Programowanie

O stylu programowania

Tomasz Wierzbicki

Instytut Informatyki UWr.

25 marca 2009

Piszę w Prologu, ale myślę w C...

```
advance(L, R) :-
   advance(L, [], W),
   reverse(W, R).

advance([], A, A).
advance([H|T], A, W) :-
   H1 is H + 1,
   advance(T, [H1|A], W).
```

Piszę w Prologu i myślę w Prologu!

```
advance([], []).
advance([H|T], [H1|S]) :-
   H1 is H + 1,
   advance(T,S).
```

Piszę w Prologu, ale myślę w C...

```
chop(N, L) :-
   chop(N, 1, [], L).

chop(0, _, A, A) :-
   !.
chop(N, K, A, L) :-
   K =< N,
   N1 is N - K,
   chop(N1, 1, [K | A], L).
chop(N, K, A, L) :-
   K =< N,
   K1 is K + 1,
   chop(N, K1, A, L).
```

Piszę w Prologu i myślę w Prologu!

```
chop(0, []).
chop(N, [K | L]) :-
   between(1, N, K),
   N1 is N - K,
   chop(N1, L).
```

Programowanie deklaratywne. Przykład: formatowanie wydruku

2	3	5	7	11	13	17	19	23
29	31	37	41	43	47	53	59	61
67	71	73	79	83	89	97	101	103
107	109	113	127	131	137	139	149	151
157	163	167	173	179	181	191	193	197
199	211	223	227	229	233	239	241	251
257	263	269	271	277	281	283	293	307
311	313	317	331	337	347	349	353	359
367	373	379	383	389	397	401	409	419
421	431	433	439	443	449	457	461	463
467	479	487	491	499	503	509	521	523
541	547	557	563	569	571	577	587	593
599	601	607	613	617	619	631	641	643
647	653	659	661	673	677	683	691	701
709	719	727	733	739	743	751	757	761
769	773	787	797	809	811	821	823	827
829	839	853	857	859	863	877	881	883
887	907	911	919	929	937	941	947	953
967	971	977	983	991	997			

Rozwiązanie imperatywne (Java)

```
void printPrimes(int number) {
   Iterator<Integer> primes = new Primes();
   int pos = 0;
   for (int p : primes) {
      if (p > number)
         break;
      if (pos++ >= numOfColumns) {
         System.out.println();
         pos = 1;
      }
      System.out.print(String.format("%" + fieldWidth
                                     + "d", p));
   }
   System.out.println();
}
```

Rozwiązanie deklaratywne (Haskell)

```
printPrimes :: Integer → IO ()
printPrimes num =
    do
        mapM_ (putStrLn . concatMap pad)
            . takeWhile (not . null)
            . unfoldr (Just . splitAt numOfCols)
            . takeWhile (≤ num)
            $ primes
    where
        pad n = reverse . take fieldWidth
                    $ (reverse . show $ n) ++ repeat ' '
```