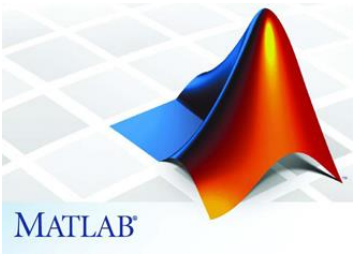
TUTORIÁL1 - ZÁKLADNÁ CHARAKTERISTIKA SIMULAČNÉHO JAZYKA MATLAB

NÁPLŇ

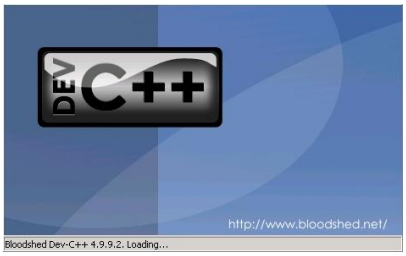
1. POROVNANIE SIMULAČNÉHO JAZYKA MATLAB A PROCEDURÁLNEHO JAZYKA C
2. DÁTOVÉ TYPY JAZYKA MATLAB
3. ZÁKLADNÉ OPERÁCIE S ÚDAJOVÝM TYPOM MATICA
4. RIADIACE ŠTRUKTÚRY JAZYKA MATLAB
5. PRÍKLADY NA SAMOSTATNÉ RIEŠENIE

T1 – 1. POROVNANIE SIMULAČNÉHO JAZYKA MATLAB A PROCEDURÁLNEHO JAZYKA C

S programovacím jazykom C ste sa už oboznámili v 1. ročníku. Na začiatok tejto časti si porovnáme programovací jazyk C a programové prostredie Matlab, s ktorým budeme pracovať na predmete Simulačné systémy v Hospodárskej informatike.



VS



MATLAB (MATrix LABoratory) je vysokoúrovňový jazyk a interaktívne prostredie pre numerické výpočty, vizualizáciu a programovanie. Pomocou simulačného jazyka Matlab, môžete analyzovať dáta, vyvinúť algoritmy, a vytvárať modely aplikácií. Už samotný názov nám prezrádza, že bol navrhnutý predovšetkým pre prácu s maticami (matrix) a vektormi. Simulačný jazyk ďalej umožňuje prácu s rôznymi grafickými nástrojmi.

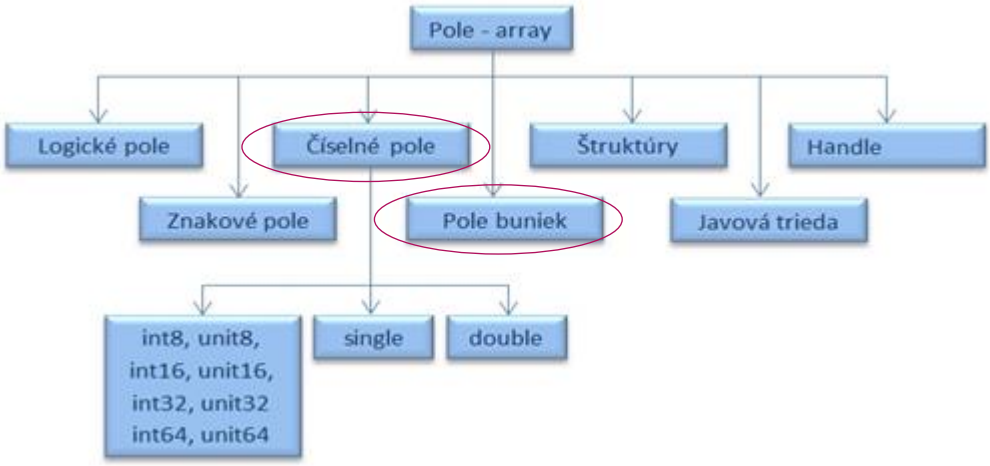
Otvorená architektúra Matlabu viedla k vzniku knižníc funkcií nazvaných toolboxmi. Toolboxy predstavujú jednou z výhod jazyka Matlab, rozširujú použitie programu v príslušných vedných disciplínach.

Jazyk C problémovo orientovaný programovací jazyk, ktorý bol pôvodne vyvinutý pre operačný systém UNIX a až neskôr bol implementovaný aj do iných operačných systémov a stal sa jedným z najpoužívanejších. Jeho hlavnou výhodou je jednoduchosť a nezávislosť na počítači. Umožňuje vytvárať rozsiahle a výkonné programy.

