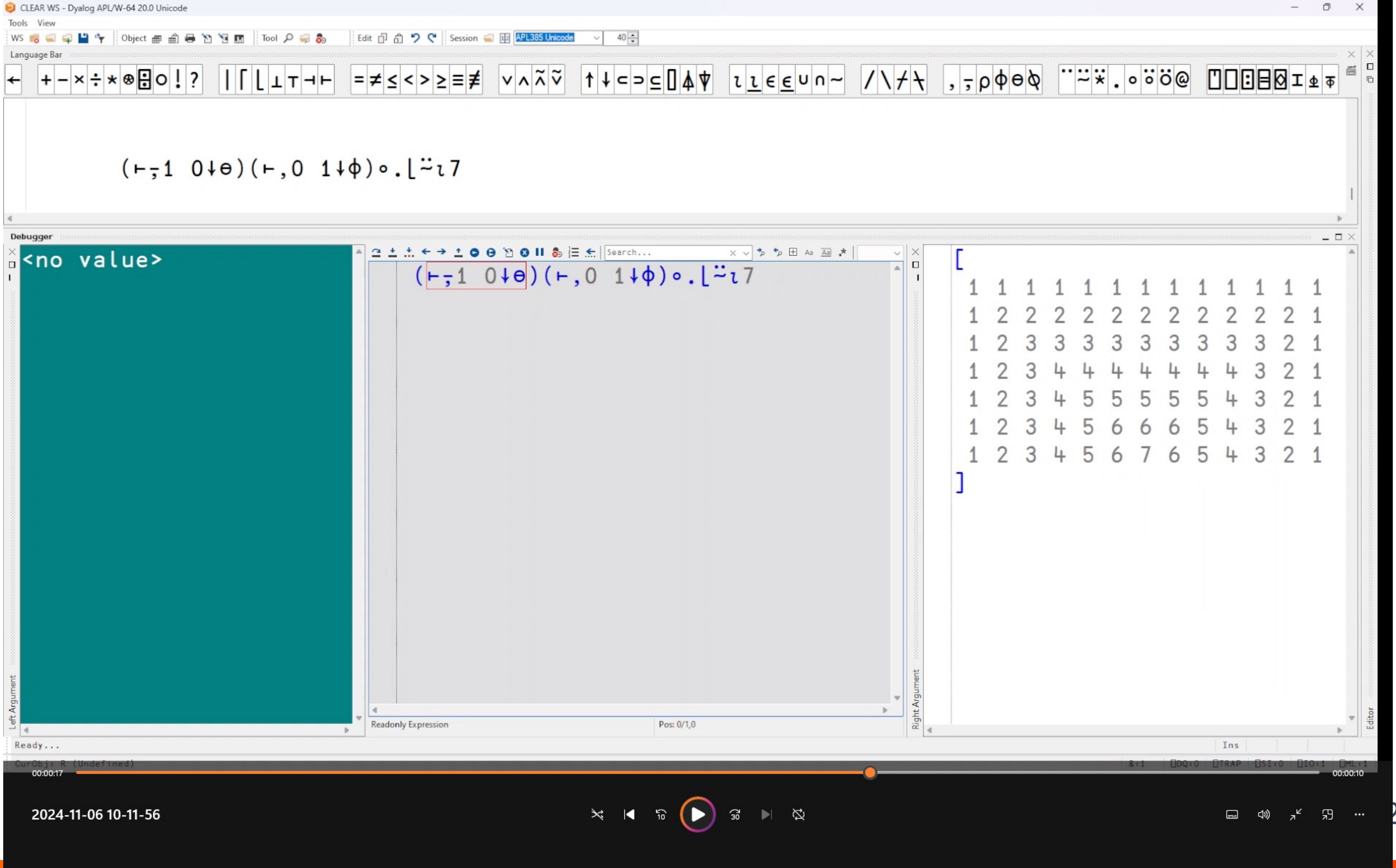
```
(⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
```

```
Readonly Expression                Pos: 0/1,0
```

Right Argument:

```
[
 1 1 1 1 1 1
 2 2 2 2 2 1
 3 3 3 3 2 1
 4 4 4 3 2 1
 5 5 4 3 2 1
 6 5 4 3 2 1
 6 5 4 3 2 1
]
```

```
Ready...                                  Ins
CurObj: R (Undefined)        &:1  ⎕DQ:0  ⎕TRAP  ⎕SI:0  ⎕IO:1  ⎕ML:1
```


CLEAR WS - Dyalog APL/W-64 20.0 Unicode
Tools View
WS Object Tool Edit Session APL385 Unicode 40
Language Bar
Debugger
Left Argument
Right Argument
Editor
Search...

```
(⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
```

```
<no value>
```

```
(⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
```

```
[
 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 2 2 2 2 2 2 2 2 2 2 2 1
 1 2 3 3 3 3 3 3 3 3 3 2 1
 1 2 3 4 4 4 4 4 4 4 3 2 1
 1 2 3 4 5 5 5 5 5 4 3 2 1
 1 2 3 4 5 6 6 6 5 4 3 2 1
 1 2 3 4 5 6 7 6 5 4 3 2 1
]
```

