

RUST

Leguaje de Programación

Integrantes

Deilyn Carrascal - 1152480

Anderson Méndez - 1152494

Andrés Manosalva - 1152469

Yorman Navarro - 1152470



Tabla de Contenido

1. Historia
2. Versión actual.
3. Ubicación en Ranking y Utilidad del lenguaje.
4. Conceptos de Clases y Objetos (métodos, propiedades, encapsulamiento)
5. Conceptos de Contenedores / Asociación, Agregación Composición
6. Conceptos de Herencia / Polimorfismo
7. Implementación de un programa en Rust.



Historia

- Solución de problemas personales por Graydon Hoare (2006).
- Graydon Hoare Empezó a diseñar un nuevo lenguaje informático con el que esperaba escribir código pequeño y rápido sin errores de memoria.
- 18 años después, Rust se ha convertido en un lenguaje popular en el planeta, hay 2,8 millones de programadores de Rust.
- La memoria dinámica de un ordenador es como un tablero.
- Hoare pretendía que los programadores no tuviesen que averiguar manualmente en qué parte de la memoria están colocando los datos, Rust lo haría por ellos.
- Hoare ya tenía 10 años de experiencia en software y trabajaba a tiempo completo en Mozilla. Al principio, Rust fue solo un proyecto paralelo.



Historia

- Al ver que este les podría ayudar a crear un motor de navegación mejor, introdujeron a varios ingenieros.



Patrick Walton



Niko Matsakis



Felix Klock



Manish Goregaokar

2009 - Patrocinio de Mozilla.

2010 - Perfección gradual del núcleo.

2015 - Publicación de la primera versión estable de Rust.

2016 - Mozilla lanzó un nuevo motor de navegador llamado Servo.

2020 - Dropbox presentó una nueva versión de su "motor de sincronización"

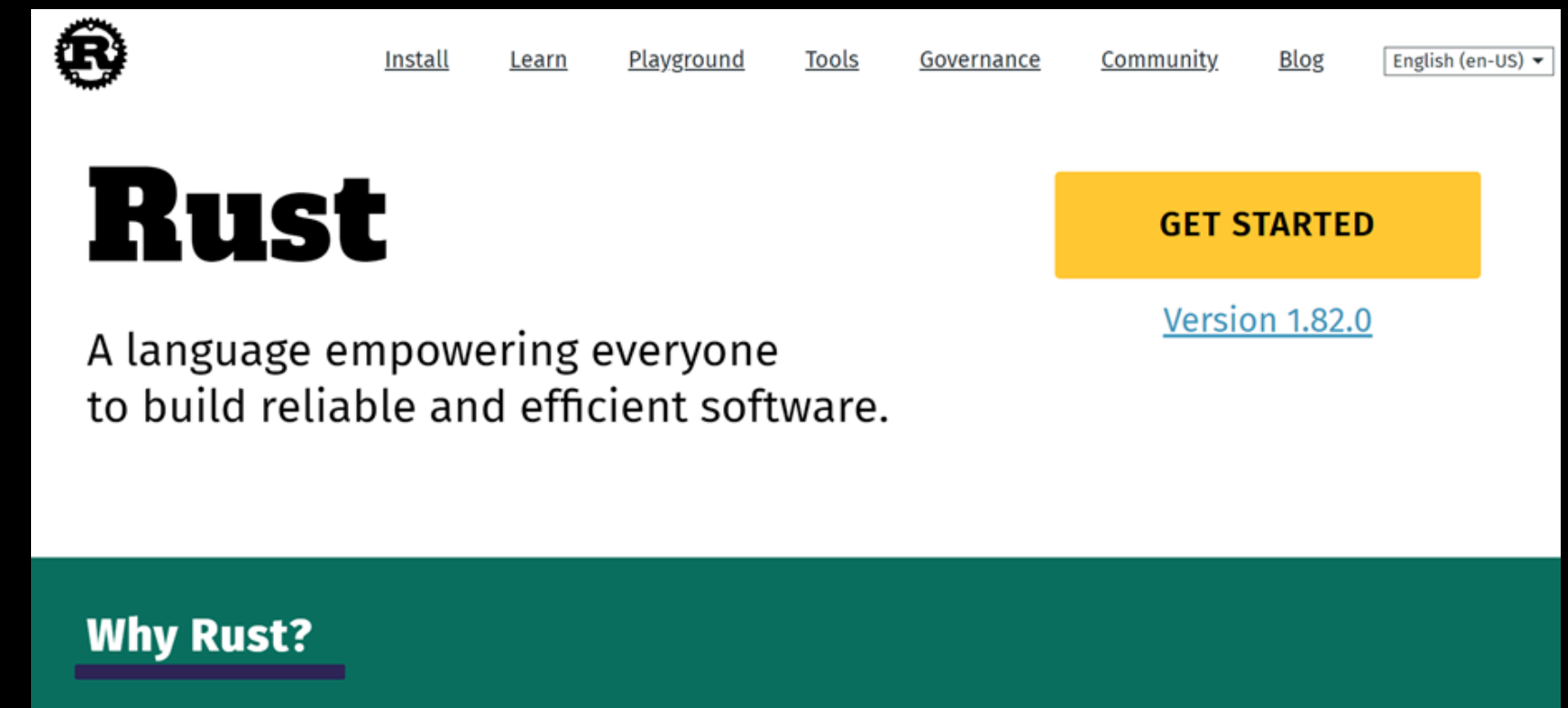
2021 - Crearon una Fundación Rust sin ánimo de lucro.