	Matlab	C
Druh jazyka	Skriptovací / simulačný	Problémovo orientovaný
Deklarácia dát	Nie je potrebná	Potrebná
Reprezentácia dát	Matica, vektor	Podľa deklarácie
Indexovanie od	1	0

T1 – 2. DÁTOVÉ TYPY JAZYKA MATLAB

Dátový typ je určenie množiny hodnôt, ktoré môže daná premenná nadobúdať. Matlab obsahuje 15 základných dátových typov a každý z nich môže byť zapísaný vo forme matice alebo poľa. Všetky základné typy sú znázornené v nasledujúcom obrázku.



OBR. 1 - DÁTOVÉ TYPY SIMULAČNÉHO JAZYKA MATLAB

V Tutoriály1 sa budú používať označené dátové typy.

Číselné premenné

Číselné premenné v Matlabe môžu byť zapísané ako znamienkové (signed), alebo neznamienkové (unsigned) ako celé čísla (integer) alebo ako desatinné čísla a to buď s jednoduchou (single) alebo dvojitou (double) presnosťou.

Pri celých číslach Matlab podporuje 8-, 16-, 32-, a 64bitový spôsob znamienkového/ neznamienkového. Rozsahy jednotlivých typov celých čísel si môžete pozrieť v nasledujúcej tabuľke.

Označenie	popis	rozsah
int8	znamienkový 8bitový integer	-2^7 – 2^7 -1
int16	znamienkový 16bitový integer	-2^{15} – 2^{15} -1
int32	znamienkový 32bitový integer	-2^{31} – 2^{31} -1
int64	znamienkový 64bitový integer	-2^{63} – 2^{63} -1
uint8	neznamienkový 8bitový integer	0 – 2^8 -1
uint16	neznamienkový 16bitový integer	0 – 2^{16}-1
uint32	neznamienkový 32bitový integer	0 – 2^{32}-1
uint64	neznamienkový 64bitový integer	0 – 2^{64}-1

Pri nezadaní konkrétneho typu číselnej premennej Matlab ukladá všetky čísla ako desatinné čísla s dvojitou presnosťou. Pokiaľ chceme desatinné číslo uložiť len s jednoduchou presnosťou musíme to spraviť pri jeho zavádzaní. Na obrázku je

znázornený rôzny číselný zápis. Pri výpise „whos“ môžeme vidieť typ premennej a jej veľkosť v bytoch.

```
>> a=uint8(136);
>> b=int32(2165);
>> c=5.214;
>> d=single(6.1247);
>> whos a b c d
  Name        Size              Bytes  Class     Attributes

  a           1x1                   1  uint8
  b           1x1                   4  int32
  c           1x1                   8  double
  d           1x1                   4  single
```

Formátovaný vstup a výstup

Matlab ponúka dve možnosti pre vstup a to funkcie ***sscanf*** (čítanie dát z reťazca, po preformátovaní ho uloží do premennej) a ***fcanf*** (číta a formátuje dáta z textového súboru.

Pre formátovaný výstup sa používajú funkcie ***sprintf*** (vzťahuje sa na všetky prvky poľa, naformátuje ich a vráti výsledok) a ***fprintf*** (zapisuje dáta do textového súboru alebo ich vypíše na obrazovku).

Pri používaní funkcií formátového vstupu a výstupu je potrebné použiť konverzné znaky. V nasledujúcej tabuľke sú uvedené najpoužívanejšie.

Celé číslo	signed	%d	Desiatkové číslo typu signed
		%ld	Desiatkové číslo typu signed
	unsigned	%u	Desiatkové číslo typu unsigned
		%lu	Desiatkové číslo typu unsigned long
		%o	Osmičkové číslo
		%x	Hexadecimetrálne číslo s malými písmenami
		%X	Hexadecimetrálne číslo s veľkými písmenami
Reálne číslo		%f	
Znaky		%c	Jeden znak
		%s	Reťazec

Príklad:

```
premenna=sscanf(x,'%o');
fprintf('Výsledok je: %f',premenna);
```

T1 – 3. ZÁKLADNÉ OPERÁCIE S ÚDAJOVÝM TYPOM MATICA

Dáta v Matlabe, ako sme už spomínali, sú reprezentované maticami. Matica je dvojrozmerné / obdĺžnikové pole reálnych alebo komplexných čísel. Matica, ktorá má *m* riadkov a *n* stĺpcov je nazývaná „matica *m* x *n*“. Každý prvok matici je indexovaný dvojitým indexom a to a_{ij}, kde *i* predstavuje poradie riadku a *j* poradie stĺpca, v ktorom sa prvok nachádza. Môžeme použiť aj skalárnu veličinu, avšak v Matlabe je tiež chápaná ako matica 1x1. Maticu môžeme zadať niekoľkými spôsobmi, a to :

