Gradia

Gradia Programming
Language Community
梶塚太智

式レベル型注釈による段階的型付きLisp系言語

```
gradia — cargo
kajizukataichi@MacBook-Pro gradia % cargo run
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.19s
     Running `target/debug/gradia`
Gradia 0.1.0
> (define '(x2 n) '(* n 2))
(lambda '(n) '(* n 2))
> (x2 4)
8
> (x2 " 5")
10
> (define '(x2 n:number) '(* n 2))
(lambda '(n:number) '(* n 2))
> (x2 6)
12
> (x2 "7 ")
Error! the passed argument value `"7 "` is different to expected type `number` of the function
>
```

型注釈が無いため文字列でも
暗黙の型変換して実行可能

型注釈があるため、
引数をその型に強制

一般的な言語では型脚注は
基本的に変数か関数のみ可能だが
Gradiaではリストから値まで
全ての式に型注釈を付けられる

```
(define:function
    '(x2:symbol n:number):list
    '(*:function n:number 2:number
):list):function

```

式のデータ構造は評価されるリストの値と期待する型の注釈を管理する

```rust
#[derive(Clone)]
pub struct Expr {
    pub expr: Type,
    pub annotate: Option<Class>,
}
```

実行時に厳密な型チェックを実施

```rust
// Type check between result value and except type
if let Some(annotate) = self.annotate.clone() {
    if &result.get_type() == &annotate.get_type() {
        Ok(result)
    } else {
        return Err(GradiaError::Type(result, annotate.get_type()));
    }
} else {
    Ok(result)
}
```

式レベルで型注釈のアイデアは、Gradia言語だけに留まらず
他の色々な言語にとって有用な機能になることが今後期待できる