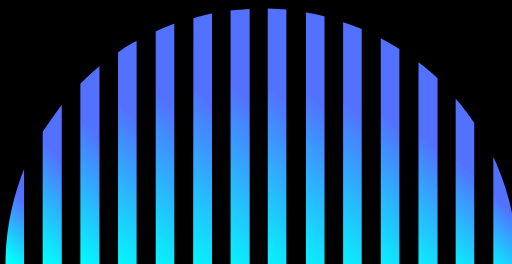

Versión Actual


- Versión: 1.82.0., puesta en funcionamiento el 17 de octubre del 2023.
- La versión beta es la 1.83.0 que se espera y sea publicada el 28 de noviembre de 2024.
- La nightly es la 1.84.0. que se estima para el 9 de enero del 2025.

Para obtener esta o cualquier otra versión de rust ingresar en la pagina oficial del equipo rust

Dar clic Dar clic en "get started" para descargarlo.





 Rust Changelogs

Search

1.84.0

1.83.0

1.82.0

1.81.0

1.80.1

1.80.0

1.79.0

1.78.0

1.77.2

1.77.1

1.77.0

1.76.0

1.75.0

1.74.1

1.74.0

1.73.0

1.82.0

- Released on: *17 October, 2024*
- Branched from master on: *30 August, 2024*

Language

- Don't make statement nonterminals match pattern nonterminals
- Patterns matching empty types can now be omitted in common cases
- Enforce supertrait outlives obligations when using trait impls
- `addr_of!(_mut)!` macros and the newly stabilized `&raw (const|mut)` are now safe to use with all static items
- `size_of_val_raw`: for length 0 this is safe to call
- Reorder trait bound modifiers *after* `for<...>` binder in trait bounds
- Stabilize opaque type precise capturing (RFC 3617)
- Stabilize `&raw const` and `&raw mut` operators (RFC 2582)
- Stabilize unsafe extern blocks (RFC 3484)
- Stabilize nested field access in `offset_of!`
- Do not require `T` to be live when dropping `[T; 0]`
- Stabilize `const` operands in inline assembly
- Stabilize floating-point arithmetic in `const fn`
- Stabilize explicit opt-in to unsafe attributes

Language

Compiler

Libraries

Stabilized APIs

Cargo

Compatibility Notes






















Internal Changes

Portal de todas las versiones estables.

<https://releases.rs/docs/1.84.0/>

Ubicación en el Ranking

En el ranking de lenguajes usados en el mundo podemos ver a rust en el listado del TIOBE.

