Skrypty basha, part 2

Na poprzednich zajęciach poznaliśmy podstawy skryptowania. Dzisiaj poznamy dalsze podstawy skryptowania, czyli funkcje, pętle i może zrobimy kilka bardziej życiowych zadanek

Pętle while

W języku C mogliście mieć już pętle while, do while i for. W skryptach basha są one całkiem podobne, ale, jak już zdążyliśmy się przyzwyczaić, z dziwną składnią. Poniższa pętla, czyli while, wykonuje się tak długo, jak warunek jest spełniony. Więc na przykład pętla while [true] będzie się wykonywać w nieskończoność.

```
while [ warunek ]
do
      cośtam
done
```

Dla przykładu:

```
#!/bin/bash
licznik=10
while [ $licznik -gt 0 ]
do
    echo obejście petli numer: $licznik
    ((licznik--))
done
```

Po krótkiej analizie kodu możemy dojść do wniosku, że nie ma on zbyt wiele sensu:

```
michal@michal-VirtualBox:~$ ./skrypt8.sh
obejście petli numer: 10
obejście petli numer: 9
obejście petli numer: 8
obejście petli numer: 7
obejście petli numer: 6
obejście petli numer: 5
obejście petli numer: 4
obejście petli numer: 3
obejście petli numer: 2
obejście petli numer: 1
```

Zadanie na 45 sekund:

Napraw powyższy program tak, żeby przejścia pętli numerowały się poprawnie, a nie od końca.

Pętla until

Składniowo identyczna do pętli while, ale wykonuje się tak długo, jak warunek **nie** jest spełniony. Aby najprościej zobrazować:

```
#!/bin/bash
licznik=10
until [ $licznik -le 0 ]
do
    echo obejście petli numer: $licznik
    ((licznik--))
done
```

```
#!/bin/bash
licznik=10
while [ $licznik -gt 0 ]
do
    echo obejście petli numer: $licznik
    ((licznik--))
done
```

powyższe skrypty dają identyczny wynik.

Pętla for

Pętla for w bashu działać może już zupełnie inaczej niż znana z C, a bardziej jak taka w Pythonie. Tutaj pętli podajemy serię argumentów, a ona wykonuje się nam tyle razy, ile podamy jej elementów.

```
for <var> in $zmienna

do

      cokolwiek

done
```

```
#!/bin/bash

potrawy='spaghetti jajecznica pomidor'
for food in $potrawy
do
    echo $food
done
```

I wynik działania powyższego, godnego inżyniera, kawałka kodu:

```
michal@michal-VirtualBox:~$ ./skrypt8.sh
spaghetti
jajecznica
pomidor
```

A żeby nie było za prosto i zbyt logicznie – poniżej pętla for taka jak w C. Też działa perfekcyjnie

```
#!/bin/bash

potrawy='spaghetti jajecznica pomidor'
for ((i=0;i<10;i++))
do
    echo hello $i
done
```

Ważna informacja: pętla for działa na dowolnej podanej liście. Dlatego często używana jest na przykład razem z poleceniem find, które wyświetla nam zadane pliki w podrzędnych katalogach.

Dla przykładu:

```
#!/bin/bash
for name in $(find -name $1)
do
    echo $name
done
```

```
michal@michal-VirtualBox:~$ ./skrypt8.sh jajco.txt
./jajco.txt
./skrypty/jajco.txt
./PJATK/jajco.txt
./Dokumenty/jajco.txt
./Dokumenty/Przepisy/jajco.txt
```

Powyższy kod nie ma większego sensu, find zrobiłby dokładnie to samo i bez pętli, a więc:

Zadanie na 3 minuty:

Używając polecenia find i pętli for wyszukaj wszystkie pliki txt i połącz je wszystkie w jeden plik polaczone.txt

Case

porównanie case do znanego z C switcha jest nieuniknione:

```
case $zmienna in

      przypadek1) instrukcja;;

      przypadek2 | przypadek3) instrukcja;;

      *)    instrukcja dla każdego innego przypadku;;

esac
```

Warto zwrócić uwagę na trzecią linię powyższego przykładu. Takie zastosowanie pozwala nam na wykorzystanie jednej instrukcji dla kilku przypadków.

```
#!/bin/bash
case $1 in
    txt) echo wszystkie pliki txt
    ls -l *.txt;;
    sh) echo wszystkie skrypty:
    ls -l *.sh;;
    *) echo wszystkie pliki i katalogi:
    ls -l;;
esac
```

Funkcje

```
nazwa_funkcji(){

      zawartość_funkcji

}

nazwa_funkcji # wywołanie funkcji
```

Do funkcji możemy podawać również argumenty. Działa to dokładnie w ten sam sposób, w jaki podawanie argumentów do skryptu, z tą różnicą, że argumenty podawane są po wywołaniu wewnątrz skryptu, nie z wiersza poleceń:

```
#!/bin/bash
jajco(){
    echo argument podany do funkcji: $1
}

jajco Argument123
```

Zadania na dziś

1. Napisz skrypt wyświetlający, który będzie sprawdzał, czy przekazano do niego przynajmniej 5 argumentów numerycznych, a jeśli tak, to doda je wszystkie do siebie i wydrukuje wynik na terminalu
2. Napisz skrypt kopiujący do katalogu Documents/copied pliki podane jako argumenty. Jeśli podany zostanie jeden argument, to przekopiuje wszystkie pliki o podanym jako argument rozszerzeniu z katalogów podrzędnych do Documents/copied
3. Napisz skrypt dodający do pliku baza.txt wartości podane jako argumenty bądź przy użyciu `read`. Plik baza.txt ma posiadać 4 kolumny: Imię, nazwisko, płeć i wiek.