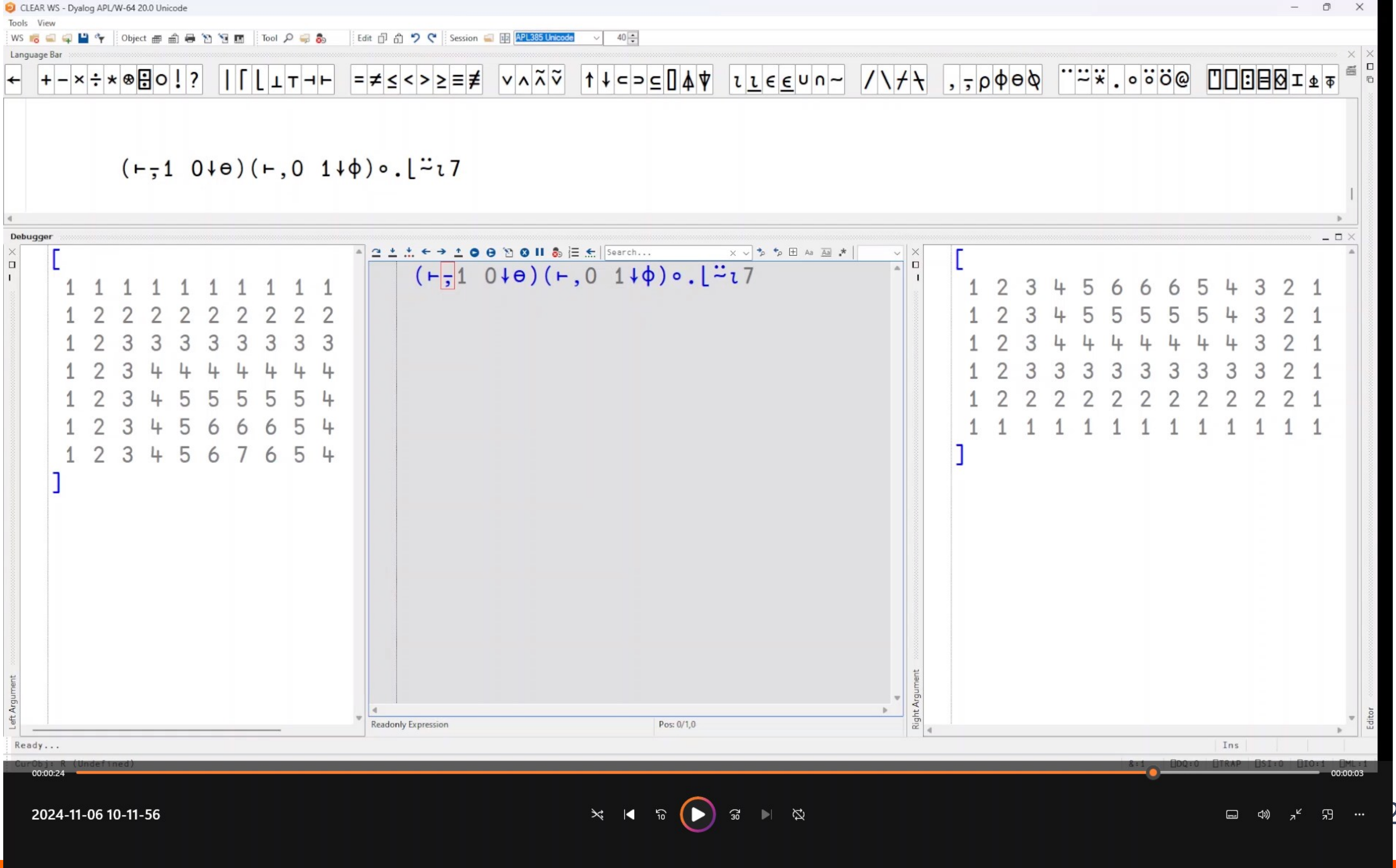
Readonly Expression   Pos: 0/1,0

Ready...   Ins

CurObj: R (Undefined)   &:1 ⎕DQ:0 ⎕TRAP ⎕SI:0 ⎕IO:1 ⎕ML:1

00:00:17   00:00:10

2024-11-06 10-11-56



CLEAR WS - Dyalog APL/W-64 20.0 Unicode

```
(⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
```

Debugger

Left Argument

```
[
 1 1 1 1 1 1 1 1 1 1
 1 2 2 2 2 2 2 2 2 2
 1 2 3 3 3 3 3 3 3 3
 1 2 3 4 4 4 4 4 4 4
 1 2 3 4 5 5 5 5 5 4
 1 2 3 4 5 6 6 6 5 4
 1 2 3 4 5 6 7 6 5 4
]
```

```
(⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
```

Readonly Expression   Pos: 0/1,0

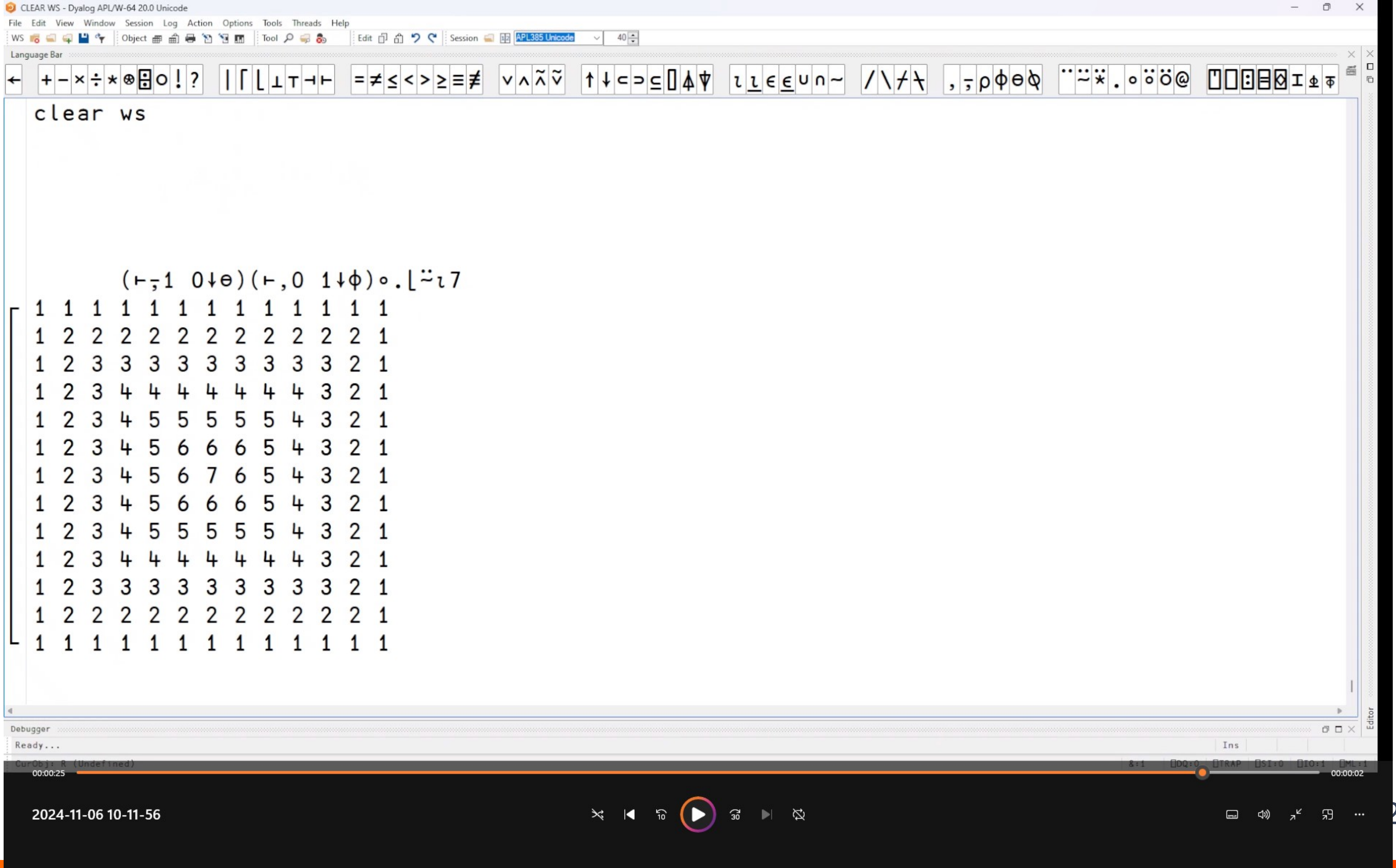
Right Argument

```
[
 1 2 3 4 5 6 6 6 5 4 3 2 1
 1 2 3 4 5 5 5 5 5 4 3 2 1
 1 2 3 4 4 4 4 4 4 4 3 2 1
 1 2 3 3 3 3 3 3 3 3 3 2 1
 1 2 2 2 2 2 2 2 2 2 2 2 1
 1 1 1 1 1 1 1 1 1 1 1 1 1
]
```