- Zadaním po prvkoch (prvky v riadku sú oddelené medzerou/čiarkou, v stĺpcoch + bodkočiarkou)
- Pomocou funkcie – špeciálne matice (eye, ones, rand...)
- Nahraním zo súboru

Základné operátory

Znak	Opis	Znak	Opis
+	Plus	.	Desatinná bodka
-	Mínus	%	Komentár
*	Maticové násobenie	=	Priradenie
.*	Násobenie po prvkoch	==	Zhodnosť
^	Umocnenie	&	Logický AND
. ^	Umocnenie po prvkoch	|	Logický OR
\	Ľavé delenie	~	Logická negácia
/	Pravé delenie		
./	Pravé delenie prvkov		

Operátor „ : “

Často používaný operátor, ktorý sa používa pri tvorbe postupností s konštantným krokom. Jeho syntax je :

```
v = začiatok : krok : koniec
```

Poznámka

- krok“ je možné vynechať a vtedy sa bude chápať ako „1“
- „krok“ môže byť aj záporné číslo, vtedy sa vytvára zostupná postupnosť

Riešený príklad 1 - V príkazovom režime vytvorte vektor v1 s prvkami od 1 po 9 s krokom 2 a vektor v2 ktorého prvky budú zostupnou postupnosťou čísel od 15 po 3 s krokom 3 pomocou dvojbodkovej konvencie.

```
>> v1=1:2:9

v1 =

     1     3     5     7     9
```

```
>> v2=15:-3:3

v2 =

    15    12     9     6     3
```

Funkcia linspace na generovanie vektorov

Funkcia linspace má podobnú funkciu ako predchádzajúci operátor „ : “ s tým rozdielom, že si presne vypočíta krok. Je výhodné použiť túto funkciu v prípadoch, kedy potrebujeme vytvoriť vektor s vopred známym počtom prvkov, ale neznámym krokom. Syntax je nasledovná :

```
v = linspace(začiatok, koniec, počet_prvkov)
```

Riešený príklad 2 - V príkazovom režime vytvorte vektor v3, s počtom prvkov 5, ktoré budú z intervalu <0;π>. Príklad riešte s využitím funkcie linspace.

```
>> v3=linspace(0,pi,5)

v3 =

         0    0.7854    1.5708    2.3562    3.1416
```

Výber prvkov matice a submatica

Ako sme už spomenuli, každý prvok matice je indexovaný v tvare a_{ij}. Pri výbere konkrétneho prvku matice stačí zadať príkaz :

```
Matica(číslo_riadku, číslo_stĺpca)
```

Pri výbere časti matice – submatice nám pomáha operácia „ **:** “. je to podobne ao pri výbere prvku s tým rozdielom, že namiesto „číslo_riadku“ sa dá rozmedzie riadkov, ktoré chceme vybrať. A rovnako aj namiesto „číslo_stĺpca“ dáme rozmedzie stĺpcov, ktoré chceme v submatici mať. Príkaz:

```
A=M(x:y,x:y)
```

Kde ***x*** – poradie riadka/stĺpca, od ktorého chceme vytvoriť submaticu
y - poradie riadka/stĺpca, pri ktorom chceme ukončiť výber

Riešený príklad 3 - Vytvorte maticu M

$$M = \begin{matrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{matrix}$$

a) Hodnotu prvku 3.riadku a 2. stĺpca vložte do premennej *a*
b) Vytvorte submaticu *B* matice *M*, ktorá bude tvorená prvkami 1.-3. riadok, 2.-4. stĺpec

```
>> M=[1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]

M =

     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16

>> a=M(3,2)

a =

    10

>> B=M(1:2,2:end)

B =

     2     3     4
     6     7     8
```

