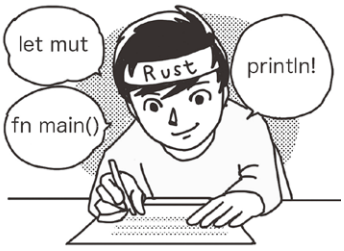
ダウンロード・データあります

Rustプログラミング問題集

ご購入はこちら



新連載 第1回

Rustの基本…標準入出力，文字列処理，コマンドライン引数

中村 仁昭，懸川 岳

本連載ではRustを使った問題と回答形式でRustを学んでいきます．さらに回答のプログラムはRustでよく使う機能を選んでいますので，Rustのプログラム集にもなっています．プログラムを作るときのコード・スニペットにもなるでしょう．

第1回の今回は，主に文法に注目した問題を取り上げます．（編集部）

● 実行環境

rustupでインストールした標準的な環境で動作確認しています．rustupでのインストール手順は公式ページ（https://www.rust-lang.org/ja/tools/install）を参照してください．インストールが難しい場合はPlayground（https://play.rust-lang.org/）で試すことも可能です．

1 構造体

Rustで複素数を表現する構造体を定義し，2つの複素数の和・差・積を計算してください．

● 回答

回答を**リスト1**に示します．構造体は`struct`キーワードを使って定義します．

リスト1　構造体を定義し，2つの複素数の和・差・積を計算する

```
struct Complex<T> {
    x: T,
    y: T
}

fn main() {
    let z1  = Complex{x: 2, y: 5};
    let z2 = Complex{x: 4, y: -7};

    // sum
    let mut sum = Complex{x: 0, y: 0};
    sum.x = z1.x + z2.x;
    sum.y = z1.y + z2.y;
    println!("Re:{}, Im:{}", sum.x, sum.y);

    // difference
    let mut diff = Complex{x: 0, y: 0};
    diff.x = z1.x - z2.x;
    diff.y = z1.y - z2.y;
    println!("Re:{}, Im:{}", diff.x, diff.y);

    // product
    let mut prod = Complex{x: 0, y: 0};
    prod.x = z1.x * z2.x - z1.y * z2.y;
    prod.y = z1.y * z2.x + z1.x * z2.y;
    println!("Re:{}, Im:{}", prod.x, prod.y);
}
```

2 配列への追加

リスト2（a）のコードを修正して，配列に値を追加してください．

● 回答

回答を**リスト2**（b）に示します．Rustでは，宣言した変数はデフォルトで`immutable`なので，値を書き換えようとするとエラーが発生します．

リスト2　配列の操作

```
fn main() {
    let numbers = vec![1, 2, 3];
    numbers.push(4); // ERROR
    println!("numbers: {:?}", numbers);
}
```

（a）配列に値を追加したい（エラーが発生する）

```
fn main() {
    let mut numbers = vec![1, 2, 3];
    numbers.push(4);
    println!("numbers: {:?}", numbers);
}
```

（b）配列に値を追加する

3 スレッド処理

複数のスレッドを生成し，全てのスレッドの処理が完了してから終了してください．

● 回答

回答を**リスト3**に示します．`std::thread::spawn`でthreadの生成，`std::thread::join`でthread