Ready...

00:00:24   00:00:03

2024-11-06 10-11-56


CLEAR WS - Dyalog APL/W-64 20.0 Unicode
File Edit View Window Session Log Action Options Tools Threads Help
Language Bar
Debugger
Ready...
Ins
00:00:25
00:00:02
2024-11-06 10-11-56

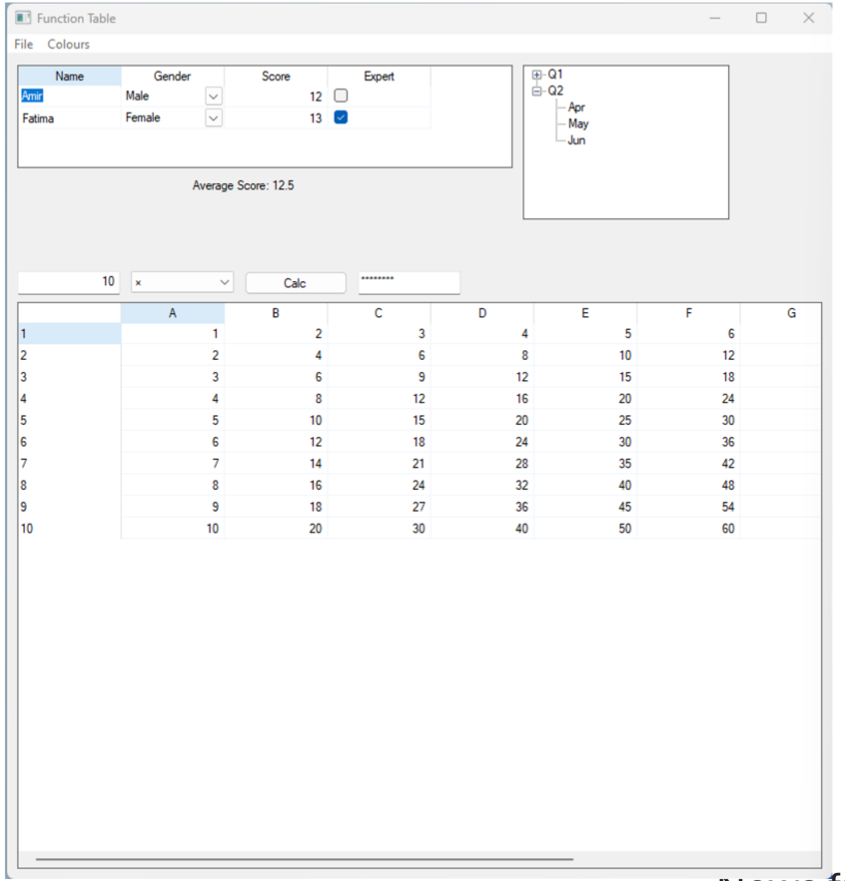
```
clear ws

      (⊢⍪1 0↓⊖)(⊢,0 1↓⌽)∘.⌊⍨⍳7
┌1 1 1 1 1 1 1 1 1 1 1 1 1
│1 2 2 2 2 2 2 2 2 2 2 2 1
│1 2 3 3 3 3 3 3 3 3 3 2 1
│1 2 3 4 4 4 4 4 4 4 3 2 1
│1 2 3 4 5 5 5 5 5 4 3 2 1
│1 2 3 4 5 6 6 6 5 4 3 2 1
│1 2 3 4 5 6 7 6 5 4 3 2 1
│1 2 3 4 5 6 6 6 5 4 3 2 1
│1 2 3 4 5 5 5 5 5 4 3 2 1
│1 2 3 4 4 4 4 4 4 4 3 2 1
│1 2 3 3 3 3 3 3 3 3 3 2 1
│1 2 2 2 2 2 2 2 2 2 2 2 1
└1 1 1 1 1 1 1 1 1 1 1 1 1
```

EWC – "Everywhere" WC

- A JavaScript emulation of Dyalog's Win32 wrappers (⎕WC, ⎕WG, ⎕WS ...)
- "Everywhere" means
  - Windows, Linux, macOS Desktops
  - Dyalog APL on any platform acting as a Web Server
    - Adds IBM AIX, Raspberry Pi

