Programowanie Funkcyjne 2023

Lista zadań nr 0

Na zajęcia 4 i 6 października 2023

Na zajęciach przede wszystkim przygotowujemy środowisko pracy OCamlem i poznajemy kilka podstawowych cech tego języka. Celem poniższych zadań jest pomoc w przeprowadzeniu tego procesu. Zadania powinny być rozwiązane w trakcie zajęć i nie są punktowane. Jednak zachęcamy do potraktowania tej listy poważnie: jeśli na koniec semestru komuś będzie brakować punktu albo dwóch do wyższej oceny, a aktywnie brał udział w dzisiejszych zajęciach, to prowadzący może (ale nie musi) ten brakujący punkt znaleźć tutaj.

UWAGA! Prowadzący używają systemu GNU Linux i zastrzegają sobie prawo do odmówienia pomocy osobom, które chcą oddawać zadania na innych platformach. Nie oznacza to, że jesteście skazani na Linuksa, ale jak chcecie używać czegoś innego, to może się okazać, że nikt Wam nie pomoże.

Zadanie 1. Zainstaluj kompilator OCamla oraz menadżer pakietów `opam` (dostępne są w repozytoriach większości dystrybucji). Zainicjuj Opama poleceniem `opam init` (jeśli tego jeszcze nie zrobiłeś), a następnie zainstaluj pakiety `utop`, `ocamlbuild` oraz `dune` używając polecenia `opam install`. Znajdź te pakiety, które zawierają w opisie słowo „proof". Wyświetl informacje o pakiecie `ocamlfind`.

Zadanie 2. Czy następujące wyrażenia są dobrze typowane? Spróbuj sam zgadnąć, a następnie sprawdź ich typy i wartości w ewaluatorze `utop`. Pamiętaj, że wprowadzania wyrażeń w ewaluatorze kończymy podwójnym średnikiem.

```
if true then 4 else 5
if false then 1 else 3.5
4.75 + 2.34
false || "ab">"cd"
if true then ()
if false then () else 4
let x = 2 in x^"aa"
let y = "abc" in y^y
(fun x -> x.[1]) "abcdef"
(fun x -> x) true
let x = [1;2] in x@x
let rec f f = f+f in f 42
[1,true]
```

Zadanie 3. Dla każdego z typów `int`, `float`, `char`, `string`, `bool` oraz `unit` napisz wyrażenie tego typu i wykorzystaj ewaluator OCamla do obliczenia jego wartości. Ile jest różnych wartości typu `unit`? Do czego taki typ może być przydatny?

Zadanie 4. Pobierz ze SKOSa przykładowy program `fib.ml`, a następnie go skompiluj (ręcznie z wiersza poleceń) za pomocą kompilatorów `ocamlc` i `ocamlopt` (*zob. Ocaml Manual, Part III*). Porównaj czas wykonania obu plików wykonywalnych otrzymanych za pomocą obu kompilatorów.

Zadanie 5. Pobierz ze SKOSa drugi przykładowy program `grep.tar.gz` i rozpakuj go. Spróbuj skompilować go na następujące sposoby:

- używając systemu budowania `ocamlbuild`[1],
- używając systemu budowania `dune`[2],

[1] `https://nicolaspouillard.fr/ocamlbuild/ocamlbuild-user-guide.html`
[2] `https://dune.readthedocs.io/en/stable/quick-start.html`

- ręcznie wykonując szereg podstawowych poleceń (powinny być 4).

Następnie zajrzyj do pliku `matcher.ml` i podmień implementację tego modułu (alternatywna wersja jest zakomentowana) i powtórz polecenie zadania. Tym razem będzie trudniej, bo program korzysta z biblioteki `Str`, która choć jest częścią biblioteki standardowej, to domyślnie nie jest dołączana do programu.

Zadanie 6. Zidentyfikuj zmienne wolne i związane w wyrażeniach:

```
let x = x in x^x
let x = 10. in let y = x**2. in y*.x
let x = 1 and y = x in x + y
let x = 1 in fun y z -> x*y*z
```

Zadanie 7. Jaka jest wartość wyrażenia `f 1` w środowisku otrzymanym po przetworzeniu następujących definicji:

```
# let m = 10;;
# let f x = m + x;;
# let m = 100;;
```

i dlaczego?

Zadanie 8. Napisz wyrażenie, które przekonuje, że Ocaml używa gorliwej strategii ewaluacji.