Laboratorul 8

1. Structura `for` permite execuţia ciclică a unei liste de comenzi, modificând la fiecare parcurgere a ciclului valoarea unei variabile specificate în `for`; are sintaxa de forma:

```
for variabila [in listvalori]
                  do
                  lista_de_comenzi
                  done
```

Dacă partea opţională lipseşte, *variabila* ia pe rând ca valori argumentele prezente în linia de comandă.

Exemplu. O structură for pentru crearea unui număr oarecare de fişiere.

```
for i do cat >$i; done
```

Se observă lipsa părţii opţionale ca şi că se poate scrie cuvântul rezervat `done`, în aceeaşi linie cu o comandă, dacă se foloseşte separatorul ';'.

Exemplu. Afişarea numelor fişierelor din catalogul curent (cu excepţia celor ascunse).

```
for i in *
do
  echo $i
done
```

Exemplu. Sortarea şi afisarea conţinutului tuturor fişierelor date ca argumente în linia de comandă.

```
for fis in *
do
  sort $fis | more
done
```

Dacă numele scriptului este `sortare` şi se tastează:

```
$ sortare fis1 fis2 fis3
```

se execută comenzile:

```
sort fis1 | more
sort fis2 | more
sort fis3 | more
```

Exemplu. Sortarea şi afişarea tuturor fişierelor din directorul curent, al căror nume contine caracterele `fis.`

```
for fisier in *fis*;do;sort fisier |more; done
```

bash recunoaşte o sintaxă împrumutată din limbajul C :

```
max=LimitaSup
for ((i=1; i <= max ; i++))
do
<secventa>
done
```

Exemplu.

```
max=10
for ((i=1; i <= max ; i++))
  do
  echo -n "$i..."
  done
  echo
```

Când se lansează în execuţie, se afişează.:

```
1...2...3...4...5...6...7...8...9...10...
```

2. Structurile de ciclare `while` şi `until`.

Structura `while` are sintaxa:

```
while listcomanda_1
do
  listcomanda_2
done
```

Valoarea testată de comanda while este codul de retur al ultimei comenzi din `listcomanda_1`. Dacă acesta este `0`, se execută `listcomanda_2`, după care se reia execuţia lui `listcomanda_1` s.a.m.d. Dacă s-a returnat o valoare diferită de zero, ciclul se termină.

Structura de ciclare `until` are sintaxa:

```
until listcomanda_1
do
  listcomanda_2
done
```

Semantica structurii `until` este asemănătoare cu cea a structurii `while;` condiţia de terminare a ciclării în cazul structurii `until` este inversă faţă de `while` .

Comanda `shift` are ca efect eliminarea primului argument din linia de comandă şi deplasarea spre stânga a argumentelor rămase: `$2` devine `$1` etc.

Exemplu. Afişarea parametrilor din linia de comandă:

```
while [ $# -gt 0 ]
do
   echo $1
   shift
done
```

Se utilizează variabila de mediu #. Prin `shift`-are, numărul de parametri se decrementează. Dacă scriptul prezentat are numele `ListFis`, şi se tastează:

```
$ListFis *
```

efectul va fi acelaşi cu cel al comenzii:

```
$ls
```

adică afişarea numelor fişierelor din catalogul curent.

Exemplu. Sortarea unui număr oarecare de fişiere specificate în linia de comandă; se verifică şi dacă argumentul dat este un fişier.

```
while [ $# -gt 0 ]
  do
  if  [ -s  $1  ]
  then sort $1 | more
  else echo ”$1 nu exista”
  fi
  shift
done
```

Acelaşi lucru se poate realiza folosind structura `until`.

```
until [  $# -eq  0 ]
  do
  if [-s $1 ]
  then sort $1 | more
  else echo ”$1 nu exista”
  fi
  shift
done
```

Comanda `read` citeşte câte o linie din fişierul standard de intrare şi atribuie cuvintele introduse unor variabile de mediu. Are sintaxa:

```
read [-r] [Lista_nume]
```

