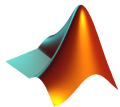


GNU Octave a Matlab: Řetězce

Programování F1400 + F1400a

doc. RNDr. Petr Mikulík, Ph.D.

podzimní semestr 2020



```
s = 'Hello world! Hello people! Hello you!';

% najdi vsechny vyskyty podretezce
strfind(s, 'Hello') % presny podretezec
regexp(s, 'o.[lp]') % regularni vyraz

% rozdel retezec na dve casti
[a,b] = strtok(s, ' ')

% rozdel na vsechny casti
v = strsplit(s, '!'),          length(v), v{1}, v{2}
w = regexp(s, 'o.l', 'split'), length(w), w{1}, w{2}
```

Vyhledávání a nahrazování v polích či maticích

- Najdi čísla dle nějaké podmínky. Logické operátory, pole či matice, příkaz `find`.
- Vygenerujeme řadu prvočísel, najdeme ta v rozsahu 10–20, a zjistíme jejich počet.

```
octave> a = primes(40)           ... vygeneruj prvočísla do 40
```

```
a =  
    2    3    5    7   11   13   17   19   23   29   31   37
```

```
octave> (a>=10 & a<=20)         ... která jsou v zadaném rozsahu od 10 do 20?
```

```
ans =  
    0    0    0    0    1    1    1    1    0    0    0    0
```

```
octave> a(a>=10 & a<=20)       ... najdi (vypiš) je
```

```
ans =  
    11    13    17    19
```

```
octave> find(a>=10 & a<=20)    ... nalezení jejich indexů v poli a
```

```
ans =  
     5     6     7     8
```

```
octave> sum(a>=10 & a<=20)     ... kolik jich je? sečtením jedniček
```

```
ans = 4
```

```
octave> length(find(a>=10 & a<=20)) ... kolik jich je? délka pole s nalezenými
```

```
ans = 4
```

Znaky (characters, chars)

- **Znak:** 1 bajt, tedy celé číslo bez znaménka, tedy v rozsahu [0; 255].
- Dříve pouze 7 bitů z 8 neslo znak, osmý bit byl „paritní“.
- Který znak dané číslo reprezentuje (číslo, písmeno, interpunkce, odřádkování a další tisknutelné a netisknutelné znaky): současný standard pro [0; 127] udává tabulka **ASCII** (American Standard Code for Information Interchange), viz následující strana.
- **Národní znaky (např. akcenty, azbuka)** – pozice 128 až 255 dle národních kódování (např. standardy ISO Latin 1, ISO Latin 2, další „code pages“ např. DOS CP 852 nebo Windows CP 1250), avšak nelze umístit ani všechny evropské znaky.
- Proto se nyní používá standardizované vícebajtové kódování podle normy **Unicode** (konkrétně kódování UTF-8, UTF-16, UTF-32), viz v \LaTeX u příkaz `inputenc{utf8x}`, pro programování řetězců je ale práce s Unicode obtížnější než s řetězci tvořenými pouze jednobajtovými znaky.

Znaky (characters) a ASCII tabulka (7 bitů)

Dec	Hex	Oct	Char	Description	C	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	000	NUL	null		32	20	040	Space	64	40	100	@	96	60	140	'
1	1	001	SOH	start of heading		33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX	start of text		34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX	end of text		35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT	end of transmission		36	24	044	\$	68	44	104	D	100	64	144	d
5	5	005	ENQ	enquiry		37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK	acknowledge		38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL	bell	\a	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS	backspace	\b	40	28	050	(72	48	110	H	104	68	150	h
9	9	011	TAB	horizontal tab	\t	41	29	051)	73	49	111	I	105	69	151	i
10	A	012	LF	line feed, new line	\n	42	2A	052	*	74	4A	112	J	106	6A	152	j
11	B	013	VT	vertical tab	\v	43	2B	053	+	75	4B	113	K	107	6B	153	k
12	C	014	FF	form feed, new page		44	2C	054	,	76	4C	114	L	108	6C	154	l
13	D	015	CR	carriage return	\n	45	2D	055	-	77	4D	115	M	109	6D	155	m
14	E	016	SO	shift out		46	2E	056	.	78	4E	116	N	110	6E	156	n
15	F	017	SI	shift in		47	2F	057	/	79	4F	117	O	111	6F	157	o
16	10	020	DLE	data link escape		48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	C1	device control 1		49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	C2	device control 2		50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	C3	device control 3		51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	C4	device control 4		52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	negative acknowledge		53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	synchronous idle		54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	end of trans. block		55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	cancel		56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM	end of medium		57	39	071	9	89	59	131	Y	121	79	171	y
26	1A	032	SUB	substitute		58	3A	072	:	90	5A	132	Z	122	7A	172	z
27	1B	033	ESC	escape		59	3B	073	;	91	5B	133	[123	7B	173	{
28	1C	034	FS	file separator		60	3C	074	<	92	5C	134	\	124	7C	174	
29	1D	035	GS	group separator		61	3D	075	=	93	5D	135]	125	7D	175	}
30	1E	036	RS	record separator		62	3E	076	>	94	5E	136	^	126	7E	176	~
31	1F	037	US	unit separator		63	3F	077	?	95	5F	137	_	127	7F	177	DEL

