Pracownia z Kursu języka Erlang
Lista 3

Można i należy używać modułów `lists` oraz `string`. Proszę używać odpowiedzialnie operatora `++`.

1. **(1pkt)** Napisz program działający jako skrypt powłoki systemowej, który ze standardowego wejścia przeczyta ciąg cyfr szesnastkowych (opisanych wyrażeniem regularnym `[0-9A-Fa-f]{2}+`). Potraktuj każde dwie cyfry jako bajt i skonwertuj całość do wartości typu `binary`, która będzie interpretowana jako pakiet `IPv4`. Wydrukuj pola nagłówków protokołu (nazwa pola, wartość), dane pakietu wydrukuj jako ciąg cyfr szesnastkowych. Interpretacja zawartości pól nagłówków nie jest konieczna. Sprawdź poprawność danych zawartych w pakiecie – długość i sumę kontrolną. Do realizacji zadania użyj rozpakowywania wartości `bitstring`. Jako dane testowe możesz użyć dane zebrane przy pomocy programu `tcpdump` np.:

   ```
   $ tcpdump -X icmp                # tylko ICMP
   $ tcpdump -X udp and dst port 53 # tylko UDP na port DNS
   $ tcpdump -X tcp and dst port 80 # tylko TCP na port HTTPS
   ```

   Powyższe polecenia przechwytują na domyślnym interfejsie sieciowym pakiety protokołu `IPv4` i drukują je w postaci szesnastkowej na ekran (twój program może przyjmować dane również w tej postaci).

2. **(1pkt bonusowy)**[1] Rozszerz poprzednie zadanie tak by interpretować również pakiety `ICMP`, `UDP` i `TCP`. Dodaj częściową interpretację pól protokołu.

3. **(1.5pkt)** Zaimplementuj moduł `db` zawierający prostą bazę danych przechowującą rekordy. Każdy rekord musi mieć przypisany unikalny numer identyfikacyjny.

   create/1 bierze krotkę `{Moduł,Rekord}`, która mówi o tym jakie rekordy będą trzymane w bazie. Zwraca krotkę `{ok,NewDb}` lub `{error,description}` jeśli nie udało się odczytać informacji o rekordzie.

   insert/2 bierze bazę oraz krotkę wcześniej podanego typu i wkłada ją do bazy danych. Zwraca krotkę składającą się z nowej bazy danych i unikalnego identyfikatora rekordu.

   select/2 bierze bazę oraz predykat, który pełni rolę zapytania. Jeśli dla rekordu predykat zwróci prawdę, to para `{Id,Rekord}` zostanie umieszczona na liście wyników. Funkcja zwróci `{ok,ListaWyników}` lub `{error,wrong_query}` jeśli wykonanie predykatu zawiedzie z jakichś przyczyn.

   select/3 j.w. ale dodatkowo pobiera nazwę pola rekordu, na którym ma działać predykat.

   delete/2 lista argumentów j.w. Jeśli dla rekordu predykat zwróci prawdę, to rekord zostanie usunięty z bazy danych i a para `{Id,Rekord}` zostanie umieszczona na liście usuniętych. Funkcja ma zwrócić `{ok,NewDb,ListaUsuniętych}` lub `{error,wrong_query}` jeśli wykonanie predykatu zawiedzie z jakichś przyczyn.

   update/3 bierze bazę, predykat zapytania i funkcję aktualizacji rekordu. Jeśli dla danego rekordu predykat zwróci prawdę, to rekord ten jest aktualizowany wywołaniem na nim *funkcji aktualizacji rekordu*, a para `{Id,NowyRekord}` zostanie umieszczona na liście wyników. Funkcja ma zwrócić `{ok,NewDb,ListaZmodyfikowanych}` lub `{error,wrong_query}` lub `{error,wrong_updater}` jeśli wykonanie predykatu zapytania lub funkcji aktualizującej rekord zawiedzie z jakichś przyczyn.

   update/4 j.w. ale dodatkowo (jako drugi argument) pobiera nazwę pola rekordu, na którym ma działać predykat i funkcja aktualizacji rekordu.

   Do wykonania zadania mogą się przydać funkcja `record_info`.

   ```
   1> rd( rekord, { foo, bar } ).
   rekord
   2> record_info( size, rekord ).
   3
   3> record_info( fields, rekord ).
   [foo,bar]
   ```

Lista i materiały znajdują się pod adresem
http://cahirwpz.cs.uni.wroc.pl/main-pl/erlang-language-summer-2010/ *Krystian Bacławski*

[1] Nie wlicza się do liczby możliwych do uzyskania punktów.