



Dyalog APL: What It Is and How It Works?

Aarush Bhat
Dyalog Ltd.



Coming up next...

APL

A Programming Language

What is APL?

Symbolic
Array-oriented Programming Language
for Communicating Algorithms
to Computers
and Humans

What is APL?

Symbolic



What is APL?

Symbolic
Array-oriented Programming Language

(5 , 6 , 7 , 8)

What is APL?

Symbolic
Array-oriented Programming Language

$$2 + (5 , 6 , 7 , 8)$$

7 8 9 10

What is APL?

Symbolic
Array-oriented Programming Language

$2 \downarrow (5, 6, 7, 8)$

7 8

What is APL?

Symbolic
Array-oriented Programming Language

$$2 \uparrow (5, 6, 7, 8)$$

5 6

What is APL?

Symbolic
Array-oriented Programming Language

$\phi(5, 6, 7, 8)$

8 7 6 5

What is APL?

Symbolic
Array-oriented Programming Language

table

T	r	y
A	P	L
n	o	w

φtable

y	r	T
L	P	A
w	o	n

What is APL?

Symbolic

Array-oriented Programming Language

table

T	r	y
A	P	L
n	o	w

$2 \downarrow \phi$ table

w	o	n
---	---	---

What is APL?

Symbolic

Array-oriented Programming Language

table

T	r	y
A	P	L
n	o	w

2 1↓φtable

o	n
---	---

What is APL?

Symbolic

Array-oriented Programming Language

table

T	r	y
A	P	L
n	o	w

q2 1↓φtable

o
n

Kenneth E. Iverson

[Article](#) [Talk](#)

From Wikipedia, the free encyclopedia

Kenneth Eugene Iverson (17 December 1920 – 19 October 2004) was a Canadian [computer scientist](#) noted for the development of the [programming language APL](#). He was honored with the [Turing Award](#) in 1979 "for his pioneering effort in programming languages and mathematical notation resulting in what the computing field now knows as APL; for his contributions to the implementation of interactive systems, to educational uses of APL, and to programming language theory and practice".^[1]

Kenneth E. Iverson



Born	17 December 1920 Camrose, Alberta, Canada
Died	19 October 2004 (aged 83) Toronto, Canada
Alma mater	Queen's University Harvard University
Known for	Programming languages: APL, J
Awards	IBM Fellow Harry H. Goode Memorial Award Turing Award Computer Pioneer Award

Symbols, symbols and symbols



Myth: “APL is Unreadable”

$$(\times / ! x - 1) \div ! (+ / x) - 1$$

$$\frac{\prod_{i=1}^n (x_i - 1)!}{\left(\left(\sum_{i=1}^n x_i \right) - 1 \right)!}$$

Математичка функција

دالة رياضية

數
學
函
數

Myth: “APL is Unreadable”

```
x.map(e=>[...Array(e).keys()].slice(1).reduce((a,b)=>a*b,1)).reduce((a,b)=>a*b)  
/[...Array(x.reduce((a,b)=>a+b)).keys()].slice(1).reduce((a,b)=>a*b,1)
```

```
Fac=x=>[...Array(x+1).keys()].slice(1).reduce((a,b)=>a*b,1)  
x.map(e=>Fac(e-1)).reduce((a,b)=>a*b)/Fac(x.reduce((a,b)=>a+b)-1)
```

```
Sum=x=>x.reduce((a,b)=>a+b,0)  
Prd=x=>x.reduce((a,b)=>a*b,1)  
Rng=x=>[...Array(x).keys()]  
Fac=x=>Prd(Rng(x+1).slice(1))  
Prd(x.map(e=>Fac(e-1)))/Fac(Sum(x)-1)
```

Myth: “APL is Unreadable”

```
x.map(e=>[...Array(e).keys()].slice(1).reduce((a,b)=>a*b,1)).reduce((a,b)=>a*b)  
/[...Array(x.reduce((a,b)=>a+b)).keys()].slice(1).reduce((a,b)=>a*b,1)
```

```
Fac=x=>[...Array(x+1).keys()].slice(1).reduce((a,b)=>a*b,1)  
x.map(e=>Fac(e-1)).reduce((a,b)=>a*b)/Fac(x.reduce((a,b)=>a+b)-1)
```

