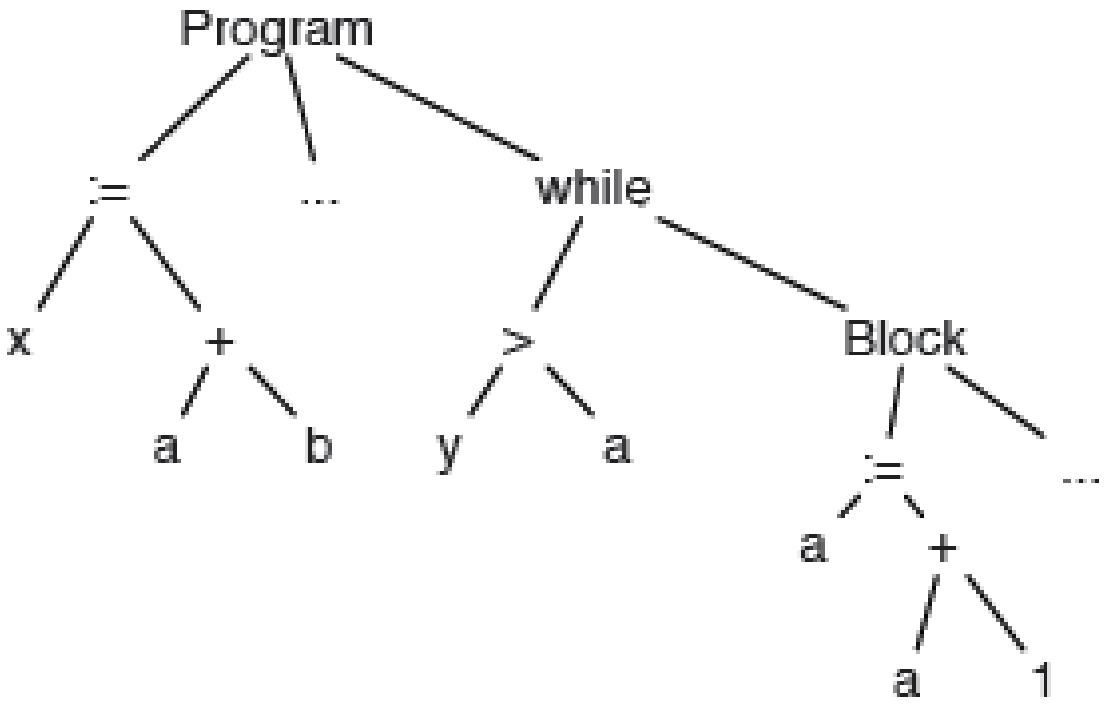
Parametrizacija poenotenega razčlenjevalnika in izpisovalnika

(Unificated parser and pretty printer parametrization)

Rok Kralj
Peter Lazar

Abstraktna sintaksa

```
x := a + b;
y := a * b;
while (y > a) {
   a := a + 1;
   x := a + b
}
```




```
Data Stmt =
    While Stmt [Stmt]
  | Assign Var Stmt
  | Var String
```

