

# Rustを始めるための資料

---

Rustを導入するための資料です。この資料ではWindowsでのRustの環境構築をサポートします。

## Rustを使う意味

---

Rustという言語を使う意味を一応紹介させていただきます。難しい部分があるかもしれませんが流し読みでOKです。

### 安全性

Rustは安全性を言語の仕様として保証するC言語に代わり得るシステムプログラミングに適したコンパイラ言語として存在しています。

何を言っているのかわからないと思います。簡単に言うと開発者の能力に関わらず最低限の安全性を持ったソフトウェアを開発できるよということです。

Rustのコンパイラは私的には「厳しい先生」というイメージがあります。コンパイラとは開発者が書いたプログラミング言語をPCなどが実行する機械語に変換するものを指しますが、Rustのコンパイラははっきり言って厳しいです。

C言語を書くときには軽い気持ちで書いていたことでもRustで実装してみるとコンパイルが失敗したりします。

### 静的型付言語

これについてはC言語と比較してどのような違いがあるのか決定いたします。

#### C言語

```
int plus_five(int value)
{
    return value + 5;
}

int main(void)
{
    // int型の変数、first_valueが宣言されている。
    int first_value = 0;

    int result = plus_five(first_value);
}
```

## Rust

```
fn plus_five(value : i32)->i32
{
    return value + 5
}

fn main()
{
    // ここでのfirst_valueは値の型が決まってない
    let first_value = 0;

    let result = plus_five(first_value);
}
```

これらはどちらもはじめに用意した**first\_value**という変数に関数によって5をプラスするプログラムです。

結果は同じであっても言語的に動きが異なっています。

まずC言語の方ですが、こちらの**first\_value**は宣言時にすでにint型になっています。そしてint型を引数とする**plus\_five**に入力されているためエラーなしに実行されることでしょう。

ですがRustの方をみると、最初の**first\_value**の宣言時には**let**というワードが変数名の前にあるだけで型のようなものは見受けられません。よってこのときの**first\_value**はint型ではないのです。しかしコンパイル時には**plus\_five**関数に入力されていることからRustコンパイラがint型として実行します。

## Rustの環境を構築する

---

RustをWindowsで開発するための環境構築をガイドします。

### VisualStudio

**Rust**ではC++コンパイラを必要するのでVisualStudio2022の更新がされていることを確認してください。

### Rustコンパイラとパッケージ管理システムをインストールする

[Rust公式サイト](#)から64bitの方の.exeファイルを入手して実行してください。

```
C:\Users\81707\Downloads> rustup install

Rust Visual C++ prerequisites

Rust requires the Microsoft C++ build tools for Visual Studio 2013 or
later, but they don't seem to be installed.

You can acquire the build tools by installing Microsoft Visual Studio.

https://visualstudio.microsoft.com/downloads/

Check the box for "Desktop development with C++" which will ensure that the
needed components are installed. If your locale language is not English,
then additionally check the box for English under Language packs.

For more details see:

https://rust-lang.github.io/rustup/installation/windows-msvc.html

Install the C++ build tools before proceeding.

If you will be targeting the GNU ABI or otherwise know what you are
doing then it is fine to continue installation without the build
tools, but otherwise, install the C++ build tools before proceeding.

Continue? (y/N)
```

このような画面が出るので「y」を入力してエンターキーを押してください。

すると色々なログが流れていきます。

```
C:\Users\81707\Downloads> rustup install

The Cargo home directory is located at:

C:\Users\81707\.cargo

This can be modified with the CARGO_HOME environment variable.

The cargo, rustc, rustup and other commands will be added to
Cargo's bin directory, located at:

C:\Users\81707\.cargo\bin

This path will then be added to your PATH environment variable by
modifying the HKEY_CURRENT_USER/Environment/PATH registry key.

You can uninstall at any time with rustup self uninstall and
these changes will be reverted.

Current installation options:

default host triple: x86_64-pc-windows-msvc
default toolchain: stable (default)
profile: default
modify PATH variable: yes

1) Proceed with standard installation (default - just press enter)
2) Customize installation
3) Cancel installation
>
```