Sum = x => x.reduce((a, b) => a + b, 0)	Sum ← +/
Prd = x => x.reduce((a, b) => a * b, 1)	Prd ← ×/
Rng = x => [... Array(x).keys()]	Rng ← ↵
Fac = x => Prd(Rng(x + 1).slice(1))	Fac ← Prd 1 ↓ (Rng +○1)
Prd(x.map(e => Fac(e - 1))) / Fac(Sum(x) - 1)	(Prd Fac``x - 1) ÷ Fac(Sum x) - 1

Myth: “APL is Unreadable”

```
x.map(e=>[...Array(e).keys()].slice(1).reduce((a,b)=>a*b,1)).reduce((a,b)=>a*b)  
/[...Array(x.reduce((a,b)=>a+b)).keys()].slice(1).reduce((a,b)=>a*b,1)
```

```
Fac=x=>[...Array(x+1).keys()].slice(1).reduce((a,b)=>a*b,1)  
x.map(e=>Fac(e-1)).reduce((a,b)=>a*b)/Fac(x.reduce((a,b)=>a+b)-1)
```

```
Sum = x => x.reduce((a, b) => a + b, 0)
```

```
Prd = x => x.reduce((a, b) => a * b, 1)
```

```
Rng = x => [... Array(x).keys()]
```

```
Fac = x => Prd(Rng(x + 1).slice(1))
```

```
Prd(x.map(e => Fac(e - 1))) / Fac(Sum(x) - 1)
```

```
Sum ← +/
```

```
Prd ← ×/
```

```
Rng ← ↳
```

```
Fac ← Prd 1 ↓ (Rng +○1)
```

```
(Prd Fac``x - 1) ÷ Fac(Sum x) - 1
```

Myth: “APL is Unreadable”

```
x.map(e=>[...Array(e).keys()].slice(1).reduce((a,b)=>a*b,1)).reduce((a,b)=>a*b)  
/[...Array(x.reduce((a,b)=>a+b)).keys()].slice(1).reduce((a,b)=>a*b,1)
```

```
Fac=x=>[...Array(x+1).keys()].slice(1).reduce((a,b)=>a*b,1)  
x.map(e=>Fac(e-1)).reduce((a,b)=>a*b)/Fac(x.reduce((a,b)=>a+b)-1)
```

```
Sum = x => x.reduce((a, b) => a + b, 0)
```

```
Prd = x => x.reduce((a, b) => a * b, 1)
```

```
Rng = x => [... Array(x).keys()]
```

```
Fac = x => Prd(Rng(x + 1).slice(1))
```

```
Prd(x.map(e => Fac(e - 1))) / Fac(Sum(x) - 1)
```

```
Sum ← +/
```

```
Prd ← ×/
```

```
Rng ← ↵
```

```
Fac ← Prd 1 ↓ (Rng +○1)
```

```
(Prd Fac``x - 1) ÷ Fac(Sum x) - 1
```

Myth: “APL is Unreadable”

```
x.map(e=>[...Array(e).keys()].slice(1).reduce((a,b)=>a*b,1)).reduce((a,b)=>a*b)  
/[...Array(x.reduce((a,b)=>a+b)).keys()].slice(1).reduce((a,b)=>a*b,1)
```

```
Fac=x=>[...Array(x+1).keys()].slice(1).reduce((a,b)=>a*b,1)  
x.map(e=>Fac(e-1)).reduce((a,b)=>a*b)/Fac(x.reduce((a,b)=>a+b)-1)
```

```
Sum = x => x.reduce((a, b) => a + b, 0)
```

```
Prd = x => x.reduce((a, b) => a * b, 1)
```

```
Rng = x => [... Array(x).keys()]
```

```
Fac = x => Prd(Rng(x + 1).slice(1))
```

```
Prd(x.map(e => Fac(e - 1))) / Fac(Sum(x) - 1)
```

```
Sum  $\leftarrow$  +/
```

```
Prd  $\leftarrow$  ×/
```

```
Rng  $\leftarrow$  1
```

```
Fac  $\leftarrow$  Prd 1 ↓ (Rng +o1)
```

```
(Prd Fac``x - 1) ÷ Fac(Sum x) - 1
```