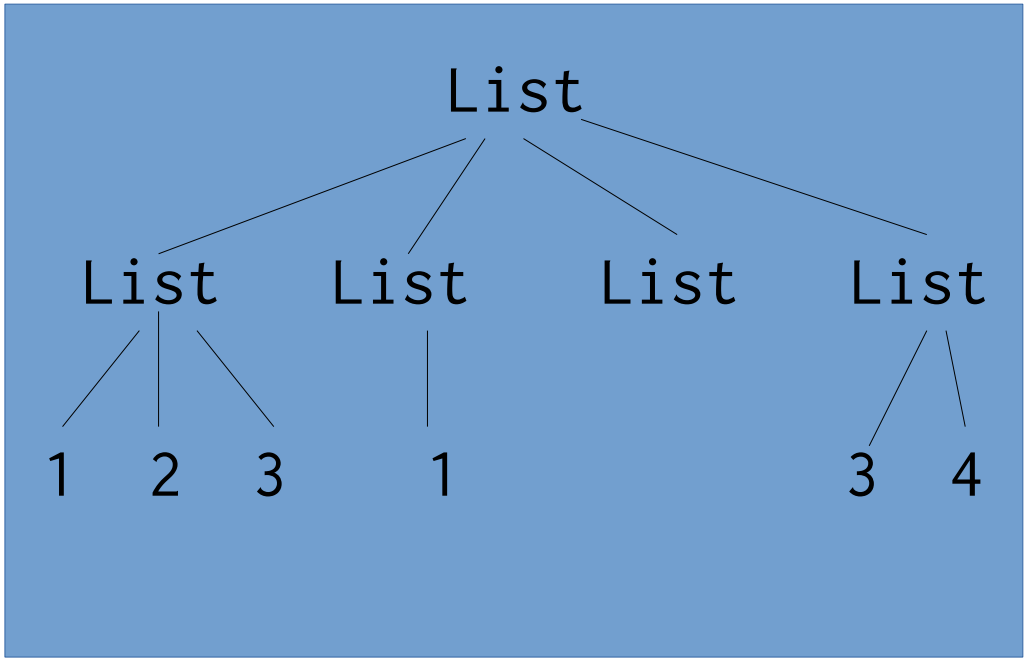
Konkretna in abstraktna sintaksa

```
[[1,2,3],[1],[],[3,4]]
```

$$\begin{aligned}\textbf{data}\ &List\ \alpha \\ &= Nil \\ &\mid Cons\ \alpha\ (List\ \alpha)\end{aligned}$$

```
[[1,   2   ,3], [    1
     ],
[  ], [3,4]]
```

```
[
   [1,2,3],
   [1],
   [],
   [3,4]
]
```



Običajno imamo več predstavitev elementa neke abstraktne sintakse v konkretni sintaksi

Parser

- Konkretna sintaksa → Abstraktna sintaksa

```
Newtype Parser alpha = Parser (String → alpha)

Newtype Parser alpha = Parser (String → Maybe alpha)

Newtype Parser alpha = Parser (String → [alpha])

Newtype Parser alpha = Parser (String → [(alpha,
String)])

parse stevilo »123abc«  →  [(123, »abc«)]
```

Printer

- Abstraktna sintaksa → konkretna sintaksa
- Enostavno.

```
Newtype Printer alpha = Printer (alpha → Maybe alpha)
```

Kombiniranje

- Zakaj kombinirati?
  - Ne želimo podvojevati funkcionalnosti.

(sledi en kratek primerček na tablo)

Članek Rendel & Ostermann

- Kako smo to počeli do sedaj?
  - Parsec za parsanje
  - lastna toString funkcija za izpisovanje

- Pol manj dela (v najbolj optimističnem primeru)
- ... ?

Najina naloga: Parametrizacija

- Osvežitev spomina: Več predstavitev istega elementa abstraktne sintakse v konkretni sintaksi.

- Katera je prava?
  - Naj si to izbere uporabnik (ne pisec sintakse).
  - Naj poda konfiguracijo kot tretji parameter v print funkcijo.

```
Newtype Printer alpha = Printer (alpha → Maybe alpha)
```

Popravimo v:

```
Newtype Printer alpha = Printer
    (alpha → ReaderT Config Maybe alpha)
```

Dodamo dva nova funktorja

- `(<$>)  ::  Iso alpha beta -> delta alpha -> delta beta`
- `(<$$>)  ::  IsoM alpha beta -> delta alpha -> delta beta`
- `(<-$>)  ::  (Config -> Config) -> delta alpha -> delta alpha`

