Feltételes elágazás

```
if kifejezes
   utasitasok
end
```

Végrehajtja az utasításokat, ha a kifejezés igaz. Egy kifejezés akkor igaz, ha csak nemnulla értékeket ad vissza.

Példa:

```
x=8;
if x>6
    disp('x nagyobb, mint 6');
end
```

Mivel az x>6 kifejezés értéke most (logikai) 1, ezért az utasítást végrehajtja.

```
x=[4,8];
if x>6
    disp('x nagyobb, mint 6');
end
```

Mivel az x>6 kifejezés értéke most [0, 1] (az x minden elemére elvégezte az összehasonlítást), azaz nem csak nemnulla értékeket adott, ezért az utasítást nem hajtja végre.

```
x=[14,8];
if x>6
    disp('x nagyobb, mint 6');
end
```

Itt a kifejezés értéke [1, 1], végrehajtja az utasítást.

A feltételes elágazás bővíthető (akár több) elseif ággal, illetve egy else ággal is.

```
if kifejezes1
   utasitasok
elseif kifejezes2
   utasitasok
else
   utasitasok
end
```

Példa:

```
x=[4,8];
if x>6
   disp('x minden eleme nagyobb, mint 6');
else
   disp('x valamelyik eleme nem nagyobb, mint 6');
end
```

Példa:

```
x=5;
if x>6
  disp('x nagyobb, mint 6');
elseif x>4
  disp('x nagyobb, mint 4, de legfeljebb 6');
elseif x>2
  disp('x nagyobb, mint 2, de legfeljebb 4');
else
  disp('x legfeljebb 2');
end
```

Itt az első kifejezés ($x > 6$) hamis, ezért megvizsgálja a következő kifejezést ($x > 4$). Mivel ez igaz, így az itteni parancsot végrehajtja, és a további kifejezéseket nem vizsgálja.

Teszteljük az előző kódot különböző x értékekre! (Pl. $x = 3$, $x = 2$, $x = 1$.)

A kifejezés egy összetett logikai kifejezés is lehet. Pl.

```
x=5;
if x>4&x<6
  disp('x nagyobb, mint 4, de kisebb, mint 6');
elseif x<=4
  disp('x legfeljebb 4');
else
  disp('x legalabb 6');
end
```

Itt az első kifejezés akkor igaz, ha $x > 4$ és $x < 6$ is igaz. A kifejezést balról jobbra haladva, rövidzár kiértékeléssel értékeli ki.

While-ciklus

```
while kifejezes
   utasitasok
end
```

Az utasítások végrehajtását addig ismétli, amíg a kifejezés igaz (azaz csak nemnulla értékeket ad vissza). Pl.

```
k=4;
while k>1
    disp(['A k erteke most:', num2str(k)]);
    k=k-1;
end
```

1.lépés: $k = 4$, így $k > 1$ igaz, elvégzi a kiiratást, k-t csökkenti 1-gyel
2.lépés: $k = 3$, így $k > 1$ igaz, elvégzi a kiiratást, k-t csökkenti 1-gyel
3.lépés: $k = 2$, így $k > 1$ igaz, elvégzi a kiiratást, k-t csökkenti 1-gyel
4.lépés: $k = 1$, így $k > 1$ nem igaz, kilép a ciklusból

Példa

Szorozzuk össze 1-től 10-ig a számokat while-ciklussal:

```
k=10; p=10;
while k>1
   k=k-1;
   p=p*k;
end
disp(['p erteke:',num2str(p)])
```

vagy

```
k=1; p=1;
while k<11
   p=p*k;
   k=k+1;
end
disp(['p erteke:',num2str(p)])
```

continue, break

Szorozzuk össze az 5 kivételével 1-től 10-ig a számokat while-ciklussal:

```
k=10; p=10;
while k>1
    k=k-1;
    if k==5
        continue;
    end
    p=p*k;
end
disp(['p erteke:',num2str(p)])
```

continue: a ciklus törzsében lévő további utasításokat átugorva a ciklus elejére lép, és a következő iterációval folytatja.

break: kilép a ciklusból (több, egymásba ágyazott ciklus esetén csak abból, ahol kiadtuk a parancsot). Az utasítások végrehajtását az adott ciklus befejezése utáni első utasítással folytatja.