Príklady na precvičenie – príklady riešte v príkazovom riadku

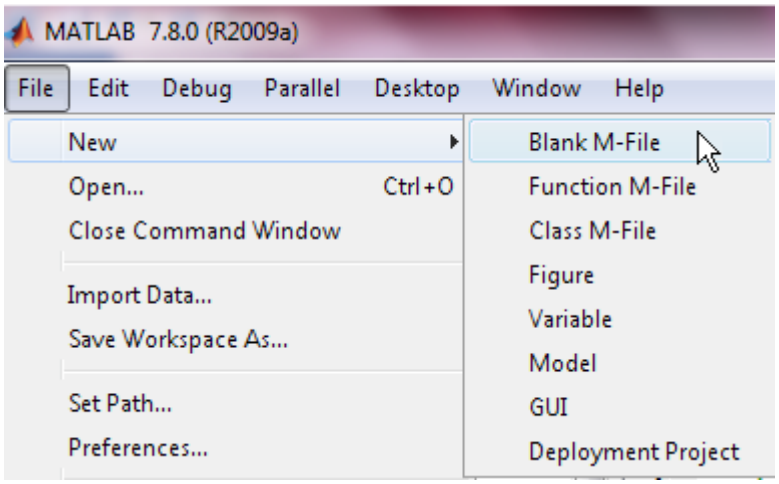
1. Do premennej *x* vložte 10 čísel v rozmedzí čísel od 1 do 5 pomocou funkcie linspace.
2. Do premennej *y* vložte logaritmy hodnôt nachádzajúcich sa vo vektore *x*.
3. Do premennej *z* vložte hodnoty premennej *x* umocnené na druhú.
4. Vytvorte vektor *s*, ktorého počet prvkov bude 8, prvky budú usporiadané zostupne a ich hodnoty budú v rozmedzí od 23 po 9 pomocou dvojbodkovej konvencie.
5. Vytvorte maticu M, ktorá bude mať v prvom riadku čísla od 3 po7, v druhom riadku bude tak isto päť prvkov ktoré budú v rozmedzí 10 – 23 a v treťom riadku nech sú čísla v zostupnom poradí od 19 do 5 s rovnomerným rozdelením.
    a. Vytvorte submaticu A matice *M*, ktorá bude obsahovať prvky 2-3riadku a 3-4stĺpca
    b. Vytvorte submaticu B matice *M*, ktorá bude obsahovať prvky 3-koncového riadku a 1-2 stĺpca, tak aby prvky boli v riadku usporiadané v opačnom poradí oproti matici *M*
6. Vytvorte ľubovoľnú maticu K s rozmermi 8x8. Vytvorte ľubovoľnú submaticu *C* matice *M*. Z matice *C* vyberte prvok nachádzajúci sa na pozícii [1,3] a vložte ho na pozíciu [1,4]. Zobrazte maticu *C* pred aj po presune prvku [1,3]
7. Vytvorte premennú *t* , ktorá bude obsahovať 20 hodnôt funkcie *cos(x)* pričom x bude v rozmedzí od <-10:10>. Hodnoty ktoré získate vypíšte na obrazovku .

T1 - 4. RIADIACE ŠTRUKTÚRY JAZYKA MATLAB

Riadiace štruktúry sú nevyhnutnou súčasťou programovacieho jazyka. Umožňujú riadiť chod programu, jeho vetvenie či opakovanie časti kódu.

Pri tvorbe zložitejších programov, v ktorých sa využívajú riadiace štruktúry je nevyhnutné vytvárať skripty a kód písať v nich.

Nový skrypt vytvoríme nasledovne:

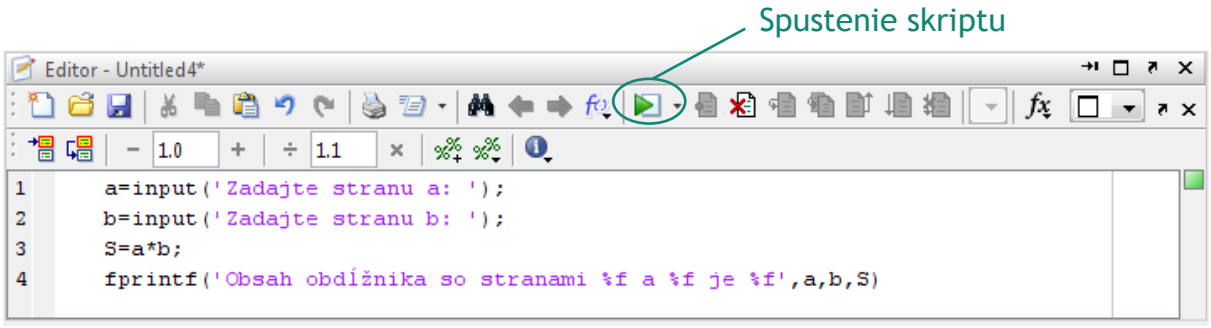


Skripty sú postupnosti príkazov uložené do súborov. Sú považované za najjednoduchšie M-fily z toho dôvodu, že nevyžadujú vstupné a výstupné argumenty. Pracujú v spoločnom prostredí programu s globálnymi premennými t.j. pracujú s premennými, ktoré sú definované vo Workspace, ale môžu vytvárať aj vlastné premenné. Hodnoty priradené premenným zostávajú v pamäti aj po vykonaní skriptu.

