Rust Programming Language

By:Submitted

07 BOGUILES, ZYREY APILADO

10 CASUPANAN, JEREMY LEWINS PACIS

22 PANGILINAN, BRYANT CASEY SANTOS

32 ULEP, WARREN DAVE RAMOS

43 POSADAS, AHLEA ROSE FONACIER

9402A 10:00-11:00 TF D515

I. HISTORICAL PERSPECTIVE of RUST

- **ABOUT RUST**

  The Rust programming language started as a personal project by Graydon Hoare with Mozilla supporting the project since it began back in 2009. In 2010 they changed the initial compiler that is written in OCaml (Objective Caml) to selfhosting compiler which is written in rust. In 2011 the programming language successfully compiled itself and it was known as rustC. The LLVM compiler was used by rust as its server. There were a number of releases of rust compiler throughout the years the first release was back in 2012, where they released its pre-alpha version. After three years the finally released version 1.0 which is the stable release for the rust compiler. In January of 2014 Rust was noticed to be a C++ competitor as well as to other following languages. Binstock commented on rust as “widely viewed as a remarkably elegant language”, Even though repeated change between versions slowed its adaption Rust became well liked programming language in the community over the years.

II. Language design goals

This project’s goal is to implement the safety, speed, memory layout and concurrency. The goal is to preserve system integrity, availability and efficiency of this language.

- Safety - Rust is safe because it will not endure a null or dangling pointer nor any undefined behavior.

- Speed- faster compiling, faster running and more type checked codes.

```
fn main() {
}
```

- Memory- Rust memory safety features also the concurrency. Rust will catch the errors. No memory leaks.

The project’s design, will have some details of primitive aspects of the language which are borrowed from other languages from to the same family.

These are:
Ownership- This is one of the Rust’s unique features. The variables are owned the allocated data resources such as stacks and other objects that they are bound to. which will result to that every variable binds must own exactly one data source.

Type System- Rust language will focus on not having a life time of data, not memory management or garbage collection, rust has a built in to support the concurrency. It is complex that there are no data races.

III. Language paradigms

Compiled Paradigm Tends to run codes faster since the compiled paradigm is used to create codes that your computer’s processor can directly execute. It can be a programming language advantage.

Concurrent Paradigm A paradigm which handles Programming with multiple tasks. The programs are designed as a collection of communicating processes that can run in parallel to each other.

Functional Paradigm A simple and clean programming paradigm because of its purely mathematical origin. It is impossible to change one of the parts that form a composite value. Though it is possible to improve upon.

Imperative Paradigm A command based paradigm where a programmer follows certain steps to be performed at machine level. It is much messier and complicated than functional paradigm.

Generic Paradigm A paradigm used to develop generic libraries. It focuses on finding what is common among the similar implementations of the same algorithm, then provides abstractions that will cover a lot of complete implementations.

IV. Language Application Areas

Rust programming language needs to have a firm, broad and wide familiar functions that will fits together to have a better understanding to make it easier to maintain and debug to have a safer and efficient result. Some people are asking questions about the language are, “Can Rust be used for Android/iOS programming?” the answer is yes. Because, there are already a lot of examples that they’re using Rust for Android and iOS. Though it will require a lot of works to configure, Rust functions fine on both Android and iOS. Another question is, “Can Rust program run in a web browser?” as of now, not yet. But there are already some projects ongoing to make Rust compile in the web.

And these are the Areas:

Web-Browser

- **3D Modern Games**
- **Light memory using software**
- **static programming user**
- **database**
- **front-end apps**
- **ACTUAL**

projects that are using Rust language:

- Firefox
  - Firefox is a open source web browser, which was developed by Mozilla Foundation. A company that supported rust that is why Firefox is written in Rust.
- Magic Pocket

- Dropbox’s file storage system.

- Servo
  - Mozilla’s web browser layout engine developed teaming up with the Samsung.

- Cargo
  - Rust’s build automation.

- Redox OS
  - An operating system developed in Rust.

- TiKV
  - A key-value database which is controlled by Rust.

- Piston
  - A game engine

V. Language Environments Integrated Development Environment (IDE)

-A tool used by programmers to create programs
-An IDE can consist of text editor, compiler and a debugger

- **Eclipse** o Eclipse provides development tools for Rust inside the Eclipse IDE

- **IntelliJ Idea** o Rust would be usable on both Windows and Linux and it’s a 1st class IDE using IntelliJ.

