Lista zagadnień nr 9

Język WHILE

Ćwiczenie 1.

Napisz w języku WHILE program do liczenia n-tej liczby Fibonacciego i największego wspólnego dzielnika dwóch liczb.

Ćwiczenie 2.

Dodaj do języka pętlę `for` (np. z semantyką taką jak w C).

Ćwiczenie 3.

Dodaj do języka WHILE konstrukcję `break`, która zrywa wykonanie najbardziej wewnętrznej pętli.

Samo-interpreter

Ćwiczenie 4.

Dodaj do języka z wykładu (`letrec.rkt`) formę specjalną `struct` pozwalającą na zdefiniowanie struktury (tj. konstruktora, selektorów i predykatu) o zasięgu podwyrażenia, analogicznie do naszych let-wyrażeń. W łatwiejszej wersji niech abstrakcja dostarczana przez struktury będzie konwencjonalna, jak poniżej:

```
> (struct foo (a b)
    (let (x (foo 1 2))
      (pair (fst x) (snd x))))
(foo 1)
```

W wersji trudniejszej — zadbaj o to, by w takiej sytuacji zwracać komunikat o błędzie, związany z użyciem niewłaściwego selektora.

Ćwiczenie 5.

Rozszerz język z wykładu o formę specjalną `case` — albo jako cukier składniowy, albo odpowiednio rozbudowując interpreter.

Ćwiczenie 6.

Zareprezentuj w języku z wykładu jego własną składnię jako typ danych, używając do tego odpowiednich struktur (lub abstrakcji konwencjonalnej) i predykatów.

Ćwiczenie 7.

Zaimplementuj w języku z wykładu interpreter samego siebie. Tym razem nie interpretuj procedur czy wyjątków przy użyciu prostszych narzędzi: każdą konstrukcję zinterpretuj przez ją samą na meta-poziomie. Nie przejmuj się też ładną obsługą błędów programisty.

Ćwiczenie 8.

Wzorując się na implementacji z wykładu (ale pomijając ładną obsługę błędów) zaimplementuj pętlę REPL dla swojego interpretera, aby móc uruchomić go wewnątrz samego siebie. Jak kolejne poziomy interpretacji wpływają na prędkość działania systemu?