V čase písania skriptu sa jeho príkazy nevykonávajú. Na ich vykonanie stačí napísať do príkazového okna MATLABu názov skriptu bez koncovky a odoslať na spracovanie.

Niekedy je vhodné nevykonávať určité časti príkazov v M-súbore. Na túto činnosť slúžia komentáre. Komentár začína znakom percenta a končí na konci aktuálneho riadku. Pre lepšiu identifikáciu sa komentáre zvýrazňujú štandardne zelenou farbou.

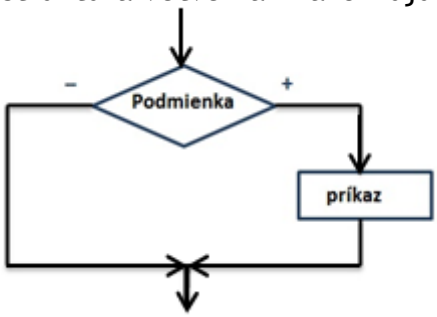
Riešený príklad 4 - Vytvorte postupnosť príkazov - skript, v ktorý vyzve užívateľa na zadanie strán obdĺžnika a následne vypočíta jeho obsah.



Podmienený príkaz if

Príkaz ***if*** sa používa na vetvenie, v prípade kedy je potrebné vykonať dané príkazy len za predpokladu splnenia určitej podmienke. Príkaz vetvenia umožňuje vyhodnotiť podmienku a na základe tohto vyhodnotenia vykonať respektíve nevykonať príkazy. Jeho základná syntax a štruktúra vetvenia znázorňujúca činnosť príkazu sú nasledovné :

```
if podmienka
    prikazy
end
```

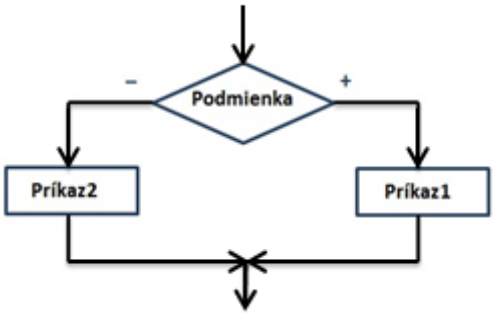


Riešený príklad 5 - Ak x < 10, potom vypíšte hlásenie „ číslo x je menšie ako 10“

```
if x<10
    disp('číslo x je menšie ako 10')
end
```

V prípade potreby vykonania iných príkazov pri nesplnení podmienky sa používa príkaz ***if*** - ***else***. Syntax a štruktúra vetvenia znázorňujúca činnosť :

```
if podmienka
    prikaz1
else prikaz2
end
```

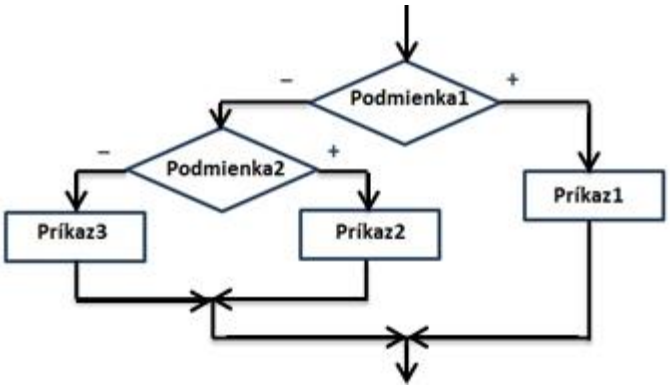


Riešený príklad 6 - Ak x < 10 potom vypíšte hlásenie „ číslo x je menšie ako 10“, v opačnom prípade vypíšte hlásenie „číslo x je väčšie alebo rovné číslu 10“.

```
if x<10
    disp('číslo x je menšie ako 10')
else disp('číslo x je väčšie alebo rovné číslu 10')
end
```

Ďalšou modifikáciou príkazu if je príkaz ***if*** - ***elseif*** - ***else***. Syntax a štruktúra vetvenia sú nasledovné:

```
if podmienka1
    prikaz1
elseif podmienka2
    prikaz2
else
    prikaz3
end
```



Pri splnení podmienky `podmienka1` sa vykoná príkaz `príkaz1`, a zvyšok príkazu ostane ignorovaný. Pri nesplnení prvej podmienky nasleduje testovanie podmienky `podmienka2`, pri jej kladnom vyhodnotení sa vykoná príkaz `príkaz2`, v opačnom prípade bude vykonaný príkaz `príkaz3`.