VI. Language tools

- Rustfmt (Code Formatting)
- Is a tool for formatting rust codes according to the style guidelines. o RustyTags
- Is a command line tool that creates tags. Tags that are being created when the rust source is supplied by defining the environment variables.
- Racer (Code Completion)
- Is intended to provide Rust code completion for editors and IDEs.

VII. Language libraries

- This is the foundation of the portable rust software, and other rust ecosystems, and also has libraries that either do not exist or incomplete.

VIII. Language frameworks

- Rocket
  - Applications that are being tested by local interface and builds web servers and web applications.
- Iron
  - Is a middleware-oriented that provides a foundation for completing complex applications.

IX. Main language Features & Constructs

A. Language Features

- Memory Safety – This language is made to be memory safe. It doesn't allow dangling pointers, null pointers or even data races.
  - Memory Management – Rust language is not using Automated Garbage Collection system. When managing their memory and their other resources it usually **uses resource acquisition as initialization.**
  - Types and Polymorphism – "Traits" is a mechanism supports by their type system inspired by Haskell Language. This is done by constraint adding to the declarations in the type variable.
  - It has a built in package manager.
  - Pattern matching.
  - It has default immutability. You have to define your variable as **mut** in order for it to be mutable.

B. Language Constructs

- **Integers**
  A number that has no fractional component. Each variant of the integer has explicit size.
    - Let a = 99 ; // Initializes a integer.
- **Floating Points**
  Rust has two primitive floating-point number types, the **f32** is a singleprecision float which is 32 bit in size, and the **f64** has double precision and is 64 bit in its size. On modern CPU's the default type is **f64** it is almost as

fast as the other type though precision is much more reliable.

- let x = 2.0; //for 64 bit

- let y: f32 = 3.0; // for 32 bit

- **Boolean Type**

  The way Rust specifies the Boolean type is by the use of **bool** where it will return two possible values **true** or **false**. The main way to use Boolean values is through the use of the if statement.

  - Let t = true; // initializes Boolean type binding.
  - Let f: bool = false; // with the use of explicit type annotation.

- **Character Type**

  The character type in rust can represent more than just ASCII because it represents a Unicode Scalar Value. The type is also Rust's primitive alphabetic type. This is represented through single quotes as oppose to double quotes to string. o Let c = 'z'; //initializes variable binding for character

- **If-else statement**

  The if-else statement is the same as to other languages. But unlike a lot of them your boolean condition doesn't need to be surrounded by parenthesis and a conditions are followed by a block.

  - Let n = 5;

```
If n < 0 {  print!("{}
   is negative", n);
} else if n > 0 {
      print!("{} is positive, n);
   }
```

References:

(2011). The Rust Programming language (1st ed.). Retrieved February 09, 2018, from https://www.rust-lang.org/en-US/documentation.html

(2017, June 19). 17 Programming Language Paradigm. Retrieved February 09, 2018, from https://mikkegoes.com/programming-language-paradigm-terms-explained/

Overview of the Four Main Programming Paradigms. (n.d.). Retrieved February 09, 2018, from http://people.cs.aau.dk/~normark/prog3-03/html/notes/paradigms_themes-paradigm_-overview-section.html#paradigms_functional-paradigm-overview_title_1

(2011). The Rust Programming language (2nd ed.). Retrieved February 09, 2018, from https://doc.rust-lang.org/book/second-edition/ch03-02-data-types.html

Bullen, L. (n.d.). Eclipse Corrosion: the Eclipse IDE for Rust. Retrieved February 12, 2018, from https://projects.eclipse.org/proposals/eclipse-corrosion-eclipse-ide-rust

The Rust Forge. (n.d.). Retrieved February 12, 2018, from https://forge.rust-lang.org/ides.html

What is Emacs. (n.d). Retrieved February 12, 2018, from http://whatis.techtarget.com/definition/Emacs

IRON. (n.d.). Retrieved February 12, 2018, from http://ironframework.io/

Benitez, S. (n.d.). Testing - Rocket Programming Guide. Retrieved February 12, 2018, from https://rocket.rs/guide/testing/

Contributing to Rust (n.d.). Retrieved February 12, 2018, from https://www.rust-lang.org/en-US/contribute-libs.html

Rust Tooling (n.d.). Retrieved February 12, 2018, from https://gist.github.com/nrc/a3bbf6dd1b14ce57f18c

(2018, Feb 18). dan-t/rusty-tags Retrieved February 19, from https://github.com/dan-t/rusty-tags