1. hodina: obecný úvod, proměnné, práce s poli

- kdo je odkud, zkušenosti s programováním, k čemu by IDL používali
- Interactive Data Language, interpreted language
- knihovna matematických, statistických a signal&image processing funkcí
- k dispozici interaktivní pomůcky, ale v principu programovací jazyk
- puštění IDL, stručný popis uživatelského prostředí, ukázka běhu na Linuxové konzoli, zmínka o „vychytávkách“ jako hromadné komentáře (+ značení komentářů, více příkazů na řádek, příkaz pokračující na dalším řádku), ukazování uzavírací závorky, prosvěcování stejných příkazů, sbalování celých procedur
- „demo“
- case insensitive (and space insensitive to an extent) – jednoduchý příklad s „print“
- uchovávání posledních příkazů (šipka nahoru)

- vsuvka: „journal“

- 3 základní typy proměnných: skalár, pole (1-8 dimenzí), struktura
- 12 základních datových typů (7 celočíselných, 2 reálné, 2 komplexní, 1 řetězec)

Type	# Bytes	Range	Declare Scalar Syntax	Array	Convert To
Byte	1	0-255	a=2B	bytarr	byte
Integer	2	±2^15-1	b=2 or b=2S	intarr	fix
Unsigned Integer	2	0-2^16-1	c=2U	uintarr	uint
Long	4	±2^31-1	d=2L	lonarr	long
Unsigned Long	4	0-2^32-1	e=2UL	ulonarr	ulong
64-bit Long	8	±2^63-1	f=2LL	lon64arr	long64
64-bit Unsigned Long	8	0-2^64-1	g=2ULL	ulon64arr	ulong64
Floating-Point	4	±10^38	h=2.0	fltarr	float
Double-Precision	8	±10^308	i=2.0D	dblarr	double
Complex	8		j=complex(2.0,2.0)	complexarr	complex
Double-Precision Complex	16		k=dcomplex(2.0D,2.0D)	dcomplexarr	dcomplex
String	n/a		l='hello'	strarr	string

- není nutné proměnné deklarovat, zavedou se při prvním použití
- typ daný použitím (ať už přiřazenou hodnotou nebo z výpočtu plynoucí)
- ukázky pomocí „help“: přiřazení hodnoty, součet dvou různých typů, podíl dvou celočíselných
- použití funkcí na převod mezi typy (posl. sloupeček tabulky)
- pole jsou indexována od nuly, můžeme použít kulatou i hranatou, hranatá „lepší“
- vytvoření pole 1. možnost: pomocí předposledního sloupce tabulky (alternativně s „/nozero“)
- vytvoření pole 2. možnost: „make_array“, a=make_array(3, 4, /integer, value=5)
- vytvoření pole 3. možnost: „replicate“
- vytvoření pole 4. možnost: „indgen“, „findgen“, atd.
- „indgen“ vícerozměrného pole, přístup k elementům pomocí jednotlivých indexů vs. jednoho
- „array_indices“; příklad: ind = array_indices(a,20)
- přístup ke skupině indexů (2:5, 2:*)
- operace s poli jako s celkem: pole * číslo, pole * pole (+ poznámka mocnění přes stříšku)

- operátor „mod“
- poznámka o povoleném „C++ zápise“: ++, += (př 1: a=1 & b=a++ vs. a=1 & b=++a)
- přidání prvku do pole (alternativně: přidání řádku do 2d-pole)
- komplexní čísla: complex, real_part, imaginary, conj
- volání funkcí na celá pole (příklad s „abs“)
- poznámka o hodnotách „nan” a funkci „finite“
- „n_elements”, „size” (počet dimenzí, jednotlivé dimenze, kód typu, počet prvků)
- „total“, „max“, „min“ (+ jak určit index maxima/minima, + /nan switch), „mean“, „median“, „stddev“
- „sort“

- vsuvka: „save“, „restore“, „.reset“

uniq

- `b = a[uniq(a, sort(a))]`

reform

reverse

```
; Create an array:
A = [[0,1,2],[3,4,5],[6,7,8]]
; Print the array:
PRINT, 'Original Array:'
PRINT, A

; Reverse the columns of A.
PRINT, 'Reversed Columns:'
PRINT, REVERSE(A)

; Reverse the rows of A:
PRINT, 'Reversed Rows:'
PRINT, REVERSE(A, 2)
```

shift

- `a = indgen(10) & print,shift(a,1)`
- `a = indgen(5,5) & print,a & print,shift(a,1)`
- `print,shift(a,1,0)`
- `print,shift(a,0,1)`

rotate

transpose

- `a = indgen(2,3,4)`
- `b = transpose(a) & c = transpose(a,[1,2,0])`

invert, determ

#,

```
; 3-column by 2-row array:
a = [ [0, 1, 2],[3, 4, 5] ]
; 2-column by 3-row array:
b = [ [0, 1],[2, 3],[4, 5] ]
print, a # b
print, a ## b
```

- relační operátory: „eq“, „ne“, „ge“, „gt“, „le“, „lt“
- logické operátory: „&&”, „||”, „~”
- bitové operátory: „and“, „or“, „not“, „xor“

<, >

where

Příklady:

- spočtěte 0.75 kvantil zadaného pole
- najděte prvek pole nejbližší zadanému číslu
- vypočtěte signum zadaného pole