## 8. Gyakorlat

Függvények:

- A függvény egy, a forráskód többi részétől viszonylag független egység.
- Adott feladatot hajtanak végre.
- Kód újrafelhasználás szempontjából hasznosak. Használatukkal olvashatóbbá tehető a kód, az összetett problémák egyszerűbb részekre bonthatók.
- Általános alakjuk bash-ben:
  ```
  function fuggvenynev { utasítások; }
  ```
- Példa:
  ```
  var=0
  function ertekNoveles {
       ((var++))
  }
  while [ $var –lt 10 ]; do
       ertekNoveles
       echo $var
  done
  ```

Lokális változók:

- A shell scriptekben megkülönböztethetünk globális és lokális változókat. Globális változó a scripten belül, bárhol létrehozható, lokális változó csak és kizárólag függvény törzsén belül.
- Lokális változó létrehozásához a változónév elé ki kell tenni a „local” kulcsszót, ezek csak annak a függvénynek a törzsén belül érhetők el, amelyben létrehoztuk.
- Példa:
  ```
  var=”globalis”
  function f {
       local var=”lokalis”
       echo $var # a lokális változó értéke íródik ki
  }
  echo $var # kiírja, hogy „globalis”
  echo $(f) # kiírja, hogy „lokalis”
  echo $var # kiírja, hogy „globalis”
  ```

Paraméteres függvények:

- Shell scriptekben a függvényeknek átadott paraméterek a scriptnek megadott parancssori paraméterekkel megegyezően kezelendők.
- $0-val függvényen belül is a program neve érhető el.
- $1, $2, …, $9 a függvénynek átadott paraméterek.
- $# a paraméterek száma, $* és $@ az összes paraméter együtt.
- Példa:
  ```
  function hello {
       echo „Hello $1!”
  }
  read –p „Hogy hivnak? ” name
  hello $name
  ```

Függvény a függvényben:

- Lehetőségünk van létrehozni egy függvényt, egy másik függvény törzsén belül.
- A belső függvény akkor jön létre, amikor a külsőre először hivatkozás történik, ezt követően normál függvényként működik.
- A belső függvény, a külső törzsén belül, és azon kívül egyaránt hívható.
- Példa:

```
function kulso {
     echo „Hello”
     function belso {
          echo „World”
     }
}
belso # hiba, belso még nem létezik
kulso # kiírja, hogy „Hello”, létrejön belso
belso # kiírja, hogy „World”
```

Az echo használata függvényeknél:

- Az echo alap funkciója, hogy a paraméterként megadott értékeket a kimenetre írja.
- Függvényeknél az echo, a függvényhívás mikéntjétől függően, a kimenetre írás mellett felhasználható visszatérési érték előállítására is.
- Példa:

```
function f {
     [ $(($1 % 2)) -eq 0 ] && echo P || echo PTLAN
}
f 10 # az eredményt a kimenetre írja
var=$(f 11) # az eredményt a változóban tárolja
```

A return utasítás:

- Csak egész szám lehet a return paramétere.
- A paramétere, mint a függvény visszatérési értéke, a függvény hibakódjának tekinthető.
- Felhasználható például parancslisták létrehozásához, ha a hívott függvény visszatérési értéke normális, azaz 0, folytatódik a végrehajtás, egyébként megszakad (&& esetén).
- Példa:

```
function f {
     return 0
}
f && echo „Normális.”
```

- A függvények ilyen módon előállított visszatérési értéke elérhető a $? kifejezéssel.
- Példa:

```
function f {
     return 153
}
f
echo $?
```

Az eval utasítás:

- Felhasználható arra, hogy egy függvénynek egy változónevet adunk paraméterként, majd ennek a változónak a segítségével adjuk vissza a függvény által elvégzett műveletek eredményét.
- Példa:

```
function f { eval $1=”érték”; }
f var; echo $var
```

- A példában az f függvény törzsén belül, ha elhagynánk az eval utasítást, hibát kapnánk.

Feladat: írjunk scriptet, amely:

- beolvas egy számot, melynek értéke nem lehet, kisebb, mint nulla,
- ha a beolvasott adat hibás, újból kéri a felhasználótól,
- ha a beolvasott szám „n”, akkor kiírja az n. Fibonacci-számot.