□WC↓

EWC→



Function Table

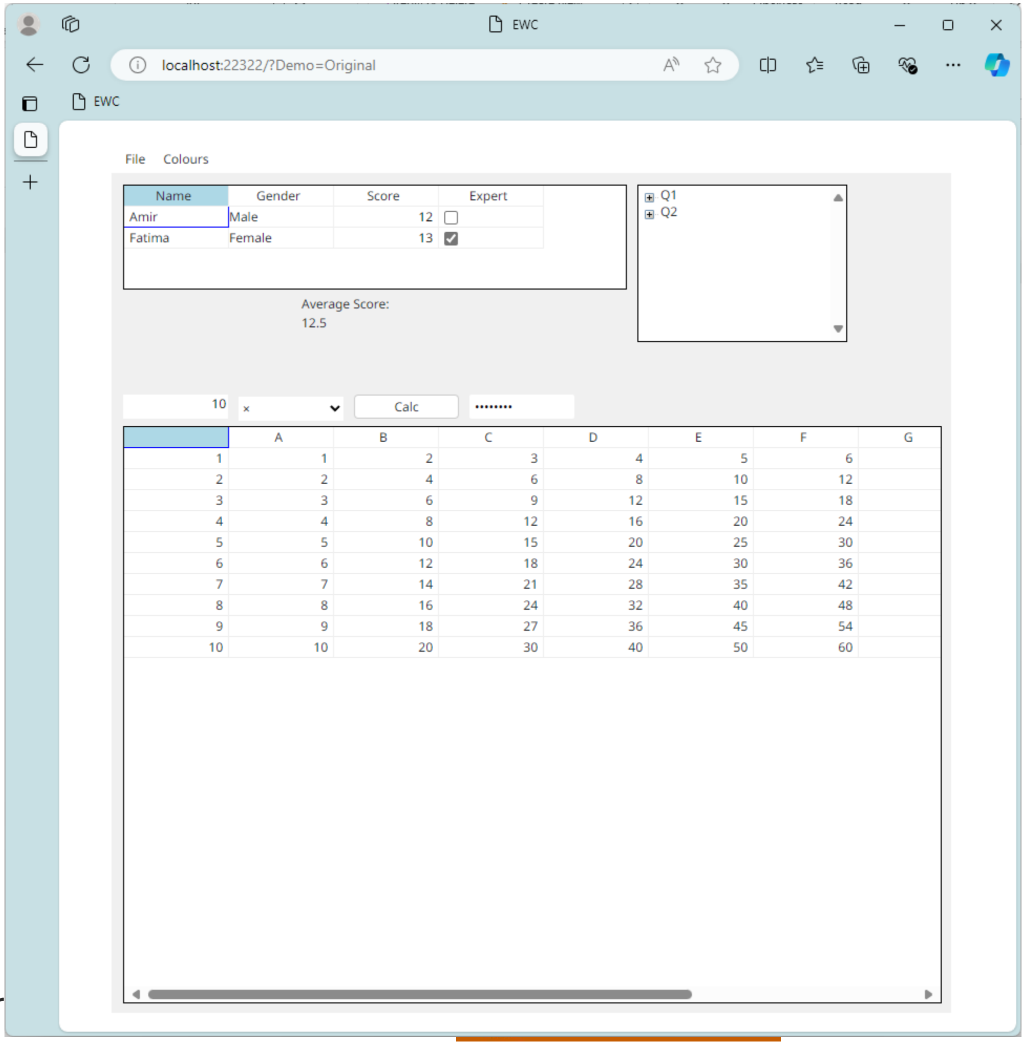
File Colours

Name	Gender	Score	Expert
Amir	Male	12	☐
Fatima	Female	13	☑

Average Score: 12.5

10 × Calc ********

	A	B	C	D	E	F	G
1	1	2	3	4	5	6	
2	2	4	6	8	10	12	
3	3	6	9	12	15	18	
4	4	8	12	16	20	24	
5	5	10	15	20	25	30	
6	6	12	18	24	30	36	
7	7	14	21	28	35	42	
8	8	16	24	32	40	48	
9	9	18	27	36	45	54	
10	10	20	30	40	50	60	



EWC

localhost:22322/?Demo=Original

EWC

File Colours

Name	Gender	Score	Expert
Amir	Male	12	☐
Fatima	Female	13	☑

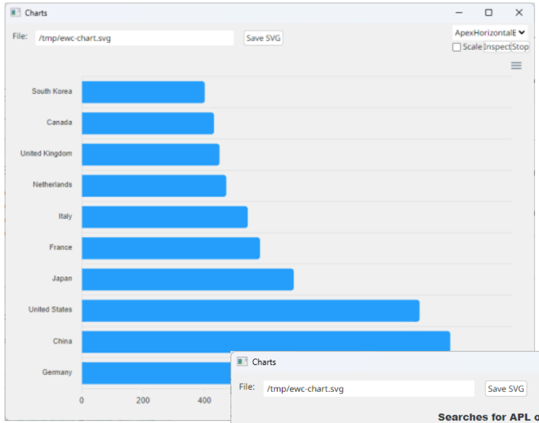
Average Score:
12.5

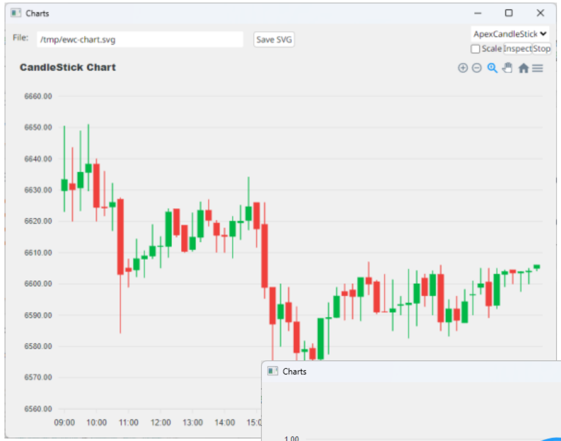
10 × Calc ********

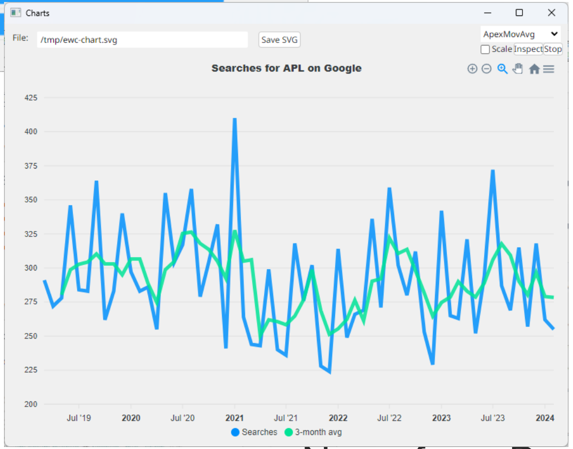
	A	B	C	D	E	F	G
1	1	2	3	4	5	6	
2	2	4	6	8	10	12	
3	3	6	9	12	15	18	
4	4	8	12	16	20	24	
5	5	10	15	20	25	30	
6	6	12	18	24	30	36	
7	7	14	21	28	35	42	
8	8	16	24	32	40	48	
9	9	18	27	36	45	54	
10	10	20	30	40	50	60	