Myth: “APL is Unreadable”

```
x.map(e=>[...Array(e).keys()].slice(1).reduce((a,b)=>a*b,1)).reduce((a,b)=>a*b)  
/[...Array(x.reduce((a,b)=>a+b)).keys()].slice(1).reduce((a,b)=>a*b,1)
```

```
Fac=x=>[...Array(x+1).keys()].slice(1).reduce((a,b)=>a*b,1)  
x.map(e=>Fac(e-1)).reduce((a,b)=>a*b)/Fac(x.reduce((a,b)=>a+b)-1)
```

```
Sum = x => x.reduce((a, b) => a + b, 0)
```

```
Sum ← +/
```

```
Prd = x => x.reduce((a, b) => a * b, 1)
```

```
Prd ← ×/
```

```
Rng = x => [... Array(x).keys()]
```

```
Rng ← ↵
```

```
Fac = x => Prd(Rng(x + 1).slice(1))
```

```
Fac ← Prd 1 ↓ (Rng +○1)
```

```
Prd(x.map(e => Fac(e - 1))) / Fac(Sum(x) - 1)
```

```
(Prd Fac``x - 1) ÷ Fac(Sum x) - 1
```

Myth: “APL is Unreadable”

```
x.map(e=>[...Array(e).keys()].slice(1).reduce((a,b)=>a*b,1)).reduce((a,b)=>a*b)  
/[...Array(x.reduce((a,b)=>a+b)).keys()].slice(1).reduce((a,b)=>a*b,1)
```

```
Fac=x=>[...Array(x+1).keys()].slice(1).reduce((a,b)=>a*b,1)  
x.map(e=>Fac(e-1)).reduce((a,b)=>a*b)/Fac(x.reduce((a,b)=>a+b)-1)
```

```
Sum = x => x.reduce((a, b) => a + b, 0)
```

```
Sum ← +/
```

```
Prd = x => x.reduce((a, b) => a * b, 1)
```

```
Prd ← ×/
```

```
Rng = x => [... Array(x).keys()]
```

```
Rng ← ↵
```

```
Fac = x => Prd(Rng(x + 1).slice(1))
```

```
Fac ← Prd 1 ↓ (Rng +○1)
```

```
Prd(x.map(e => Fac(e - 1))) / Fac(Sum(x) - 1)
```

```
(Prd Fac''x - 1) ÷ Fac(Sum x) - 1
```

Myth: “APL is Unreadable”

```
x.map(e=>[...Array(e).keys()].slice(1).reduce((a,b)=>a*b,1)).reduce((a,b)=>a*b)  
/[...Array(x.reduce((a,b)=>a+b)).keys()].slice(1).reduce((a,b)=>a*b,1)
```

```
Fac=x=>[...Array(x+1).keys()].slice(1).reduce((a,b)=>a*b,1)  
x.map(e=>Fac(e-1)).reduce((a,b)=>a*b)/Fac(x.reduce((a,b)=>a+b)-1)
```

```
Sum = x => x.reduce((a, b) => a + b, 0)
```

```
Prd = x => x.reduce((a, b) => a * b, 1)
```

```
Rng = x => [... Array(x).keys()]
```

```
Fac = x => Prd(Rng(x + 1).slice(1))
```

```
Prd(x.map(e => Fac(e - 1))) / Fac(Sum(x) - 1)
```

```
Sum ← +/
```

```
Prd ← ×/
```

```
Rng ← ⌊
```

```
Fac ← Prd 1 ↓ (Rng +•1)
```

```
(Prd Fac x - 1) ÷ Fac(Sum x) - 1
```

Myth: “APL is Unreadable”

```
x.map(e=>[...Array(e).keys()].slice(1).reduce((a,b)=>a*b,1)).reduce((a,b)=>a*b)  
/[...Array(x.reduce((a,b)=>a+b)).keys()].slice(1).reduce((a,b)=>a*b,1)
```

```
Fac=x=>[...Array(x+1).keys()].slice(1).reduce((a,b)=>a*b,1)  
x.map(e=>Fac(e-1)).reduce((a,b)=>a*b)/Fac(x.reduce((a,b)=>a+b)-1)
```

```
Sum = x => x.reduce((a, b) => a + b, 0)
```