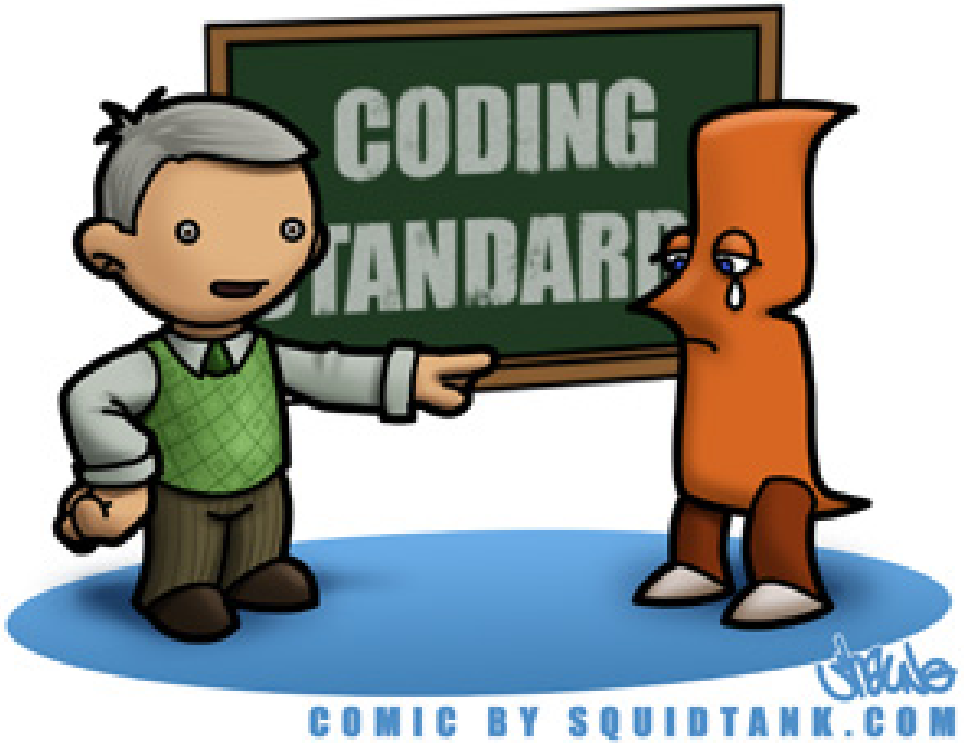
Baterije so priložene

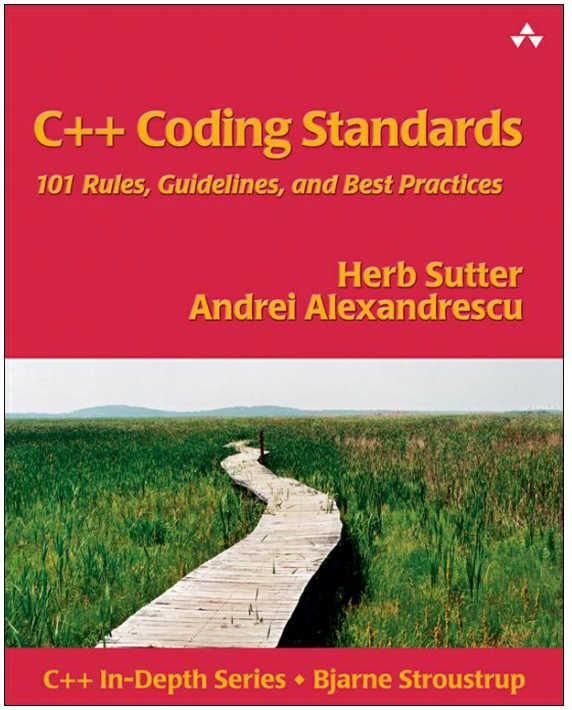
- Parser in pretty printer za:
  - Json
  - C