Riešený príklad 7 - Porovnajte číslo x s číslom 10 a vypíšte príslušné hlásenia.

```
if x<10
    disp('číslo x je menšie ako 10')
elseif x>10
    disp('číslo x je väčšie ako 10')
else disp('číslo x je rovné číslu 10')
end
```

Riešený príklad 8 - Zostavte program na určenie minima a jeho polohy z troch hodnôt zadaných z klávesnice.

```
a=input('Zadajte číslo: ');
b=input('Zadajte číslo: ');
c=input('Zadajte číslo: ');
min=a;
pozm=1;
if min>b
    min=b;
    pozm=2;
end
if min>c
    min=c;
    pozm=3;
end
fprintf('Hodnota minima zo zadaných čísel je %f \n',min);
fprintf('Pozícia minima zo zadaných čísel je %d \n',pozm);
```

Riešený príklad 9 - Vytvorte program na výpočet minimálneho potrebného počtu jázd výťahom. Výťah má obmedzenú nosnosť a sú známe hmotnosti troch cestujúcich, tieto hodnoty zadá užívateľ z klávesnice.

```
h=input('Zadajte nosnosť výťahu: ');
m1=input('Zadajte hmotnosť 1.pasažiera: ');
m2=input('Zadajte hmotnosť 2.pasažiera: ');
m3=input('Zadajte hmotnosť 3.pasažiera: ');
n=3; %maximálny potrebný počet jázd
if m1+m2<=h | m1+m3<=h | m2+m3<=h %overenie-nájde sa dvojica, ktorá by mohla ísť spolu
   n=2; % nutné sú dve jazdy
end
if m1+m2+m3<=h  % overenie, či by mohli ísť všetci spolu
   n=1; % stačí jedna jazda
end
if m1>h | m2>h | m3>h   % overenie, či určitá hmotnosť nepresahuje nosnosť výťahu
   n=inf;   % nie je možné aby išli všetci výťahom
end
fprintf('Potrebný počet jázd výťahom je: %d \n',n)
```

Príklady na precvičenie

1. Vytvorte vektor *v*, ktorý bude obsahovať 10 náhodných čísel z intervalu <-10 ; 10>.
   a. Zistite koľko prvkov z vektora *v* je kladných, koľko záporných a koľko rovných nule.
   b. Vytvorte vektory *k a z*. Do vektora *k* vložte všetky prvky z vektora *v*, ktoré sú kladné. Do vektora *z* vložte všetky prvky z vektora *v*, ktoré sú záporné. Vypíšte na obrazovku vektory *v, k, z*.

Príkaz switch - case

V prípade potreby väčšieho počtu vetiev pri jednej podmienke je príkaz *if*, dosť nepraktický a nahrádza ho príkaz ***switch***, ktorého syntax je nasledovná :

```
switch podmienka
    case {hodnota1}
        prikazy1
    case {hodnota2}
        prikazy2
    otherwise
        prikazy3
end
```

`Podmienka` predstavuje hodnotu, ktorú ideme porovnávať s `hodnota1` a `hodnota2`. Po vyhodnotení zhodnosti s jednou z týchto hodnôt sa vykoná príslušný `príkaz`. V prípade, že sa `vyraz` nebude zhodovať ani s jednou `hodnotou`, vykoná sa príkaz `príkaz3`.

Riešený príklad 10 - Nech číslo a je vstupom Vášho programu. Zistite či zvyšok po delení číslom 7 je párny, nepárny, alebo číslo a je k-násobkom čísla 7.

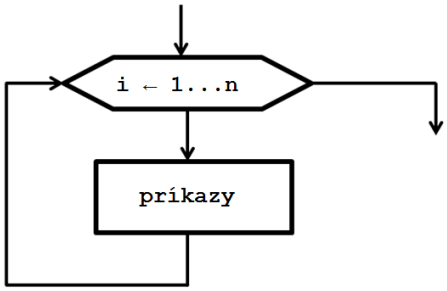
```
a=input('Zadaj celé číslo a : ')
b=mod(a,7)
switch b
    case {1, 3, 5}
        disp('zvyšok po delení číslom 7 je nepárny')
    case {2, 4, 6}
        disp('zvyšok po delení číslom 7 je párny')
    case {0}
        disp('vami zadané číslo je deliteľné číslom 7')
end
```

Cykly

Cykly nám slúžia na opakované vykonávanie príkazu, alebo skupiny príkazov. Cykly môžeme rozdeliť do dvoch skupín a to cykly s pevne daným počtom opakovaní a cykly, pri ktorých nepoznáme počet opakovaní.