Sum $\leftarrow +/$

```
Prd = x => x.reduce((a, b) => a * b, 1)
```

Prd $\leftarrow \times/$

```
Rng = x => [... Array(x).keys()]
```

```
Fac = x => Prd(Rng(x + 1).slice(1))
```

Fac $\leftarrow !$

```
Prd(x.map(e => Fac(e - 1))) / Fac(Sum(x) - 1)
```

(Prd Fac x - 1) \div Fac(Sum x) - 1

Myth: “APL is Unreadable”

```
x.map(e=>[...Array(e).keys()].slice(1).reduce((a,b)=>a*b,1)).reduce((a,b)=>a*b)  
/[...Array(x.reduce((a,b)=>a+b)).keys()].slice(1).reduce((a,b)=>a*b,1)
```

```
Fac=x=>[...Array(x+1).keys()].slice(1).reduce((a,b)=>a*b,1)  
x.map(e=>Fac(e-1)).reduce((a,b)=>a*b)/Fac(x.reduce((a,b)=>a+b)-1)
```

```
Sum = x => x.reduce((a, b) => a + b, 0)
```

```
Prd = x => x.reduce((a, b) => a * b, 1)
```

```
Rng = x => [... Array(x).keys()]
```

```
Fac = x => Prd(Rng(x + 1).slice(1))
```

```
Prd(x.map(e => Fac(e - 1))) / Fac(Sum(x) - 1)
```

Sum $\leftarrow +/$

Prd $\leftarrow \times/$

Fac $\leftarrow !$

(Prd Fac x - 1) \div Fac(Sum x) - 1

Myth: “APL is Unreadable”

```
x.map(e=>[...Array(e).keys()].slice(1).reduce((a,b)=>a*b,1)).reduce((a,b)=>a*b)  
/[...Array(x.reduce((a,b)=>a+b)).keys()].slice(1).reduce((a,b)=>a*b,1)
```

```
Fac=x=>[...Array(x+1).keys()].slice(1).reduce((a,b)=>a*b,1)  
x.map(e=>Fac(e-1)).reduce((a,b)=>a*b)/Fac(x.reduce((a,b)=>a+b)-1)
```

```
Sum = x => x.reduce((a, b) => a + b, 0)  
Prd = x => x.reduce((a, b) => a * b, 1)  
Rng = x => [... Array(x).keys()]  
Fac = x => Prd(Rng(x + 1).slice(1))  
Prd(x.map(e => Fac(e - 1))) / Fac(Sum(x) - 1) ( ×/ ! x - 1 ) ÷ ! ( +/ x ) - 1
```

Myth: “APL is Unreadable”

```
x.map( (×/!x - 1) ÷ !(+/x) - 1 )
```

```
Fac=x=>[...Array(x+1).keys()].slice(1).reduce((a,b)=>a*b,1)
```

```
x.map(e=>Fac(e-1)).reduce((a,b)=>a*b)/Fac(x.reduce((a,b)=>a+b)-1)
```

```
Sum = x => x.reduce((a, b) => a + b, 0)
```

```
Prd = x => x.reduce((a, b) => a * b, 1)
```

```
Rng = x => [... Array(x).keys()]
```

```
Fac = x => Prd(Rng(x + 1).slice(1))
```

```
Prd(x.map(e => Fac(e - 1))) / Fac(Sum(x) - 1)    (×/!x-1)÷!(+/x)-1
```

But, where is this used?