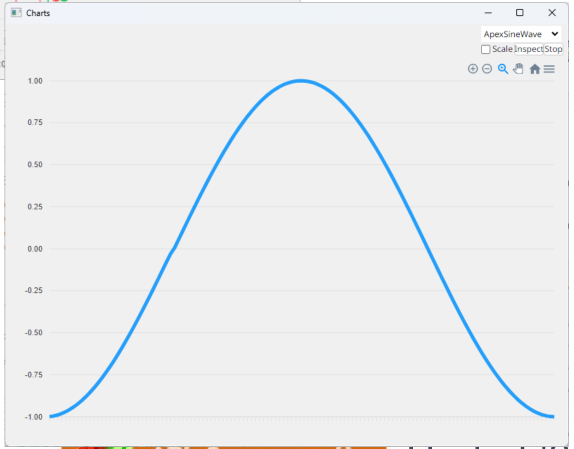
News fr

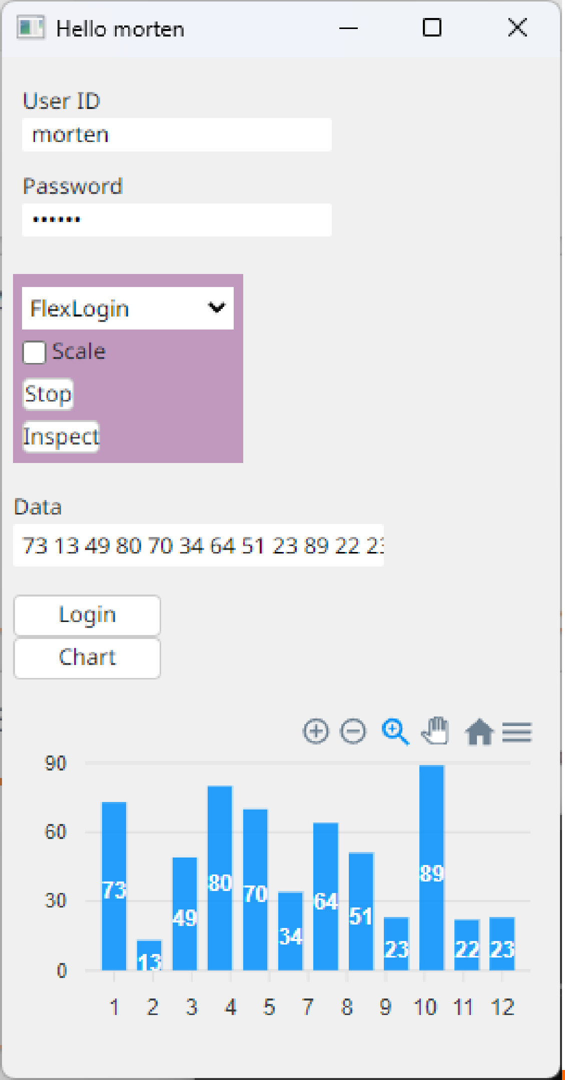
+ ApexCharts


Charts
File: /tmp/ewc-chart.svg
Save SVG
ApexHorizontalE
Scale Inspect Stop
South Korea
Canada
United Kingdom
Netherlands
Italy
France
Japan
United States
China
Germany
0
200
400


Charts
File: /tmp/ewc-chart.svg
Save SVG
ApexCandleStick
Scale Inspect Stop
CandleStick Chart
6660.00
6650.00
6640.00
6630.00
6620.00
6610.00
6600.00
6590.00
6580.00
6570.00
6560.00
09:00
10:00
11:00
12:00
13:00
14:00
15:0


Charts
File: /tmp/ewc-chart.svg
Save SVG
ApexMovAvg
Scale Inspect Stop
Searches for APL on Google
425
400
375
350
325
300
275
250
225
200
Jul '19
2020
Jul '20
2021
Jul '21
2022
Jul '22
2023
Jul '23
2024
● Searches ● 3-month avg


Charts
ApexSineWave
Scale Inspect Stop
1.00
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.00


