

Paradygmaty programowania - ćwiczenia Lista 6

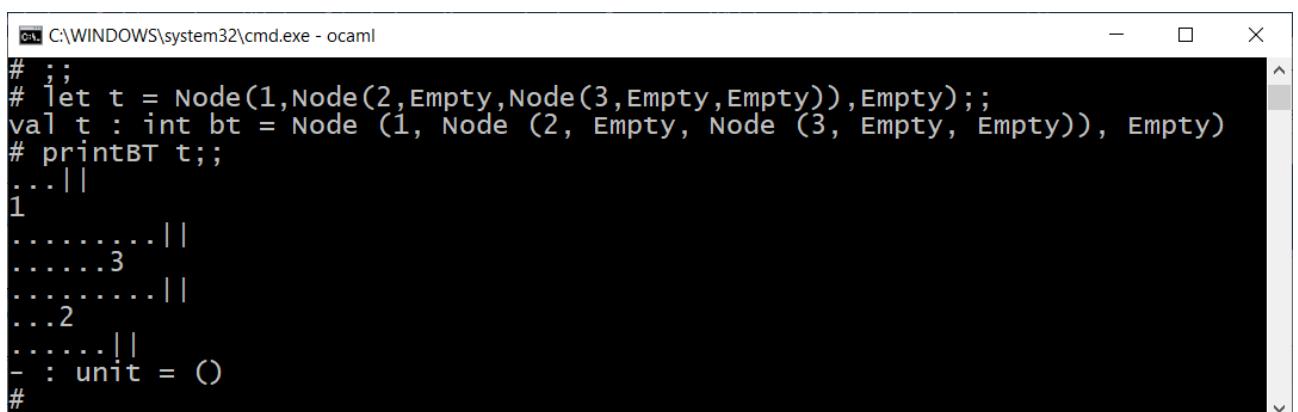
Na wykładzie 4 został zdefiniowany następujący typ dla drzew binarnych:

`type 'a bt = Empty | Node of 'a * 'a bt * 'a bt;;`

Jako (użyteczny!) przykład programu z efektami obliczeniowymi w języku funkcyjnym poniżej zdefiniowano funkcję `printBT: int bt -> unit`, która w czytelny sposób wyświetla zadane drzewo binarne (odwrócone w lewo o 90 stopni).

```
let printBT tree =  
  let rec aux(t, height) =  
    match t with  
    | Node(elem, left, right) ->  
      begin  
        aux(right, height + 1);  
        for i = 0 to height - 1 do print_string "..."; print_int elem; print_newline();  
        aux(left, height + 1)  
      end  
    | Empty -> for i = 0 to height - 1 do print_string "..."; print_endline "||"  
  in aux(tree, 0)  
;;
```

Poniższy zrzut ekranu ilustruje jej działanie dla prostego drzewa binarnego.



```
C:\WINDOWS\system32\cmd.exe - ocaml  
# ;;  
# let t = Node(1,Node(2,Empty,Node(3,Empty,Empty)),Empty);;  
val t : int bt = Node (1, Node (2, Empty, Node (3, Empty, Empty)), Empty)  
# printBT t;;  
...||  
1  
.....||  
.....3  
.....||  
...2  
.....||  
- : unit = ()  
#
```

Paradygmaty programowania - ćwiczenia

Lista 6

1. (Scala) Jedną z pętli w języku Scala ma następującą składnię: *while* (*warunek*) *wyrażenie*. Napisz w Scali funkcję *whileLoop* (**bez używania efektów obliczeniowych**), która pobiera dwa argumenty: warunek oraz wyrażenie i dokładnie symuluje działanie pętli *while* (w Scali 2 również składniowo). Jakiego typu (i dlaczego) muszą być argumenty i wynik funkcji? Przetestuj funkcjonalność *whileLoop* na poniższym programie (tj. zasymuluj jego działanie).

```
var count = 0
while count < 3 do
    println(count)
    count += 1
```

2. (Scala) Zapisz w języku Scala zaprezentowane na wykładzie funkcje a) *swap*, b) *partition*, c) *quick* i d) *quicksort*, zachowując ich styl programowania. Funkcje nie muszą być polimorficzne.

Działanie programów z zadań 3-5 należy wyjaśniać, rysując „obrazy pamięci” tych programów, tzn. rysując referencje jako strzałki, komórki pamięci i ich zawartości jako prostokąty. Należy pokazać, co znajdzie się na stosie, a co na sterwie (patrz wykład 2, str. 15-16).

3. Co i **dlaczego** wydrukuje poniższy program w Javie?

```
public class IsEqual{
    static boolean isEqual1(int m, int n){return m == n;}
    static boolean isEqual2(Integer m, Integer n){return m == n;}
    public static void main(String[] args){
        System.out.println(isEqual1(500,500));
        System.out.println(isEqual2(500,500));
    }
}
```

4. Co i **dlaczego** wydrukuje poniższy program w Javie?

```
public class Porównanie{
    public static void main(String[] args){
        String s1 = "foo";
        String s2 = "foo";
        System.out.println(s1 == s2);
        System.out.println(s1.equals(s2));
        String s3 = new String("foo");
        System.out.println(s1 == s3);
        System.out.println(s1.equals(s3));
    }
}
```

5. Co i **dlaczego** wydrukuje poniższy program w Javie?

```
public class Aliaszy{
    public static void main(String[] args){
        int[] ints = {1,2,3};
        for(int i : ints) {
            System.out.println(i); i = 0;
        }
        for(int i : ints)
            System.out.println(i);
        int[] ints2 = ints;
        for(int i=0; i<ints2.length; i++) {
            System.out.println(ints2[i]); ints2[i] = -1;
        }
        for(int i : ints) System.out.println(i);
    }
}
```