^ DI in Research

algebra
h
city



$-VP + \Delta(TS)$ (TS förf.)
A vector pro

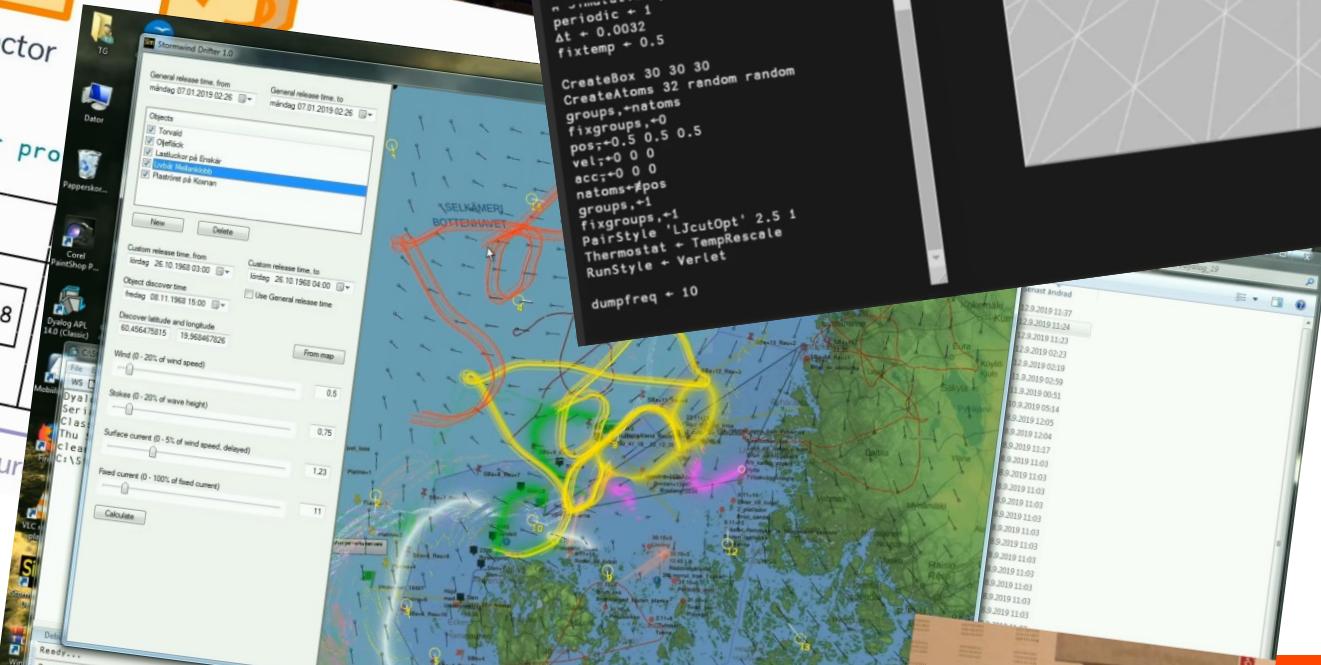
2

6 7 8

26 44 0 0 0 0 8 8

APL and Metallur

dyalog.com



APL in Academia



Deutsch-Jozsa algorithm

*_DJ_ + {
 the state according the ancilla qubit.*

*prep+{
 ancilla state to apply X and SWAP to the
 qubit
 state) ◇*

- ▶ Problem 1: To find 4 unique, positive digit numbers {1...9} which make up a sum of 29.