 About us Knowledge News Coding Standards TIOBE Index Contact						
Products Quality Models Markets Schedule a demo						
Nov 2024	Nov 2023	Change	Programming Language		Ratings	Change
1	1			Python	22.85%	+8.69%
2	3	▲		C++	10.64%	+0.29%
3	4	▲		Java	9.60%	+1.26%
4	2	▼		C	9.01%	-2.76%
5	5			C#	4.98%	-2.67%
6	6			JavaScript	3.71%	+0.50%
7	13	▲▲		Go	2.35%	+1.16%
8	12	▲		Fortran	1.97%	+0.67%
9	8	▼		Visual Basic	1.95%	-0.15%
10	9	▼		SQL	1.94%	+0.05%
11	16	▲		Delphi/Object Pascal	1.48%	+0.33%
12	7	▼▼		PHP	1.47%	-0.82%
13	14	▲		MATLAB	1.28%	+0.12%
14	20	▲		Rust	1.17%	+0.26%
15	17	▲		Swift	1.14%	+0.11%
16	11	▼▼		Scratch	1.11%	-0.21%
17	18	▲		Ruby	1.08%	+0.09%
18	19	▲		R	1.02%	+0.09%
19	10	▼▼		Assembly language	0.97%	-0.39%
20	15	▼▼		Kotlin	0.92%	-0.23%

Utilidad

- Escribir sistemas, como sistemas operativos.
- Desarrollar aplicaciones de bajo nivel, como compiladores o intérpretes.
- Crear herramientas de línea de comandos.
- Desarrollar aplicaciones web.
- Programar dispositivos integrados.
- Desarrollar software para servicios de rendimiento crítico.
- Desarrollar aplicaciones con el modelo cliente-servidor.
- Crear código para coches y otros dispositivos de las empresas automovilísticas y aeroespaciales.

Características

- Seguridad de memoria y en hilos
- Eliminación de muchos tipos de bugs durante la compilación
- Documentación completa
- Compilador accesible con mensajes de error útiles
- Herramientas integradas, como un gestor de paquetes y de proyecto
- Soporte multi-editor con autocompletado e inspecciones de tipos
- Auto-formateador

Conceptos de Rust

Encapsulamiento

Limitar el acceso a las propiedades de un objeto a los elementos que lo necesitan



Métodos

Funciones que permiten a los objetos interactuar con otros objetos o realizar tareas internas



Clases

Son plantillas para crear objetos.

Objetos

Son instancias de una clase creada con datos específicos.



No implementa clases ni herencia.

¿Rust es un lenguaje Orientado a Objetos o Basado en Objetos?

Tour de Rust

[Tabla de Contenidos](#)

Encapsulación Con Métodos

Rust admite el concepto de un *objeto* que es una estructura asociada a algunas funciones (también conocidas como *métodos*).

El primer parámetro de cualquier método debe ser una referencia a la instancia asociada a la llamada de dicho método (por ejemplo, `instanceOfObj.foo()`). Rust utiliza:

- `&self` - para una referencia inmutable a la instancia.
- `&mut self` - para una referencia mutable a la instancia.

Los métodos se definen dentro de un bloque de implementación haciendo uso de `impl`:

```
1 struct SeaCreature {
2     noise: String,
3 }
4
5 impl SeaCreature {
6     fn get_sound(&self) -> &str {
7         &self.noise
8     }
9 }
10
11 fn main() {
12     let creature = SeaCreature {
13         noise: String::from("blub"),
14     };
15     println!("{}", creature.get_sound());
16 }
17
```

Conceptos

Asociación

La asociación representa una relación entre dos objetos, donde ambos pueden existir independientemente. En Rust, esto se modela comúnmente utilizando referencias o valores propios dentro de un struct.

Rust

```
struct Persona {
    nombre: String,
    edad: u32,
}

struct Libro {
    titulo: String,
    autor: Persona, // Asociación: Un libro tiene un autor
}

fn main() {
    let autor = Persona {
        nombre: "Juan Pérez".to_string(),
        edad: 30,
    };

    let libro = Libro {
        titulo: "Mi primera novela".to_string(),
        autor,
    };

    println!("El autor de {} es {}", libro.titulo, libro.autor.nombre);
}
```

Conceptos

Agregación

La agregación es un tipo especial de asociación donde un objeto es parte de otro, pero puede existir independientemente. En Rust, se modela de manera similar a la asociación, pero a menudo se usa una referencia débil (como `Option`) para indicar que el componente puede ser nulo.

