

プログラミング Rust

環境構築 / 変数と if

岡本 悠吾

2024-hoge-hoge

OECU Programming Circle

この講義の目的

1. Rust のプログラミング環境を構築できる
2. 変数と if 文が使えるようになる

備考

- : 実践項目
 - プログラミング, ターミナルでの作業など
 - : 検索したりして是非調べてほしいもの

Rust とは？

What is Rust language?

- Mozilla が開発しているプログラミング言語
- オープンソースのコミュニティベースで絶賛開発中
- 2015 年に version 1.0 がリリース
 - Golang v1.0: 2012 年, Kotlin v2.0: 2012 年
- 2016 以降, 每年 StackOverflow にて 「**プログラマーが最も愛する言語**」 1 位を獲得し続いている
- Rust = 鑄

使いみち

- Web アプリのバックエンド部 / Linux kernel / Windows NT kernel / Android SDK / 組み込みシステム

1. メモリ安全

- 変数の所有権, 借用チェック, 参照, 超強力な型システム
- メモリ(RAM)系のバグやセキュリティホールから守る
 - Java, Go, Python

2. 実行速度が爆速

- コンパイラ基盤 `llvm` を使用
- ガベージコレクションがない
 - メモリ安全な言語で GC が無いのはこいつだけ
- 理論上, C 言語と同等速度

3. IDE レスで開発可能

- rustup, cargo などの有能ソフトウェアが標準で付属

環境構築

macOS, GNU/Linux, xBSD, Solaris

1. パッケージマネージャーから (推奨)

```
sudo apt install rustup # Debain  
sudo pacman -S rustup # Arch  
brew install rustup # macOS  
sudo pkg install rustup # FreeBSD
```

2. curl から

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

Windows10, 11

Windows で Rust 用の開発環境を設定する - Microsoft learn

<https://learn.microsoft.com/ja-jp/windows/dev-environment/rust/setup>

- Visual Studio のインストールが必須

オンラインエディタ

Rust Playground

<https://play.rust-lang.org/?version=stable&mode=debug&edition=2021>

便利なツールをインストール

```
rustup default stable
```

- toolchain の stable 版（一番安定しているやつ）をインストール
 - よく使うのは stable 版, nightly 版

component とは

開発を便利にする公式ツール

- clippy: リンター
- rust-analyzer: LSP. テキストエディタで自動補完やセーブ時にフォーマットできたりする
- rust-docs: 公式ドキュメントのコピー
- rustfmt: フォーマッタ

`rustup component add HOEHOGE` で追加インストール可能

プロジェクトの作成

作成方法

1. プロジェクトを作成したいディレクトリで

```
cargo new hogehoge
```

2. 既にあるディレクトリをプロジェクトディレクトリにする

```
cargo init
```

実行方法

```
# 実行  
cargo run  
# 最適化をかけて実行  
cargo run --release  
# ビルド(プログラムは実行されない)  
cargo build  
# 最適化をかけてビルド(プログラムは実行されない)  
cargo build --release
```

コードは `fn main(){}` に書く

- C での `int main(void){}` と同じ

実行すると `Hello, World` と表示される

- まさかの Hello,world を書かなくてもお k

变数

let キーワード

- 変数宣言は「`let` 変数名 = 値;」
 - 変数は不变（イミュータブル）
 - 型推論あり
- ミュータブル(再代入可能)にするには `mut` を追記
- シャドーイング OK
 - 同じ名前の変数を何度も定義すること

```
let a:i64 = 10;
a = 1; // Error!
let mut a: i64 = 10;
a = 20; // OK!

let b = 5; // これでもおｋ

let love = ; // 絵文字もいけるぞ！
```

int (整数型)

- i8, i16, i32, i64, isize
- i のうしろは Bit 数
- size は OS の Bit 数に依存

uint (符号なし整数型)

- u8, u16, u32, u64, u128, usize

float

- f32, f64
- 基本的に f64 を使えばよい

if 式

- if 式を用いて条件分岐できる
 - C,C++のように if のあとの()は不要

```
let a = 50;
if a > 49 {
    println!("over 50");
}else if a < 49{
    println!("under 50");
}else{
    println!("{}",50);
}
```

- `println!()`マクロで標準出力ができる
 - !が付いてたらマクロ
- 変数を出力するには{}を使う
 - `format`

```
println!("文字列のみの出力");
let f = 3.14;
println!("円周率は{}" , f);
```

練習問題

練習問題 1

- 変数 `fz` には符号なし整数が入ります。
- 15 の倍数であれば `fizzbuzz`, `fz` が 3 の倍数であれば `fizz`, 5 の倍数であれば `buzz` と出力してください。
- 以上のどれでもなければ `none` と出力してください。

```
fz = 60 -> fizzbuzz  
fz = 65 -> buzz  
fz = 66 -> fizz
```

```
fn main() {  
    let fz: usize = NUM;  
    // これ以降にコードを書く  
}
```