- ▶ There is just one way : 5 7 8 9

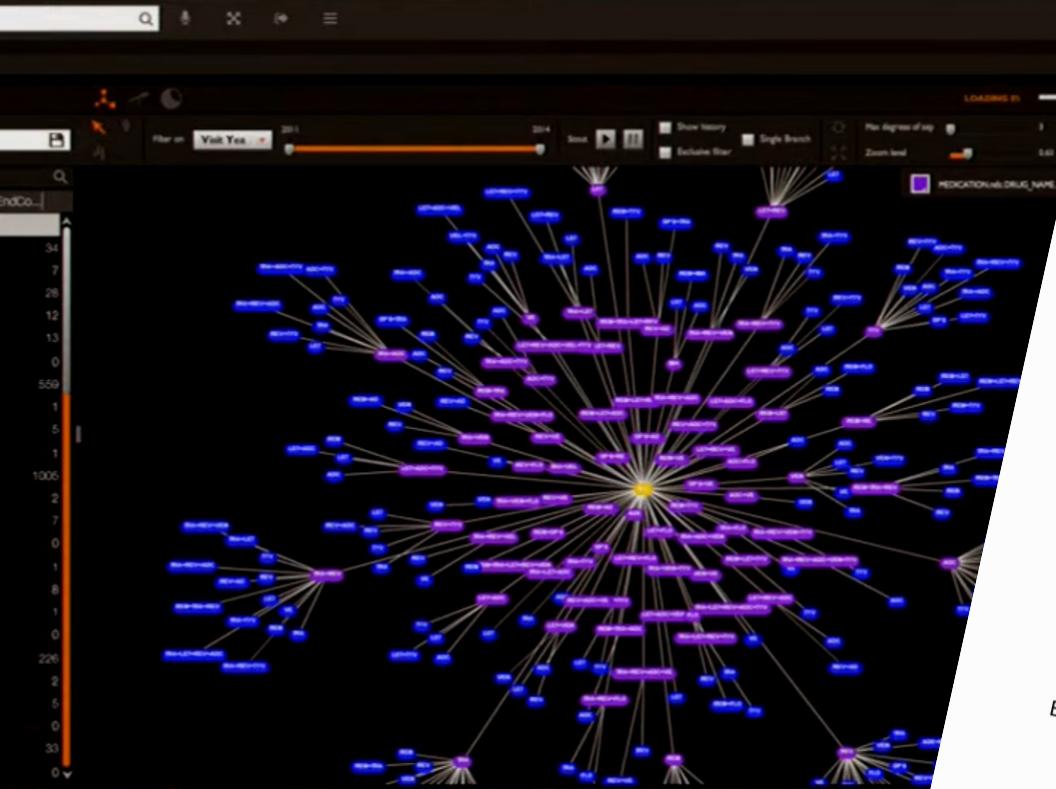
▶ _____

```
▶ Sort←{w[⍋w]}  
▶ clean←{ vSort" ({w≡u w}"w)/w }  
▶ NCat←{ ., *(α-1)~ w }  
▶ sum←{ok←w=+/''all+, α NCat ⌊9 ◇ all←ok/all ◇ clean  
  all }  
▶ 4 sum 29  
▶ (5 7 8 9)
```



A PROBLEM OF COMBINATORIAL MATHS

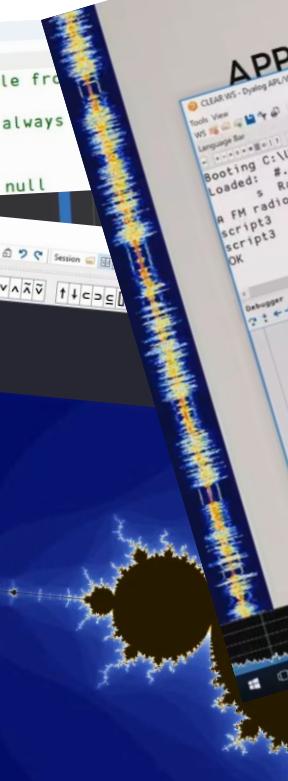
APL in Industry



- Simplified version of our custom Dockerfile
 - Base Image → `FROM redhat/ubi8-minimal:8.8`
 - Add all components (specify a version!) → `ADD APISource /app`
`ADD linux_64_18.2.45405_unicode.x86_64`
`RUN git clone https://github.com/dyalo...`
 - Specify environment variables → `ENV JarvisConfig="/app/Config.json"`
`ENV LOAD="/Jarvis/Source..."`
 - Executable which runs at startup → `ENTRYPOINT ["/app/entrypoint.sh"]`

And more...

```
WriteBitChord
File Edit Syntax Refactor View
Search... x v A
tie WriteLong 0
A Size to end of file from
A 9 -12 "WAVE" File Type Header. For our purposes, it always
A .WAV tag
tie WriteChar'WAVE'
A 13-16 "fmt" Format
tie WriteChar'fmt'
A 17-20 "16" Length
tie WriteLong 16
A 21-22 "1" Type
tie WriteShort 1
A 23-24 "2" Number
chans<-2
tie WriteShort chans
A 25-28 "44100"
A Common values
A Sample Rate = 44000
rate<-44000
tie WriteLong rate
A 29-32 "176400"
bits<-8
size-rate(bits)
tie WriteLong size
A 33-34 "4" (bits)
    256 pal[;its]
3355443 3355443 5066061 6710886 1
3355443 5066061 8421504 0 0
3355443 0 0 0 0
3355443 5066061 8421504 0
3355443 3355443 5066061 6710886 1
pixmat + 256 pal[;its]
'C:/tmp/webinar/test.png' p
its + 50 Mandelbrot -.6 (3
pal + GreyPalette 50
pixmat + 256 pal[;its]
'C:/tmp/webinar/test.png' p
)ed Palette
```



```
APPLICATION·FM RADIO
CLEAR WS - Dyalog APL/W - [Home]
File Edit Syntax Refactor View
Search... x v A
Booting C:\users\moris\Dropbox\APL\src\workingdir\demo_scr3.dyapp
Loaded: #.filters #.gr question #.rds script1 script2 script3 script4 #.sndx
A FM radio
scripts
OK
samp_d.ipassfilter samp_d(u.Catch t_bw fs)(ssamp_d)
samp_d((!samp_d)pder_rate(t))/samp_d
View gr.rplot 10000t samp_d
samp_d-(+/#) samp_d
View gr.rplot 10000t samp_d
samp_d sndx.PlaySound 16, #Fs_new
Last saved by moris martedì 12 settembre 2017 15:38
Function
Ready...
CurObj:
```