Rust

```
struct Motor {
    cilindros: u8,
}

struct Coche {
    motor: Option<Motor>, // Agregación: Un coche puede tener un motor,
}

fn main() {
    let motor = Motor { cilindros: 4 };
    let coche = Coche { motor: Some(motor) };

    // El coche puede existir sin motor:
    let coche_sin_motor = Coche { motor: None };
}
```


Conceptos

Composición

La composición es una relación más fuerte que la agregación, donde un objeto es parte de otro y no puede existir independientemente. En Rust, esto se logra al tomar propiedad de los componentes dentro del struct.

Rust

```
struct Rueda {  
    radio: f64,  
}  
  
struct Coche {  
    ruedas: [Rueda; 4], // Composición: Un coche está compuesto de 4 ruedas  
}  
  
fn main() {  
    let rueda1 = Rueda { radio: 0.3 };  
    let rueda2 = Rueda { radio: 0.3 };  
    let rueda3 = Rueda { radio: 0.3 };  
    let rueda4 = Rueda { radio: 0.3 };  
  
    let coche = Coche {  
        ruedas: [rueda1, rueda2, rueda3, rueda4],  
    };  
}
```

Conceptos

Herencia y Polimorfismo

Aunque Rust no tiene herencia clásica como en lenguajes como C++ o Java, utiliza traits para lograr polimorfismo, un concepto fundamental en la programación orientada a objetos.

Un trait define un conjunto de métodos que un tipo debe implementar. Al implementar un trait, un tipo se compromete a proporcionar una implementación para cada método definido en el trait.

Rust

```
trait Animal {
    fn make_sound(&self);
}

struct Dog {
    name: String,
}

impl Animal for Dog {
    fn make_sound(&self) {
        println!("Woof!");
    }
}

struct Cat {
    name: String,
}

impl Animal for Cat {
    fn make_sound(&self) {
        println!("Meow!");
    }
}

fn main() {
    let dog = Dog { name: "Buddy".to_string() };
    let cat = Cat { name: "Whiskers".to_string() };

    make_animal_sound(&dog);
    make_animal_sound(&cat);
}

fn make_animal_sound(animal: &dyn Animal) {
    animal.make_sound();
}
```

Repositorio en GitHub

Triangulo

The screenshot shows the GitHub interface for a repository named 'RustExpo' owned by 'IngAnderson24'. The repository is public and currently has 0 stars, 1 watcher, and 0 forks. The main content area shows a file named 'Rust.zip' uploaded 39 minutes ago. Below the file list, there is a section for adding a README, with a green button labeled 'Add a README'. The right sidebar contains sections for 'About' (no description provided), 'Activity' (0 stars, 1 watching, 0 forks), 'Releases' (no releases published), and 'Packages' (no packages published).

IngAnderson24 / RustExpo

Code Issues Pull requests Actions Projects 1 Wiki Security Insights Settings

RustExpo Public

main 1 Branch 0 Tags

Go to file

Add file Code

IngAnderson24 Add files via upload 5c713df · 39 minutes ago 1 Commit

Rust.zip Add files via upload 39 minutes ago

README

Add a README

About

No description, website, or topics provided.

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Link al repositorio:
<https://github.com/IngAnderson24/RustExpo.git>

WebGrafías

1) Rust, el lenguaje de programación que ha destronado a C-

- <https://www.technologyreview.es/s/15106/breve-historia-de-rust-el-lenguaje-de-programacion-que-ha-destronado-c>

2) Pagina oficial de rust

- <https://blog.rust-lang.org>

3) Versiones de rust

- <https://releases.rs/docs/1.84.0/>

4) Fundación rust

- <https://foundation.rust-lang.org>

5) Ranking de rust

- <https://www.tiobe.com/tiobe-index/>

6) Rust, ¿orientado a objetos o basado en objetos?

- https://www.reddit.com/r/rust/comments/17vwmom/can_rust_be_described_as_an_objectbased_language/?tl=es-es&rdt=33181



Muchas Gracias

Fin de presentación