そしてこのような画面が出るので「1」を押してまたエンターキーを押してください

```
C:\Users\81707\Downloads>
info: downloading component 'clippy'
info: downloading component 'rust-docs'
 16.0 MiB / 16.0 MiB (100 %) 6.3 MiB/s in 2s ETA: 0s
info: downloading component 'rust-std'
 20.5 MiB / 20.5 MiB (100 %) 5.9 MiB/s in 3s ETA: 0s
info: downloading component 'rustc'
 58.9 MiB / 58.9 MiB (100 %) 6.2 MiB/s in 9s ETA: 0s
info: downloading component 'rustfmt'
info: installing component 'cargo'
info: installing component 'clippy'
info: installing component 'rust-docs'
 16.0 MiB / 16.0 MiB (100 %) 1.2 MiB/s in 17s ETA: 0s
info: installing component 'rust-std'
 20.5 MiB / 20.5 MiB (100 %) 6.1 MiB/s in 3s ETA: 0s
info: installing component 'rustc'
 58.9 MiB / 58.9 MiB (100 %) 7.0 MiB/s in 8s ETA: 0s
info: installing component 'rustfmt'
info: default toolchain set to 'stable-x86_64-pc-windows-msvc'

stable-x86_64-pc-windows-msvc installed - rustc 1.81.0 (eeb90cda1 2024-09-04)

Rust is installed now. Great!

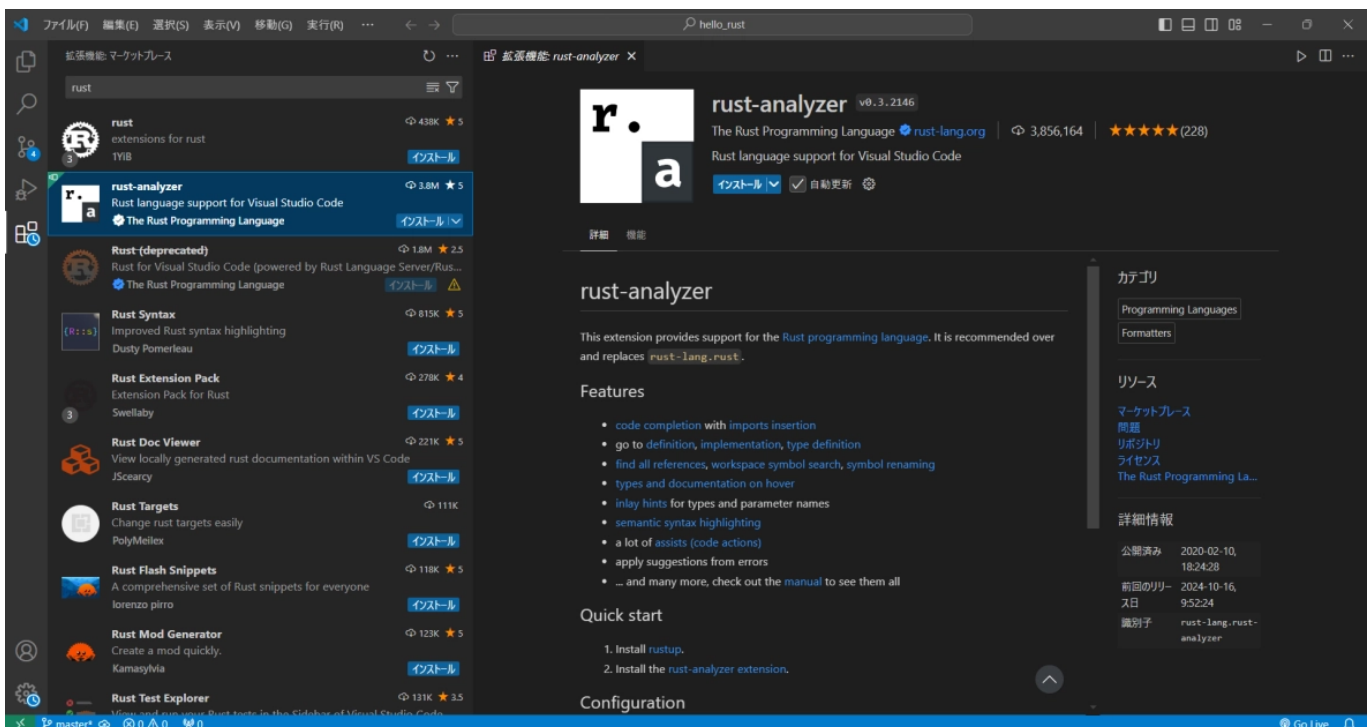
To get started you may need to restart your current shell.
This would reload its PATH environment variable to include
Cargo's bin directory (%USERPROFILE%\cargo\bin).

Press the Enter key to continue.
```