Hello morten

User ID: morten

Password: ••••••

FlexLogin

☐ Scale

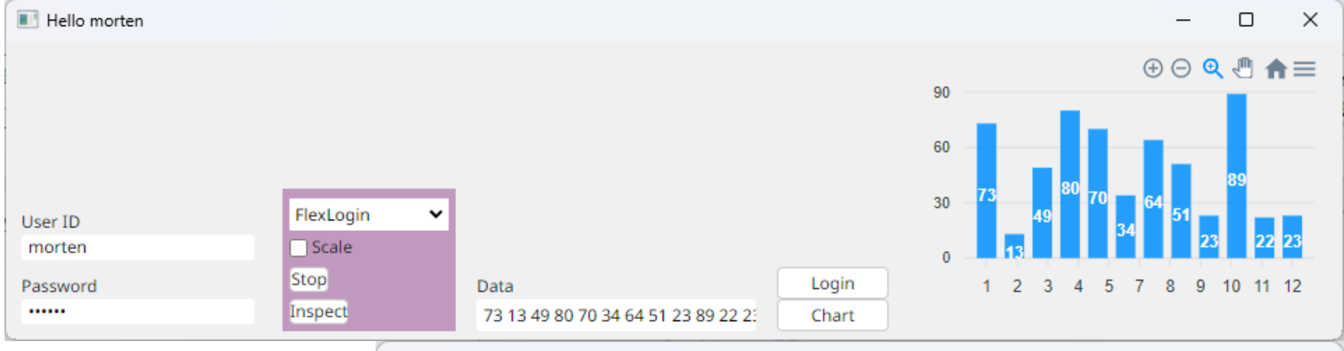
Stop

Inspect

Data: 73 13 49 80 70 34 64 51 23 89 22 23

Login

Chart


Hello morten

User ID: morten

Password: ••••••

FlexLogin

☐ Scale

Stop

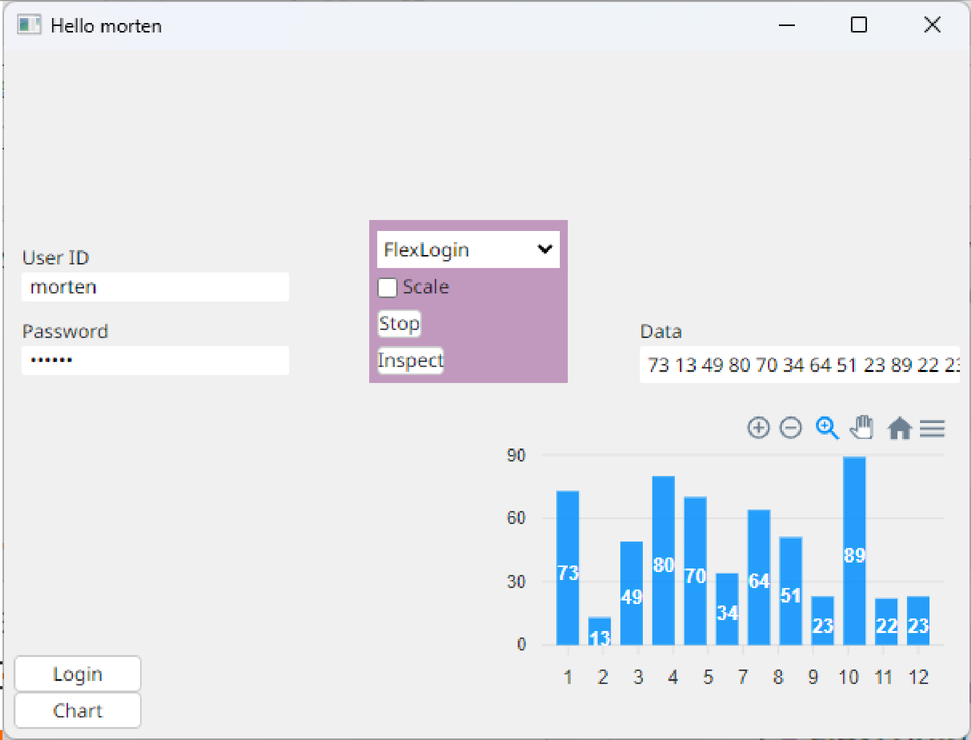
Inspect

Data: 73 13 49 80 70 34 64 51 23 89 22 23

Login

Chart

+ Flex Layout


Hello morten

User ID: morten

Password: ••••••

FlexLogin

☐ Scale

Stop

Inspect

Data: 73 13 49 80 70 34 64 51 23 89 22 23

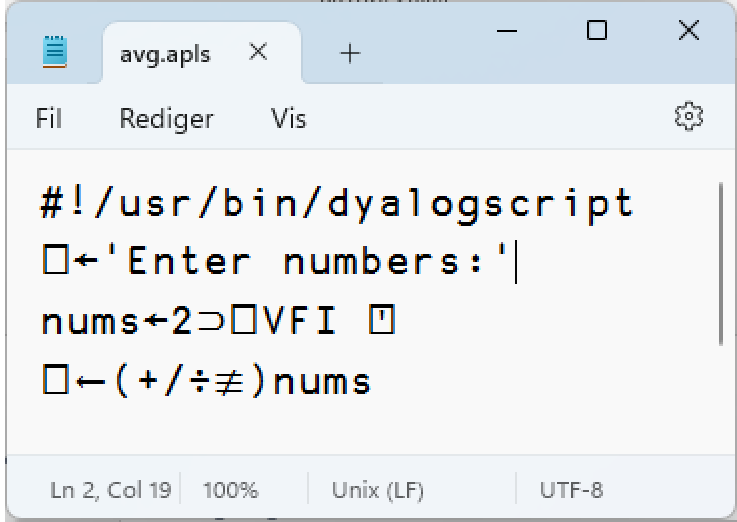
Login

Chart

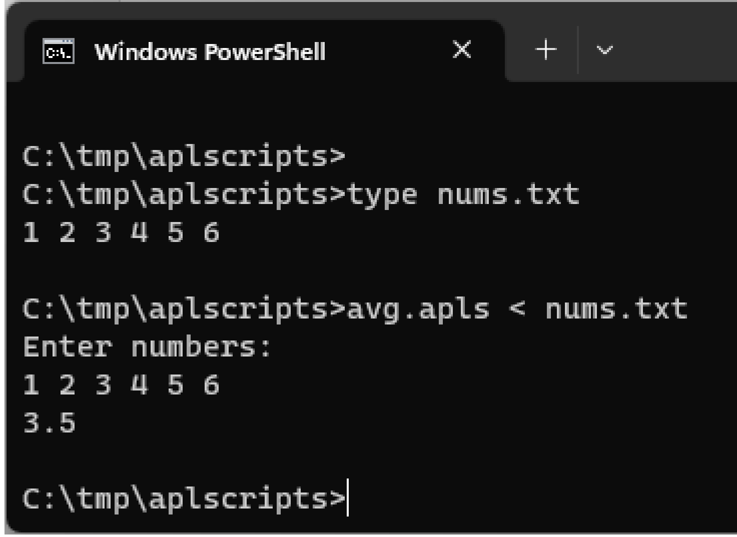
News from D

Script Support

- #! (hash bang) scripting
- Script engine is critical for new users
- Makes "Continuous Integration" easier
- Still a bit of a prototype in v19.0
- Will be improved in v20.0
    - Need to be able to debug scripts via RIDE



```
#!/usr/bin/dyalogscript
⎕←'Enter numbers:'
nums←2⊃⎕VFI ⍞
⎕←(+/÷≢)nums
```



```
C:\tmp\aplscripts>
C:\tmp\aplscripts>type nums.txt
1 2 3 4 5 6

C:\tmp\aplscripts>avg.apls < nums.txt
Enter numbers:
1 2 3 4 5 6
3.5

C:\tmp\aplscripts>
```

Behind / Reverse Compose

Dyadic: ⍺ f⍛g ⍵ ⇔ (f ⍺) g ⍵

```
      10⍳⍛∊2 3 5 8        ⍝ (⍳10)∊2 3 5 8
0 1 1 0 1 0 0 1 0 0
```

Behind / Reverse Compose

Monadic: `f⍛g ⍵` ←→ `(f ⍵) g ⍵`

```
      f←5∘<             ⍝ Predicate function
      f⍛⌿ 2 7 1 8 2 8   ⍝ Filter by f
7 8 8

      ⌈/⍛= 2 7 1 8 2 8 3 ⍝ Max behind Equal
0 0 0 1 0 1 0
```

