How FP Deals With Effects

✳ PoolC 양제성

Source: 2024/01/24 (Wed)

목차

1st Session

1. 함수형 프로그래밍 Intro
   - Overall Structure
   - Historical Review (CS + Math)
2. SW 엔지니어링의 목표
   - SW Maintainability
   - FP vs OOP vs PP
3. FP는 정말 순수한가?
   - Purity of Functions
   - File I/O Scenario

Basic Haskell Knowledge

2nd Session

1. 함수 합성을 위한 도구들
   - Partial Application
   - Kleisli Composition
2. ...중 하나인 모나드
   - Functor to Monad
   - IO Monad
3. 부수 효과의 관리
   - Action / Calculation / Data
   - Preventing Action Propagation

Overall Structure

FP is all about **composing pure functions**.

```
int main(void) {
  f(); g(); h(); ..
}
```

VS

```
f(g(h(..)))
```

[Procedural Promramming]

[Functional Programming]

Overall Structure

How?

FP is all about **composing pure functions**.

```
int main(void) {
  f(); g(); h(); ..
}
```

VS

```
f(g(h(..)))
```

[Procedural Promramming]

[Functional Programming]

Overall Structure

Sum all. `[stdin <- "5\n1 2 3 4 5"]`

C++ (imperative)

```cpp
int main() {
  int n, result;
  std::cin >> n;
  for (size_t i = 0; i < n; ++i) {
    int a;
    std::cin >> a;
    result += a;
  }
  std::cout << result << '\n';
  return 0;
}
```

[Procedural Promramming]

Haskell (declarative)

```haskell
main =
  interact
    (show . sum .
      map read . drop 1 . words)
```

Python3 (declarative)

```python
from sys import stdin

print(sum(map(
  int, stdin.read().split()[1:]
)))
```

[Functional Programming]

Overall Structure

1. Purity
   - Side Effect
   - Referential Transparency
   - Significance of ...
2. Immutability
   - Recursion (feat. Tail Call Optimization)
   - C vs Haskell in file IO
3. First Class Function
   - Currying
   - Linked List

<Let's code!>