Kartkówki Erlang 2018/19

Kartkówka 1

1. Wymień twórców erlanga
   Joe Armstrong, Robert Virding, and Mike Williams

2. Napisz funkcję silnia

   silnia(0) -> 1;

   silnia(N) -> N * silnia(N-1).

3. Napisz czym różnią się == i =:=

   W języku Erlang == i =:= są dwoma różnymi operatorami porównania.

   Operator == porównuje wartości z uwzględnieniem konwersji typów. Oznacza to, że porównuje wartości i próbuje je przekształcić do jednego typu, jeśli różnią się typami. Na przykład, 1 == 1.0 zwróci wartość true, ponieważ wartości są równe, a operator == uwzględnił konwersję typów.

   Operator =:= porównuje wartości, ale nie wykonuje konwersji typów. Oznacza to, że tylko wartości o takim samym typie mogą być porównywane. Na przykład, 1 =:= 1.0 zwróci wartość false, ponieważ wartości są różnego typu, a operator =:= nie wykonuje konwersji typów.

   Podsumowując, operator == porównuje wartości z uwzględnieniem konwersji typów, podczas gdy operator =:= porównuje wartości bez konwersji typów.

   1> (1 == 1.0). → True

   2> (1 =:= 1.0). → False

4. Czym się różni krotka od listy?

   W języku Erlang krótka (tuple) to zbiór wartości różnych typów, które są przechowywane jako pojedyncza jednostka. Każda wartość w krótkiej ma przypisany indeks, który można użyć do uzyskania dostępu do tej wartości.

   Lista w Erlangu, z drugiej strony, jest sekwencją elementów tego samego typu, która może zawierać elementy różnych typów. Elementy w liście są przechowywane w kolejności i można do nich uzyskać dostęp za pomocą funkcji listowych.

   Innymi słowy, krótka jest niezmienialną strukturą danych, która przechowuje stałą liczbę wartości różnych typów, podczas gdy lista jest zmienialną sekwencją elementów tego samego typu.

5. Opisać zalety korzystania z rekurencji ogonowej.

Efektywność pamięciowa - dzięki rekurencji ogonowej możemy uniknąć stosu wywołań, co pozwala na efektywne wykorzystanie pamięci. W przypadku rekurencji zwyczajnej, każde wywołanie rekurencyjne musi być przechowywane na stosie, co może prowadzić do przekroczenia limitu stosu w przypadku dużych danych wejściowych.

Wydajność - dzięki rekurencji ogonowej nie ma potrzeby tworzenia nowych ramek stosu dla każdego wywołania rekurencyjnego, co oznacza, że czas wykonywania programu może być znacznie krótszy niż w przypadku rekurencji zwyczajnej.

Czytelność kodu - rekurencja ogonowa pozwala na zastosowanie prostszej i bardziej zwięzłej składni, co ułatwia zrozumienie kodu i jego utrzymanie w przyszłości.

Bezpieczeństwo - rekurencja ogonowa pomaga uniknąć błędów związanych z przekroczeniem limitu stosu. W przypadku rekurencji zwyczajnej, jeśli stos wywołań zostanie przekroczony, program może się zawiesić

6. Podać składnię pozwalającą wywołać funkcję function z modułu module z argumentem argument.
```
module:function(argument).
```
module:function(argument).

7. Jak wypisać 'hello world' na ekran (io:format)
```
io:format("hello world\n").
```
io:format(“hello world”).

8. Sygnatura funkcji, która uruchamia się tylko po przekazaniu liczby (użycie when is_number(x))
```
my_function(X) when is_number(X) ->
    % ciało funkcji dla argumentu będącego liczbą
```
function(X) when is_number(X) ->
        %% ciało funkcji

9. Dodanie elementu na początek listy (np. [element | Lista])
[X | Lista].
[X, Y | Lista].
[0 | [1, 2, 3 ,4]].
[-1, 0 | [1, 2, 3, 4]].