Znaky (characters, chars)

- **Znaky:** použijeme písmena bez diakritiky, pak lze volně přecházet mezi číselným a „obrázkovým“ vyjádřením znaku:

```
octave> c = 66;  
octave> fprintf('Znak s ASCII kodem %i je %c\n', c, c);  
znak s ascii kodem 66 je B
```

- Znaky v C – oblíbené funkce z knihovny `ctype.h` – výběr podmnožin ASCII tabulky:

```
islower(); isupper();  
isspace(); isblank();  
isdigit(); isxdigit();  
isalnum(); isalpha(); isascii();  
iscntrl(); isgraph();  
isprint(); ispunct();  
  
tolower(); toupper(); ... Matlab: lower(), upper()  
toascii();
```

- V C lze míchat reprezentaci znaků číslem a syntaxí s apostrofy (např. 'B'), zatímco v GNU Octave/Matlab se k tomu používají řetězce.
- Funkce z `ctype.h` jsou v GNU Octave dostupné pro řetězce. V Matlabu jsou dostupné jen některé z nich a to ještě pod jinými názvy :-)

- **Řetězec** – posloupnost znaků, tj. pole či vektor znaků.
- Jazyk C a Unixová implementace řetězců: řetězec je posloupnost znaků ukončená číslem nula. Céčko: funkce pro práci s řetězci jsou v knihovně `string.h`.
- Většina jazyků: řetězce se zadávají v **uvozovkách**.
Matlab a GNU Octave: řetězce se zadávají v **apostrofech** (ale GNU Octave akceptuje i uvozovky).

- **Naplnění a vypsání řetězce:**

```
octave> s = 'Hello world'  
s = Hello world
```

```
octave> fprintf('Pozdrav je: %s.\n', s);  
Pozdrav je: Hello world.
```

```
octave> length(s)          ... jednoznačná délka, pokud nejsou nabodenička  
ans = 11
```

- **Naplnění řetězce formátováním proměnných** – příkaz `sprintf()`:

```
m = 2;  
s = sprintf('Cislo pi krat %i se rovna %.16g', m, m*pi)
```

```
=> s = Cislo pi krat 2 se rovna 6.283185307179586
```

Řetězce – přístup k prvkům

- **Řetězec** – posloupnost znaků, tj. pole či vektor znaků – přístup k prvkům (znakům) je tedy stejný jako u číselných vektorů.
- **Přístup k jednotlivým znakům** řetězce – příklad:

```
octave> s = 'Hello.'; s(1)='X', s(end)='!'  
s = Hello.  
s = Xello.  
s = Xello!
```

- **Příklad – výpis všech znaků řetězce:**

```
s = 'Hello world!'  
fprintf('Řetězec "%s" má %i znaku.\n', s, length(s));  
for k=1:length(s)  
    fprintf('Znak číslo %2i je: %3i ... %c\n', k, s(k), s(k));  
end
```

```
=>  
Znak číslo 1 je: 72 ... H  
Znak číslo 2 je: 101 ... e  
Znak číslo 3 je: 108 ... l  
Znak číslo 4 je: 108 ... l  
Znak číslo 5 je: 111 ... o  
...
```