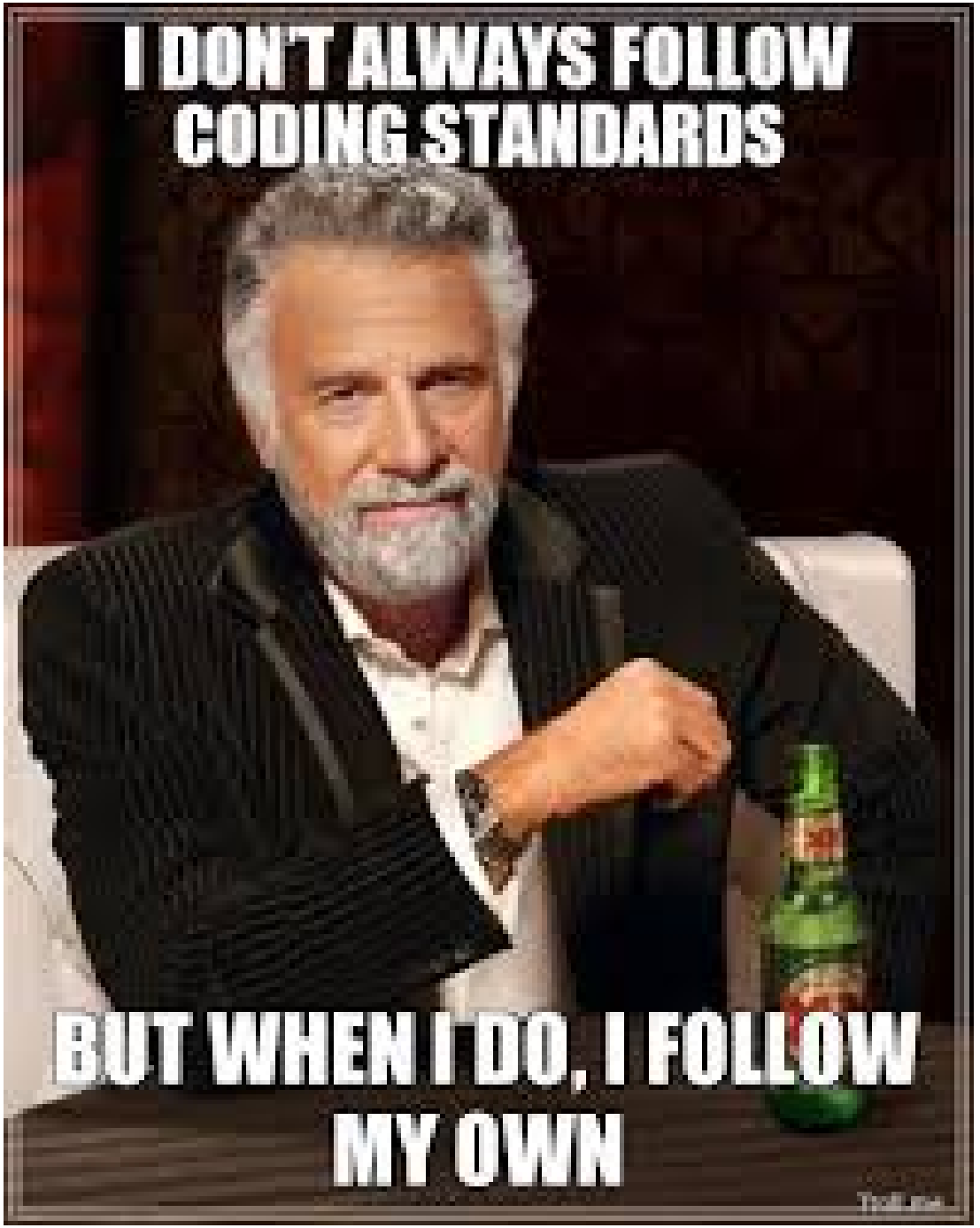
```haskell
data JsonConfig = JsonConfig {
    indentDepth :: Int,
    indentOneLevel :: String,
    spaceAfterColon :: Bool,
    unicodeEscape :: Bool
} deriving (Show)
```

(demonstracija)

Praktična uporaba? Vsekakor.






I DON'T ALWAYS FOLLOW CODING STANDARDS
BUT WHEN I DO, I FOLLOW MY OWN

Shema

Rok

- zamika s tabulatorji
- zavite oklepaje pušča v isti vrsti: if (x) {
- dela presledke okoli *

Peter

- ima raje presledke
- zavite oklepaje daje v naslednji vrsto: if (x) {
- dela presledke okoli *

1. Ko Rok naredi commit, se njegova koda sparsa in sprinta glede na zahteve podjetja / skupnosti

2. Ko Peter odpre datoteko, se ta izpiše glede na njegovo specifikacijo.

git hook

Vsa koda v sistemu za nadzor verzij tako upošteva dan coding standard skupnosti

Hvala.