Cyklus for

Príkaz ***for*** nám slúži na vykonanie určitých príkazov niekoľko krát. Počet opakovaní určujeme v deklarovaní premennej za príkazom for, v našom prípade „i“. Určíme začiatok pre i od ktorého sa cyklus začne vykonávať, krok ,s ktorým sa bude cyklus vykonávať. V prípade vynechania časti „krok“ sa za krok automaticky považuje číslo 1. Keď i dosiahne hodnotu koniec príkazy prebehnú posledný krát. Vývojový diagram a syntax príkazu:



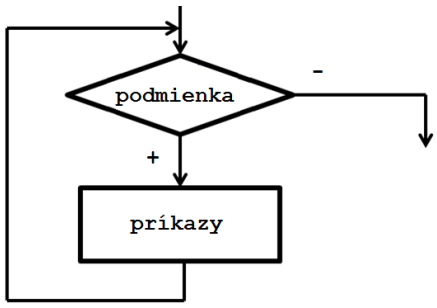

```
for i=zaciatok:krok:koniec
    prikaz
end
```

Riešený príklad 11 – Vytvorte vektor a ktorého prvky budú zodpovedať ich poradiu a dĺžka bude načítaná zo vstupu.

```
n=input('Zadaj dĺžku vektora : ')
for i=1:n
    a(i)=i;
end
disp(a)
```

Cyklus while

Príkaz ***while*** sa využíva väčšinou na vykonanie príkazu, alebo skupiny príkazov s vopred neznámym počtom opakovaní. `Príkaz` sa vykonáva dovtedy, kým je splnená `podmienka`. Syntax príkazu:




```
while podmienka
    prikaz
end
```

Riešený príklad 12 – Je zadaná funkcia f(a)= $\frac{(a^2-3a-2)}{(a-1)^2(a+2)}$. Vytvorte program pre výpočet hodnoty tejto funkcie pre ručne zadaný vstup hodnoty *a*. Overte hodnotu *a* a vytvorte cyklus, ktorý výpočet vykoná až po zadaní vhodnej hodnoty *a* .

```
a=input('Zadajte hodnotu a pre ktorú chcete vykonať výpočet výrazu: ');
while (a==1)| (a==-2)
   fprintf('\nZadali ste hodnotu, pre ktorú výraz nemá zmysel.\n')
   a=input('\nZadajte hodnotu a pre ktorú chcete vykonať výpočet výrazu:');
end
x=(a^2-3*a-2)/((a-1)^2*(a+2));
fprintf('Hodnota výrazu:(a^2-3*a-2)/((a-1)^2*(a+2)) pre zadanú hodnotu a=%d je %f',a,x)
```

Riešený príklad 13 - Vytvorte ľubovoľnú maticu s rozmermi 6x8. Vytvorte cyklus, ktorý vytvorí novú maticu, ktorá bude obsahovať prvky pôvodnej matice nachádzajúce sa na pozícií, ktorej poradové číslo stĺpca aj riadku je párne.

```
M=rand(6,8)
for i=1:6   % cyklus pre prechádzanie riadkov
    for j=1:8   % cyklus pre prechádzanie stĺpcov
        if (rem(i,2)==0 & rem(j,2)==0)
% podmienka: zvyšok po delení čísla i číslom 2 je nula a zároveň zvyšok po
% delení čísla j číslom 2 je nudla
            a=i/2;  % zavedenia nového počítadla riadkov
            b=j/2;  % zavedenie nového počítadla stĺpcov
            A(a,b)=M(i,j);   % vkladanie prvkov z pôvodnej matice do novej
        end
    end
end
A
```

Riešený príklad 14 - Predpokladajme, že do „ideálnej banky“, ktorej ročná úroková miera je 5,5% sme vložili 30 000€. Banka úrok pripočítava na konci každého roka zo sumy, ktorá tam je s tým, že počas roka sa s peniazmi nehýbe. Vypočítajte nárast úroku v nasledujúcich piatich rokoch jednotlivo.

```
BH(1)=30000*(1+0.055);  %výpočet budúcej hodnoty po prvom roku
for i=2:5
    BH(i)=BH(i-1)*(1+0.055);  %výpočet budúcej hodnoty v ďalších rokoch
end
U(1)=BH(1)-30000; %výpočet úroku, kt. pribudol po prvom roku
for i=1:4
    U(i+1)=BH(i+1)-BH(i);   %výpočet úroku, kt. pribudol v ďalších rokoch
end
U
```