Cuvintele din linia citită se atribuie pe rând variabilelor din `Lista_nume`. Dacă numărul de cuvinte din linie este mai mare decât numărul de variabile, valoarea primită de ultima variabilă cuprinde toate cuvintele rămase. Dacă este prezentă opţiunea `-r`, *backslash* se consideră parte din linie. În felul acesta pot fi specificate valori pe mai multe linii (caracterul '\' nu va face parte din nici o valoare). În caz că lista de nume lipseşte, cuvintele citite se atribuie variabilei predefinite REPLY. Execuţia comenzii `read` returnează codul `0`, cu excepţia cazului când se ajunge la sfârşitul fişierului standard de intrare.

Exemplu. O procedură shell de creare a unei agende telefonice

```
while read nume pren tel
do
echo -n $nume '' '' $pren '''' $tel >>agenda
done
cat agenda
```

Comanda `break [n].` Dacă argumentul `n` lipseşte, are ca efect ieşirea dintr-o structură de ciclare (`for`, `while` sau `until`); codul de retur este `0`, cu excepţia cazului când shell-ul nu execută un ciclu în momentul întâlnirii unui break. Dacă argumentul `n` este prezent, el reprezintă numărul de cicluri imbricate din care se iese. Dacă argumentul `n` este mai mare decât nivelul de imbricare, se părăseşte ciclul cel mai din exterior.

Comanda `continue [n].` Dacă argumentul `n` lipseşte are ca efect trecerea la iteraţia următoare a unei structuri de ciclare. Dacă argumentul `n` este prezent, se trece la iteraţia următoare a unui ciclu exterior celui în care apare comanda.

Exemplu. Ieşirea dintr-o structură interioară.

```
for (( a = 1; a < 4; a++ ))
do
echo "Structura exterioara: $a"
for (( b = 1; b < 100; b++ ))
do
if [ $b -eq 5 ]
then
break
fi
echo " Structura interioara: $b"
done
done
```

Exemplu. Ieşirea din ambele structuri de ciclare.

```
for (( a = 1; a < 4; a++ ))
do
echo " Structura exterioara: $a"
for (( b = 1; b < 100; b++ ))
do
if [ $b -gt 4 ]
then
break 2
fi
echo " Structura interioara: $b"
done
done
```

Exemplu. Trecerea la iteraţia următoare.

```
for (( var1 = 1; var1 < 15; var1++ ))
do
if [ $var1 -gt 5 ] && [ $var1 -lt 10 ]
then
continue
fi
echo "Numar Iteratie: $var1"
done
```

Exemplu. Utilizarea comenzii `continue` împreună cu structurile `for` şi `while`.

```
var1=0
while echo " Iteratia lui while: $var1"
[ $var1 -lt 15 ]
do
if [ $var1 -gt 5 ] && [ $var1 -lt 10 ]
then
continue
fi
echo " Numar de iteratie din interior: $var1"
var1=$[ $var1 + 1 ]
done
```

Exemplu. Trecerea peste 2 iteraţii.

```
for (( a = 1; a <= 5; a++ ))
do
echo "Iteratia $a:"
for (( b = 1; b < 3; b++ ))
do
if [ $a -gt 2 ] && [ $a -lt 4 ]
then
continue 2
fi
var3=$[ $a * $b ]
echo " Rezultatul lui $a * $b este $var3"
done
done
```

1. Se cere un script care concatenează un număr oarecare de fişiere date ca argumente în linia de comandă.
2. Se cere un script care pentru fiecare fişier ASCII dintr-un director dat ca parametru şi din toate subdirectoarele lui, se vor afişa primele `n` linii (`n` dat ca parametru).
3. Se cere un script, care din `10` în 10 secunde (folosind un ciclu while) afişează încărcarea sistemului (comanda `uptime`) şi spaţiul ocupat pe disc.
4. Se cere un script care din 10 in 10 secunde afişează pe ecran utilizatorii din reţea (în ordine alfabetică) şi staţiile pe care sunt conectaţi.