最後にこの画面がでたらエンターキーを押して**Rust**のインストールは完了です。

## Visual Studio CodeでRustを開発する準備

次にRustをVSCodeで開発しやすくなる拡張機能を導入しましょう。以下のように検索して**rust-analyzer**をインストールしてください。これがまじで神なので。



## Rustを動かしてみる

Rustの環境構築が完了したので軽く動作確認をしてみましょう。以下の動作は**Windows Power Shell**で行います。

## パッケージの新規作成

Rustは**Cargo**というシステムによってパッケージを管理しています。このシステムによって使いたいライブラリの指定やパッケージにバージョンをつけることができたり色々便利です。さらに**Cargo**は**GitHub**との相性もとても良いと思います。

Rustを開発するときには基本Cargoというシステムを扱います。以下のコマンドでRustのパッケージを新規作成してください。

```
cargo new [パッケージの名前]
```

今回は下のコマンドによって**hello\_rust**というパッケージを作成しました。

```
PS C:\Users\81707> cargo new hello_rust
  Creating binary (application) `hello_rust` package
note: see more `Cargo.toml` keys and their definitions at https://doc.rust-lang.org/cargo/reference/manifest.html
```

するとhello\_rustというフォルダが作成されていてその中には以下のファイル、フォルダが存在するはずで

```
Cargo.toml // ライブラリなど依存関係を記述するファイル
src // ソースコードを入れるフォルダ
  main.rs // srcフォルダ内にある実際に実行されるファイル「.rs」が拡張子
```