Základní funkce s řetězci

- **Změna malých a velkých písmen**; mezery a číslice: **vyhledání**

```
octave> s = 'Hello World 123'
```

```
s = Hello World 123
```

```
octave> upper(s), lower(s)
```

```
ans = HELLO WORLD 123
```

```
ans = hello world 123
```

```
octave> islower(s), isupper(s)
```

```
ans =
```

```
0 1 1 1 1 0 0 1 1 1 1 0 0 0 0
```

```
ans =
```

```
1 0 0 0 0 0 1 0 0 0 0 0 0 0 0
```

```
octave> find(islower(s)), find(isupper(s))
```

```
ans =
```

```
2 3 4 5 8 9 10 11
```

```
ans =
```

```
1 7
```

```
octave> isspace(s), isdigit(s)
```

```
ans =
```

```
0 0 0 0 0 1 0 0 0 0 0 1 0 0 0
```

```
ans =
```

```
0 0 0 0 0 0 0 0 0 0 0 1 1 1
```


Manipulace se znaky v řetězcích – cyklem

- **Manipulace cyklem** – konverze řetězce na malá písmena (různé verze, stejný výsledek; vzdálenost písmen a a A je 32):

```
s = 'Hello World 123';
for k=1:length(s)
    if (s(k)>='A' && s(k)<='Z') s(k) = s(k) + 32; end
    % nebo: if (s(k)>='A' && s(k)<='Z') s(k) = lower(s(k)); end
    % nebo: if (s(k)>='A' && s(k)<='Z') s(k) = tolower(s(k)); end
    % nebo: if (isupper(s(k))) s(k) = tolower(s(k)); end
end
s
```

=> s = hello world 123

- **Manipulace cyklem** – výměna malých písmen za velká a malých za velká:

```
s = 'Hello World 123';
for k=1:length(s)
    if (islower(s(k)))           % nebo: (s(k)>='a' && s(k)<='z')
        s(k) = upper(s(k));     % nebo: s(k) = s(k) - 32
    elseif (isupper(s(k)))      % nebo: (s(k)>='A' && s(k)<='Z')
        s(k) = lower(s(k));     % nebo: s(k) = s(k) + 32
    end
end
s
```

=> s = hELLO WORLD 123

Manipulace se znaky v řetězcích – vektorově

- **Vektorová verze** – výměna mezer za podtržítka:

```
octave> s = 'Hello World 123';  
octave> ss=s; ss(isspace(s)) = '_'  
ss = Hello_World_123
```

- **Vektorová verze** – smazání mezer:

```
octave> s = 'Hello World 123';  
octave> ss=s; ss(isspace(s)) = []  
ss = HelloWorld123
```

- **Vektorová verze** – konverze řetězce na malá písmena (dvě verze):

```
lower(s)  
s(s>='A' & s<='Z') = s(s>='A' & s<='Z') + 32
```

- **Vektorová verze** – výměna malých písmen za velká a malých za velká:

```
s = 'Hello World 123';  
ss = s;  
s(ss>='A' & ss<='Z') = s(ss>='A' & ss<='Z') + 32  
s(ss>='a' & ss<='z') = s(ss>='a' & ss<='z') - 32
```

```
=> s = hELLO WORLD 123
```

- **Setřídění** písmen v řetězci, a to včetně výpisu, na které pozici původního řetězce se utříděné znaky nacházely – příkaz `sort()`:

```
s = 'Hello World 123';  
octave> sort(s)  
ans = 123HWdellloor
```

```
octave> [ss,ind] = sort(s)  
ss = 123HWdellloor  
ind =  
     6     12     13     14     15     1     7     11     2     3     4     10     5     8     9
```

- **Načtení čísla a řetězce z klávesnice:** – příkaz `input()`:

```
octave> x = input('Zadej cislo: ')  
Zadej cislo: 123.45  
x = 123.45
```

```
octave> s1 = input('Zadej řetězec: ', 's')  
Zadej řetězec: Ahoj nazdar!  
s1 = Ahoj nazdar!
```

- **Vyhledání podřetězců:**

```
octave> s = 'Hello world! Hello people! Hello you!';
octave> strfind(s, 'Hello')
ans = 1    14    28
octave> index(s, 'Hello')
ans = 1
```

