Pen 言語の進捗報告

@raviqqe

目次

- 進捗報告
  - 並列計算構文の実装
  - Rust FFI の強化
- 今後の予定

進捗報告

並列計算構文の実装

`go` 構文とは

- Go 言語にある goroutine (fiber 的なもの) を実行する構文
- Generics がある言語ではただの関数でもよい
    - 例: Rust の `std::thread::spawn()` や `tokio::spawn()`

```
func main() {
  go func() {
    doSomething()
  }()
}
```

並列計算構文の実装

Pen 言語での実装

- Go 言語の `go` 構文ほぼそのまま
  - 式では無く実行させる関数を渡す
  - 元の関数の結果を返すフューチャを作る
- 型としては `\(\() T) \() T`
- `\(\() any) \() any` の型を持つ外部関数が本体
- ポリモーフィズム等はコンパイラ側で対応

```
main = \() {
  future = go \() number {
    doSomething()
  }

  ...
}
```

並列計算構文の実装

Rust FFI 側の実装

- 各システムパッケージが実装
- **並列・並行に計算することは保証されていない**
- 以下は `os` パッケージ（非同期ランタイム版）の例

```
#[no_mangle]
extern "C" fn _pen_spawn(closure: ffi::Arc<ffi::Closure>) -> ffi::Arc<ffi::Closure> {
    ffi::future::to_closure(spawn_and_unwrap(ffi::future::from_closure(closure)))
}

async fn spawn_and_unwrap<F: Future<Output = ffi::Any> + Send + 'static>(future: F) -> ffi::Any {
    spawn(future).await.unwrap()
}
```

Rust FFI の強化

- Rust の型・関数の Pen 言語へのエクスポートを簡略化
- Rust の `std::future::Future` 型と Pen 言語のフューチャ型の相互変換

例

```
#[ffi::any]
struct Foo {
  ...
}
```

```
#[ffi::bindgen]
fn foo() { ... }

#[ffi::bindgen]
async fn foo() { ... }
```

Rust FFI の強化

例（続き）

- `ffi::future::to_closure<O, F: Future<Output = O>>(future: F) -> ffi::Arc<ffi::Closure>`
- `async ffi::future::from_closure<T>(closure: ffi::Arc<ffi::Closure>) -> T`

今後の予定

- リスト内包表記
- Select 構文
- 実用性向上
  - 複数のシステムパッケージの利用
  - 標準ライブラリの拡張

まとめ

- `go` 構文を実装した
- 今月は、言語機能の強化が主になりそう