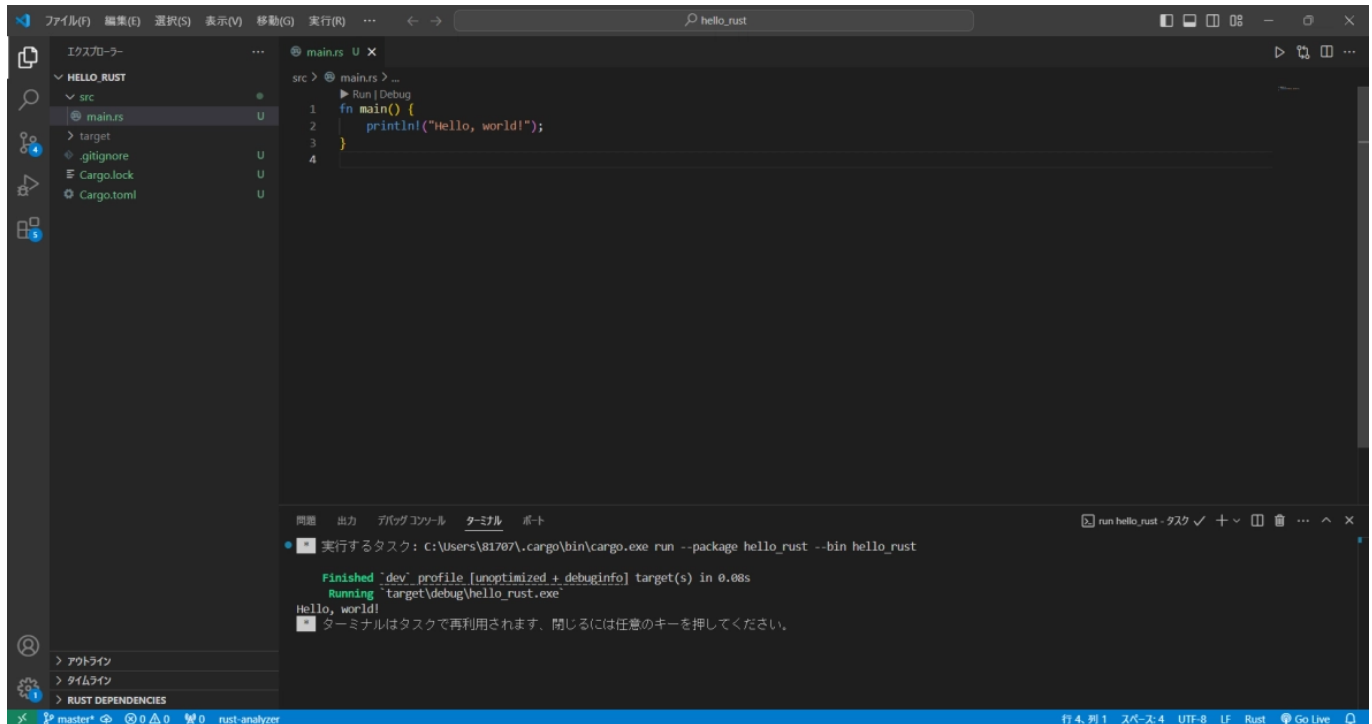
## Visual Studio Codeで編集する

次にパッケージを編集する方法です。以下のコマンドで先程作成したパッケージフォルダ内に移動してください

```
cd hello_rust
```

そしてそのパッケージフォルダをVisualStudioCodeで開きましょう

```
code .
```



まずはじめにCargo.tomlの中身を見ていきましょう。

```
[package]
name = "hello_rust"
version = "0.1.0"
edition = "2021"

[dependencies]
```

- **name** : このパッケージの名前を指定する
- **version** : このパッケージのバージョンを指定する
- **edition** : 特にいじることはない
- **[dependencies]** : この下に使うライブラリを追記する。下は行列計算ライブラリを導入する例

```
[dependencies]
nalgebra = "0.33.0"
```

なんか簡単そうです。

次にsrc/main.rsの中身をみてみましょう。

```
fn main() {
    println!("Hello, world!");
}
```

なにやら「Hello, world!」と表示したようなプログラムですね。適当に書き換えます。

```
fn main() {  
    println!("Hello, 焦土作戦実行！！");  
}
```

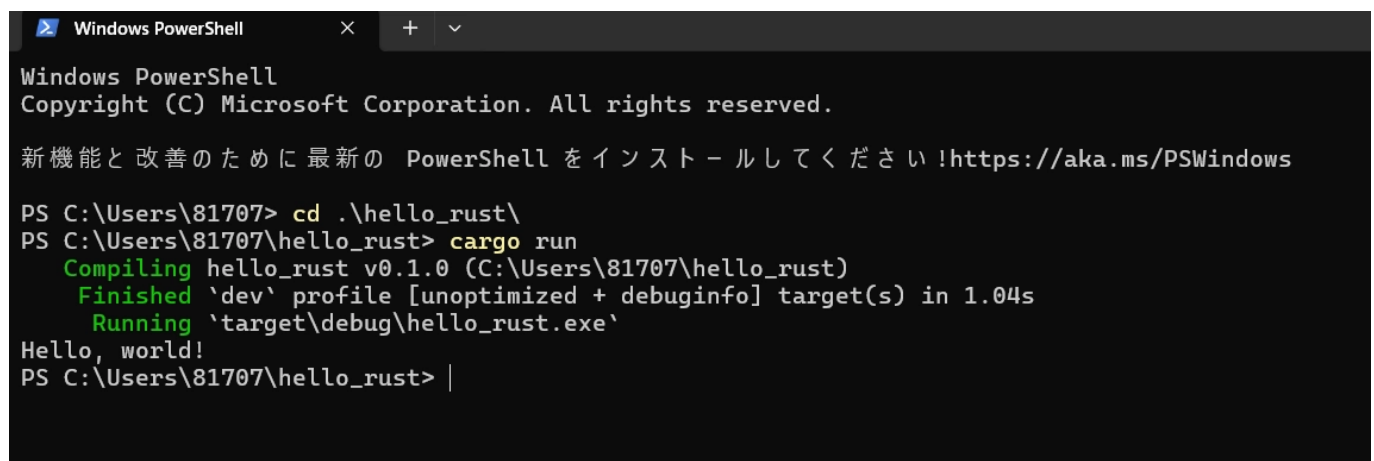
## 焦土作戦実行！！

それではRustのコードを実行してみましょう。もう一度 **Windows PowerShell**を開いてください。

実行する場合には以下のコマンドです。

```
cargo run
```

以下のような実行結果が得られるはず。



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

新機能と改善のために最新の PowerShell をインストールしてください!https://aka.ms/PSWindows

PS C:\Users\81707> cd .\hello_rust\
PS C:\Users\81707\hello_rust> cargo run
   Compiling hello_rust v0.1.0 (C:\Users\81707\hello_rust)
   Finished `dev` profile [unoptimized + debuginfo] target(s) in 1.04s
   Running `target\debug\hello_rust.exe`
Hello, world!
PS C:\Users\81707\hello_rust> |
```