DYALOG

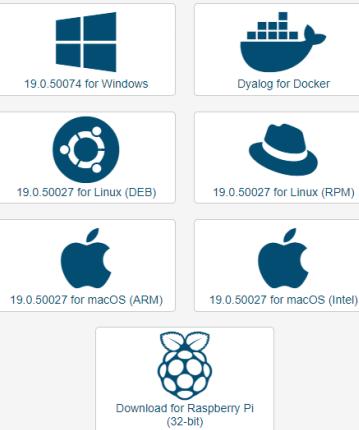
Dyalog Interpreter



Download Dyalog for Free

Use the buttons below to download Dyalog APL for your platform of choice.

Setup instructions for all platforms can be found here: [setup_readme](#)



Dyalog APL: What It Is and How It Works?

DYALOC

Co-dfns Compiler

The screenshot shows the GitHub repository page for 'Co-dfns'. The repository is public and has 5,385 commits. It features a sidebar with navigation links like Code, Pull requests, Actions, Security, and Insights. The main area displays a list of recent commits, including one from 'arcfide' adding support for parsing with globally free values. On the right, there's an 'About' section with details about the repository, such as its purpose ('High-performance, Reliable, and Parallel API'), license ('AGPL-3.0, Unknown licenses found'), and activity metrics (702 stars, 32 watching, 32 forks). A 'Report repository' button is also present.

Code Pull requests 2 Actions Security Insights

Sponsor Edit Pins Unwatch 32 Fork 32 Starred 702

master 4 Branches 53 Tags Go to file Add file Code

arcfide Add preliminary support for parsing with globally free values e387b4c · 3 days ago 5,385 Commits

.github Add some funding information 4 years ago

attic Update 7api.c last year

cmp Add preliminary support for parsing with globally free values 3 days ago

docs Docs update 6 months ago

ime Rework the test suite 6 years ago

rtm Update prim.c last month

About

High-performance, Reliable, and Parallel API

Readme

AGPL-3.0, Unknown licenses found

Activity

Custom properties

702 stars

32 watching

32 forks

Report repository

Everyone should, at least once

TRY
APL

<https://tryapl.org/>

Dyalog APL



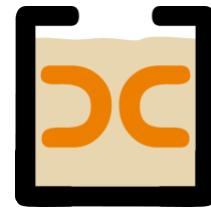
Dyalog IDE
For Windows



Dyalog IDE
For Unix



Tatin
Package Manager



Cider
Project Manager

- ◆ 6 platforms
- ◆ Object Oriented
- ◆ Functional
- ◆ UI Tools
- ◆ Interop-able
- ◆ Multilingual

Getting started

- ◆ Chat in [the APL Orchard](#)
- ◆ Ask a question on [Stack Overflow](#) (apl, dyalog)
- ◆ Use the [r/apljk](#) subreddit
- ◆ [Learning APL](#) by Stefan Kruger
- ◆ Checkout on YouTube:
 - ◆ [@DyalogLtd](#)
 - ◆ [@DyalogUserMeetings](#)

So, Dyalog APL

- ◆ Symbolic concise syntax
- ◆ Array Oriented Programming language
- ◆ Object oriented and functional programming
- ◆ High performance
- ◆ Cross platform

Notation as a Tool of Thought

A language that doesn't affect the way you think about programming is not worth knowing. —*Alan Perlis*

Thank you!