Riešený príklad 15 - Vytvorte skript, ktorý vypočíta aritmetický priemer troch najväčších zadaných čísel. Čísla bude zadávať užívateľ z klávesnice a zápis ukončí vložením čísla 0. Ošetrite kód tak, že v prípade, že užívateľ zadal menší počet čísel ako 3 program do aritmetického priemeru vloží „0“.

```
i=1;
a(1)=input('Zadajte číslo: ');
% kým zadané číslo nie je 0 pokračujte v zadávaní
while a(i)~= 0
    i=i+1;
    a(i)=input('Zadajte číslo: ');
end
% zníženie počítadla o 1  aby hodnota premennej i predstavovaa počet
% zadaných čísel bez čísla 0
i=i-1;
% overenie, či užívateI zadal menší počet čísel ako 3
% v opačno prípade sa vykoná vetva "else"
if i<3
    APmax=0;
else
    % načítanie prvých troch hodnôt do maxím
    for j=1:3
        max(j)=a(j);
    end
    % usporiadaie prvých "maxím" zostupne
    if max(1)<max(2)
        p=max(1);
        max(1)=max(2);
        max(2)=p;
    end
    if max(1)<max(3)
        p=max(1);
        max(1)=max(3);
        max(3)=p;
    end
    if max(2)<max(3)
        p=max(2);
        max(2)=max(3);
        max(3)=p;
    end
    % overenie či zvyšné čísla sú väčšie ako čísla v uložené v premenných
    % max(i)a následné nahradenie hodnoty
    for k=4:i
        if max(3)<a(k)
            max(3)=a(k);
            if max(3)>max(2)
                p=max(2);
                max(2)=max(3);
                max(3)=p;
                if max(2)>max(1)
                    p=max(1);
                    max(1)=max(2);
                    max(2)=p;
                end
            end
        end
    end
    APmax=(max(1)+max(2)+max(3))/3;
end
fprintf('Aritmetický priemer troch najväčších zadaných čísel je: %f',APmax)
```

Príklad na precvičenie

1. Vytvorte program na spravovanie bankového účtu. Užívateľ bude zadávať vklady a výbery prostredníctvom kladných a záporných čísel, ukončenie bude vložením hodnoty 0. Zisti počet výverov, počet príjmov, sumu výberov a sumu príjmov.
2. Naplňte maticu rozmerov 4x4 hodnotami zadanými z klávesnice. Využite pritom cykly.
3. Pomocou cyklu vypíšte sumu všetkých riadkov matice a sumu celej matice spolu.

T1 – 5. PRÍKLADY NA SAMOTATNÉ RIEŠENIE

1. Vytvorte pomocou cyklu maticu rozmerov 4x4, ktorá bude obsahovať len párne čísla. Čísla do matice sa budú vkladať z klávesnice. Zadajte podmienku, aby sa do matice dali vložiť len párne čísla.

2. Vytvorte cyklus, ktorý vypíše minimálne hodnoty stĺpcov z ľubovoľnej matice, ktorú zadáte. Následne porovnajte výsledky, ktoré vyhodnotil vás cyklus s výsledkami, ktoré vygeneroval príkaz min().

3. Vytvorte maticu ľubovoľných hodnôt s rozmermi 5x5. Napíšte cyklus na vytvorenie vektora, ktorý bude obsahovať prvky hlavnej diagonálu zadanej matice. Vypočítajte sumu tohto vektora, výsledok vypíšte.

4. Predpokladajme, že do banky, ktorá má ročnú úrokovú mieru 7% vložíme 10 000€.
    a. zistime za koľko rokov budeme mať aspoň 20 000€
    b. ako sa nám bude meniť suma v jednotlivých rokoch

5. Firma ABC s.r.o. si zakúpila počítačovú zostavu, ktorej cena je 4 000€ (zaradenie : dlhodobý hmotný majetok). Majetok bol zaradený do prvej odpisovej skupiny. Zostavte odpisový plán pre rovnomerné a zrýchlené odpisy.

Poznámka :
Úrokovanie
budúca_hodnota = súčasná_hodnota * (1 + počet_rokov * ročná_úroková_miera)