D11: New Tatin Packages (Brian Becker)

Tatin

Package manager for Dyalog APL
(A tasty way to package APLs)

2023

```
     ]z←tatin.listPackages
     {⍺,≢⍵}⌸{(¯1+⍵⍳'-')↑⍵}¨3↓z[;1]
aplteam  42
davin     4
dyalog    2

     ¯2↑z
dyalog-HttpCommand  1
dyalog-Jarvis       1
```

2024

```
     ]z←tatin.listPackages
     {⍺,≢⍵}⌸{(¯1+⍵⍳'-')↑⍵}¨3↓z[;1]
aplteam  44
davin     4
dyalog    5 ⍝ 150% growth!

     ¯5↑z
dyalog-APLProcess   1
dyalog-HttpCommand  1
dyalog-Jarvis       1
dyalog-NuGet        1
dyalog-OpenAI       1
```

Medium Things on The Way

- New Shell System Function
- .NET Bridge Enhancements
- HTMLRenderer Enhancements
- Static Analysis of APL Code
- Health Monitor
- HTMLRenderer Enhancements

⎕SHELL to ~~replace~~ complement ⎕SH

Invoke OS commands from APL

- Interruptible
- Optionally return data as an asynchronous stream
- Manage stdin, stdout & stderr (& other streams) independently
- Handle variety of data encodings

D12: New Function for Shell Calls (Peter Mikkelsen)

.NET Bridge Enhancements

- The v19.0 bridge to .NET 6/7/8 is now equivalent to the Framework bridge
- New features will ONLY target the new .NET versions (8+):
  - Generics, Delegates
  - Async Methods (probably not v20.0)

Static Analysis of APL Code

- Static Analysis of application code is seen as a required "best practice" by some corporations
- We are building a prototype of a tool which will
  - Detect vulnerabilities and other bad practices
  - "Lint" APL Code
- This tool will initially be licensed separately
- A free "community edition" may follow

D06 (22/11): Static Analysis of APL in APL (Brandon Wilson)

D13 (22/11): Co-dfns Roadmap and Updates (Aaron Hsu)

HTMLRenderer Enhancements

The HTMLRenderer continues to grow in importance

- File Upload
- Modal HTMLRenderer Windows
- More control over "Chrome"
  - The EWC project will accelerate the evolution of the HTMLRenderer (more tomorrow)

Small Things

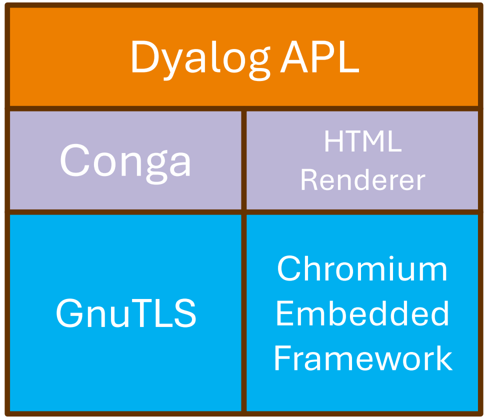
- ⎕NINFO w/Callbacks
- Set File Attributes using ⎕NATTRIBUTES
- Unicode decomposition / normal forms

Laying New Foundations

- ⎕WC Plugin Mechanism
- Relax Interpreter Limits
- Artificial Intelligence

An Open [Source] Plugin Architecture

- A modern replacement for Auxiliary Processors
  - Uses Direct Workspace Access (DWA) for high performance
  - Accessible via ⎕WC / ⎕NEW
- Make interpreter extensions open source, eg.
  - HTMLRenderer, Conga (our TCP platform)
  - Cryptographic Library
- Many of these are interfaces to open source components

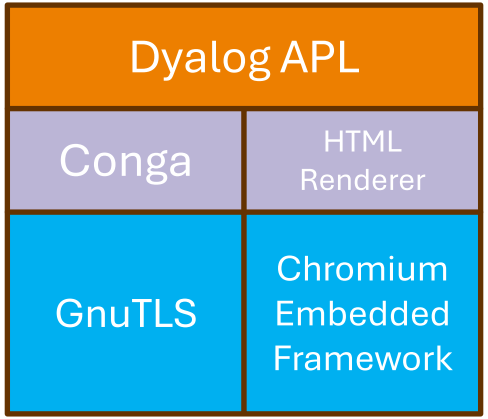


Plugin Architecture Benefits

- Allow users to move faster
  - Adopt new versions of Chromium or OpenSSL
- Make the Dyalog community more inclusive
  - Allow users to contribute to our eco-system
- Users can develop and share new extensions
- Makes our encryption tools transparent and verifyable

➔ Easier to comply with FOSS licence constraints

D05: WC Plugins (John Daintree)



Relax Interpreter Limits

We plan to relax limitations in the interpreter, like:

- Max Rank (15)
- Number of tokens in a line (4,095)
- Number of token types
- (and many more)

Relax Limits – Why Now?

- We need more token types for new primitives, new control structures, **array notation**, ...
- Migrants to Dyalog have functions with >9,999 lines ☺☹
- Challenge: Structure has not changed for decades
  - (except adding new types like Unicode and Complex Numbers)
- Must avoid "big bang" data conversion
  - Different versions of APL must share data
  - (Restore & use archived component files)

D16: Interpreter Limits (John Daintree)

Artificial Intelligence

Research Into:

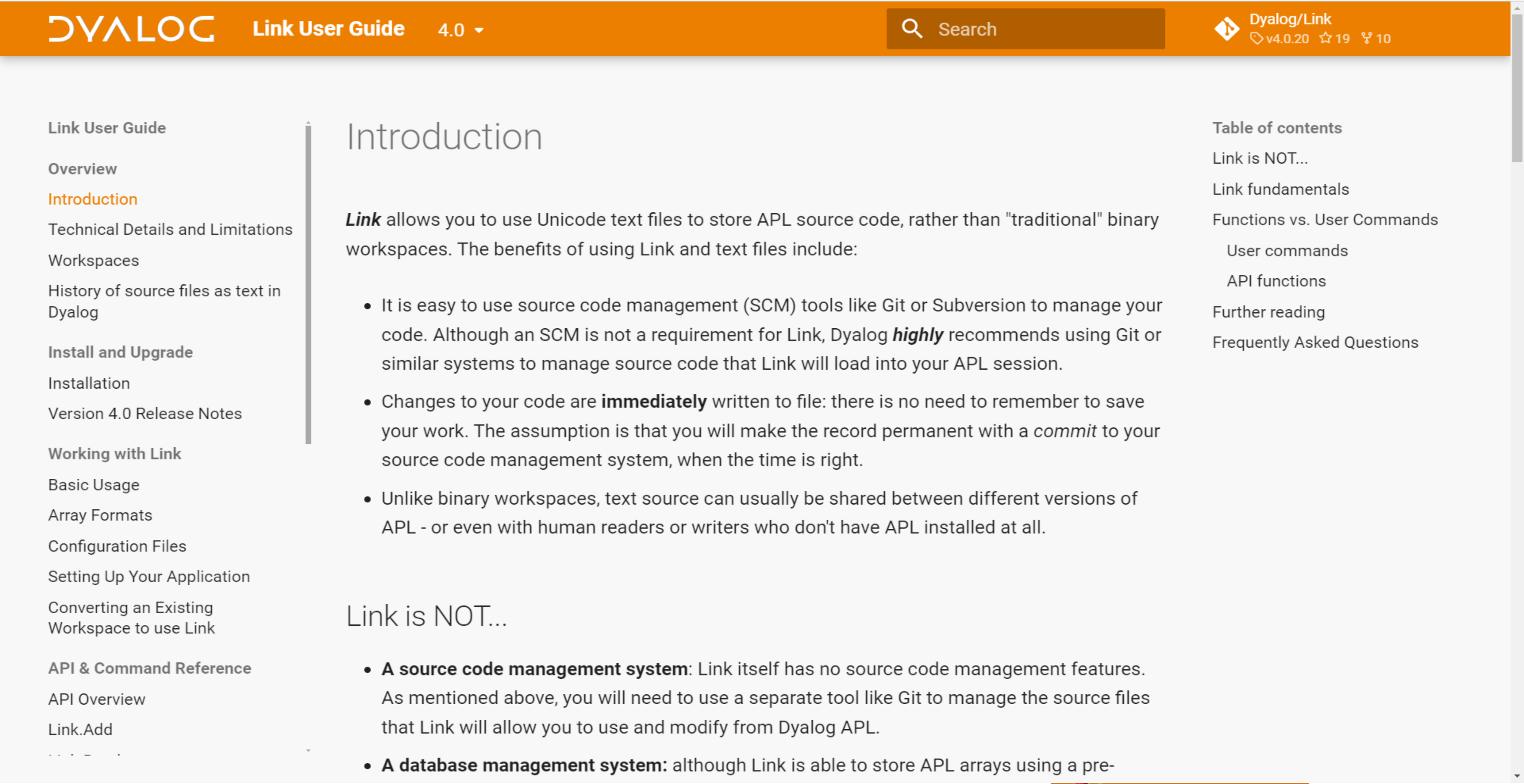
- OpenAI interface (Tatin Package)
- Using "Language Models" to search our Documentation
- Are Language Models good enough for a meaningful "APL Co-Pilot?"
  - (at least for newcomers to APL)

Tidying Up

- PCRE 10.x
  - Upgrade ⎕R & ⎕S to use latest PCRE
- Documentation Format

Documentation

- help.dyalog.com and "core" documentation will be produced using **MkDocs** on **GitHub**
- Anyone (including **you**) can raise "issues"
  - Or even submit "Pull requests"
  - You **CAN** still email docs@ or support@
- Some tools already use MkDocs...


DYALOG Link User Guide 4.0

Search

Dyalog/Link v4.0.20 19 10

Introduction

Link allows you to use Unicode text files to store APL source code, rather than "traditional" binary workspaces. The benefits of using Link and text files include:

- It is easy to use source code management (SCM) tools like Git or Subversion to manage your code. Although an SCM is not a requirement for Link, Dyalog ***highly*** recommends using Git or similar systems to manage source code that Link will load into your APL session.
- Changes to your code are **immediately** written to file: there is no need to remember to save your work. The assumption is that you will make the record permanent with a *commit* to your source code management system, when the time is right.
- Unlike binary workspaces, text source can usually be shared between different versions of APL - or even with human readers or writers who don't have APL installed at all.

Link is NOT...

- **A source code management system**: Link itself has no source code management features. As mentioned above, you will need to use a separate tool like Git to manage the source files that Link will allow you to use and modify from Dyalog APL.
- **A database management system:** although Link is able to store APL arrays using a pre-

Documentation: Why change?

- Easier to contribute
  - Internally and externally (and Pete has retired)
- Open formats, platform agnostic tools
- Better search
- Human-friendly, predictable URLs, like https://docs.dyalog.com/20.0/object-reference/properties/depth/

Dyalog APL v20.0 Documentation

Welcome! This is the official documentation for Dyalog APL version 20.0.

Release Notes v20.0

New and improved since the last release

- Release Notes

Installation and Configuration

How to install and configure Dyalog APL

- Windows Installation and Configuration Guide
- UNIX Installation and Configuration Guide

Reference Guides

Reference guides for Dyalog APL and system interfaces

- Programming Reference Guide
- Dyalog APL Language Reference Guide
- Object Reference Guide
- Microsoft Windows: Interface Guide
- Microsoft Windows: .NET Framework Interface Guide

UI Guides

The Dyalog APL Development Environment

- Microsoft Windows UI Guide
- UNIX User Guide

Other Stuff

- Link v4.1
- Kafka Interface
- Telemetry
- BSIMM Audit
- Isolates connect back to server
  - (Docker containers)

- Performance
  - Set functions
  - NSs kept alive by children
  - Garbage Collector
- Exhaustive Primitive Tests

- 64-bit ARM support
  - (Pi & AWS Image)
- New Mac installer
- Directory naming
- Keyboard layouts
- Platform Features

Big Things

- Array Notation
- Set & Get Variables
- Token-by-Token Tracing
- Everywhere WC
- Script Support
- Reverse Compose

New Foundations

- ⎕WC Plugin Mechanism
  - Open-source more components
  - HTMLRenderer, Conga, Crypto Library
- Relax Interpreter Limits
- New UCMD mechanism

Thank You!