- **Rozřezávání řetězců (tokens):**

```
octave> s = 'Hello world! Hello people! Hello you!';
octave> [a,b] = strtok(s, ' ')
a = Hello
b = world! Hello people! Hello you!
octave> j = index(b, '!'), d = b(j+2:end)
j = 7
d = Hello people! Hello you!
```

- **Rozřezávání řetězců (split):**

```
octave> s = 'Hello world! Hello people! Hello you!';
octave> z = strsplit(s, '!'), length(z), z{2}
z = {
    [1,1] = Hello world
    [1,2] = Hello people
    [1,3] = Hello you
    [1,4] =
}
ans = 4
ans = Hello people
```

- Porovnávání řetězců – příkaz `strcmp()`:

```
octave> a = 'ahoj', b = 'nazdar', c = 'ahoj', d = 'AHOJ'  
octave> strcmp(a,b), strcmp(a,c), strcmp(a,d)  
ans = 0          ... řetězce jsou různé  
ans = 1          ... řetězce jsou stejné  
ans = 0          ... řetězce jsou různé
```

- Porovnávání řetězců bez ohledu na velikost písmen – příkaz `strncmpi()`:

```
octave> a = 'ahoj', b = 'nazdar', c = 'ahoj', d = 'AHOJ'  
octave> strncmpi(a,b), strncmpi(a,c), strncmpi(a,d)  
ans = 0          ... řetězce jsou různé  
ans = 1          ... řetězce jsou stejné  
ans = 1          ... stejné, velká a malá písmena ignorována
```

- *Poznámka: v jazyce C dává funkce `strcmp()` tři hodnoty: 0 pro případ shody, a ± 1 podle toho, který řetězec je abecedně dříve.*

Řetězce se znaky s diakritikou

- České a další ne-anglické texty mohou obsahovat znaky s diakritikou např. „žlutý kůň“. Ty musí být reprezentovány 8bitovým rozšířením 7bitového ASCII nebo vícebajtovou sadou Unicode. Co vzít za délku takového řetězce? Octave pracuje v UTF-8 a vypíše počet bajtů, Matlab pracuje v UTF-16 a vypíše počet znaků:

```
s='Příliš žluťoučký kůň úpěl ďábelské ódy.', length(s)
```

- Řešení pro zpracování českých textů: převedte řetězec do jednobajtového kódování: `unicode2native(s, encoding)`, kde `encoding` může být `ascii`, `latin2`, `cp1250`, `cp850`, `cp852`, apod.

(Poznámka: za `encoding` můžete dosadit i `utf-16` a zpracovávat vektor po dvojicích čísel.)

```
% s='Příliš žluťoučký kůň úpěl ďábelské ódy.'
```

```
s='kůň'
```

```
z=unicode2native(s, 'ascii')
```

```
length(s), length(z)
```

```
z(1)='x', zz=sprintf('%c', z)
```

```
z=unicode2native(s, 'latin2')
```

```
length(s), length(z)
```

```
z(1)='x'
```

```
z2=native2unicode(z, 'latin2')
```

```
zz=sprintf('%c', z2)
```

- **Regulární výrazy** – **regular expressions**, česky též **výrazy se zástupnými znaky** – umožňují hledat více (podobných) řetězců současně.
- Tečka odpovídá libovolnému znaku: $A.B \rightarrow A@B, AAB, ABB, AzB, A0B, \dots$
Výčet znaků v hraných závorkách: $A[xyz]B \rightarrow AxzB, AyB, AzB$
Dále – pro posloupnost, * pro opakování, ? a {...} pro počet opakování, aj.
- **Příklady použití:**

```
octave> s = 'Hello world! Hello people!'
octave> regexp(s, 'o') ... funguje stejně jako strfind()
=> ans = 5 8 18 22
octave> regexp(s, 'o[rp]') ... hledá všechna o následovaná r či p
=> ans = 8 22
octave> regexp(s, 'o.l') ... hledá o, pak jakýkoliv znak, pak l
=> ans = 8 22
octave> regexp(s, 'o*p'), regexp(s, 'o.*p') ... povoleno vícenásobné opakování
=> ans = 20 22 ans = 5
octave> regexp(s, 'o.l', 'split') ... rozřezání řetězce na části
=> ans = {
    [1,1] = Hello w
    [1,2] = d! Hello pe
    [1,3] = e!
}
```