GFS BLOCKCHAIN · VBI VN BLOCKCHAIN INNOVATION · VIETNAM BLOCKCHAIN UNION · octopus network

MIỄN PHÍ KHÓA ĐÀO TẠO

LẬP TRÌNH BLOCKCHAIN TÙY CHỈNH TRÊN SUBSTRATE OCT + MINIHACKATHON

Nội Dung Khóa Học:

Phần 1:
Làm quen với lập trình rust cơ bản (2-3 tuần)

Phần 2:
Làm quen cơ bản với substrate theo hướng dẫn (1 Tuần)

Phần 3:
Lập trình Blockchain nâng cao (thực chiến với giảng viên 6 tuần)

Phần 4:
Teamup tham gia Minihackathon (2 tuần)

Giải Thưởng: 4.000$ /khóa

Class 1:
Basic of Rust

Nội dung

- Giới thiệu về Rust
- Cài đặt môi trường
- Biến
- Các kiểu dữ liệu
- Luồng điều khiển
- Thực hành & BTVN

Giới thiệu về Rust

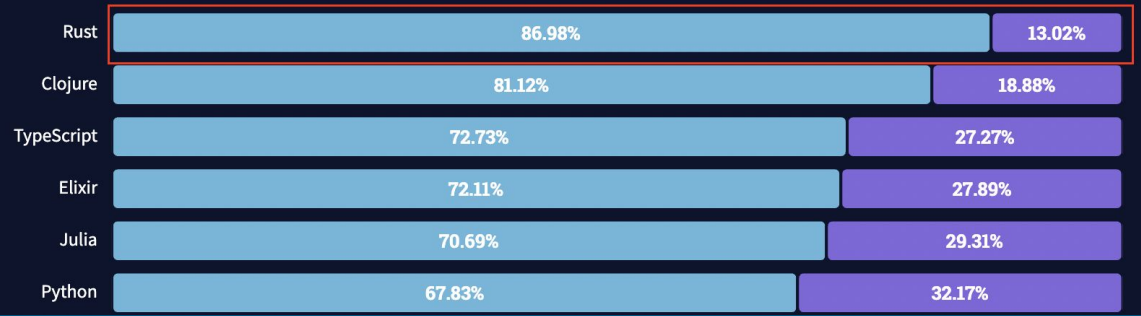
- Là ngôn ngữ lập trình bậc thấp, đa mục đích:
  - Core engine search
  - Network system
  - Kernel, embedded system
  - Blockchain Systems
  - Web applications
- Tốc độ, An toàn, Hiệu năng cao, cơ chế quản lý bộ nhớ an toàn

Programming, scripting, and markup languages

For the sixth-year, Rust is the most loved language, while Python is the most wanted language for its fifth-year.

Loved vs. Dreaded | Want

82,914 responses



https://insights.stackoverflow.com/survey/2021#overview

Cài đặt Rust

- Cài đặt Rustup

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

  Rustup: cài đặt/quản lý các phiên bản rust & các công cụ hỗ trợ khác

    - Rustc: Trình biên dịch Rust
    - Rustup: Bộ cài đặt toolchain (để build ra file thực thi trên nhiều nền tảng khác nhau: Linux, windows, macos, android, embedded devices.
    - Cargo: quản lý project : tạo, run, build project và quản lý các gói thư viện,...
- Công cụ lập trình Rust
  - IDE: Visual Studio Code
  - Plugin: rust-analyzer, ErrorLen
    - Auto completion
    - Kiểm tra và gợi ý lỗi.

Tạo project mới: Hello world

- cargo new helloworld
- cargo run
- cargo build
- cargo build --release

Biến

```
let x = 5;

let user_name = “Tom”;
```

- Định dạng tên biến kiểu `snake_case`
- Mặc định là kiểu immutable - không thay đổi giá trị
- “mut” : mutable thay đổi giá trị của biến
  - let mut count = 0;

const - hằng số

```
const PI: f32 = 3.14;
```

- Định nghĩa HẰNG SỐ và không được thay đổi giá trị
- Khai báo: kiểu dữ liệu và giá trị
- Tồn tại vòng đời của chương trình

Biến immutable và const khác nhau ở điểm nào ?

Biến immutable & const

Biến immutable	**const**
Khi khai báo không bắt buộc phải khai báo kiểu dữ liệu và giá trị. Khi khai báo có thể là giá trị, hoặc kết quả của 1 hàm.	Phải khai báo kiểu dữ liệu + giá trị
Tồn tại trong phạm vị khai báo (hàm, block)	Tồn tại theo vòng đời của chương trình
snake_case	SNAKE_CASE

Biến shadowing

let x = 5;

let x = “Tom”;

- Rust cho phép khai báo lại biến mới có cùng tên nhưng có thể có kiểu dữ liệu khác.
- **Biến shadowing & Biến mutable khác nhau ở điểm nào ?**

So sánh Biến shadowing & Biến mutable

Giống nhau :

- Đều có thể làm thay đổi giá trị của 1 biến đã tồn tại

Khác nhau:

Biến shadowing	**Biến mutable**
Tạo ra biến mới	Vẫn là biến cũ
Có thể thay đổi kiểu dữ liệu của biến	Không thay đổi kiểu dữ liệu

Kiểu dữ liệu

2 loại kiểu dữ liệu

- Scalar : lưu trữ **đơn** giá trị
  - Integer, Float, char, bool
- Compound: lưu trữ **đa** giá trị
  - Array, Tuple, Slice

Scalar - Integer

- let x = 3; // mặc định là kiểu i32
- let y = 10i16; // khai báo kiểu dữ liệu i16
- let y:i8 = 2;

- Số nguyên có dấu (+/-): i8, i16, i32, i64, i128, isize
  - isize: phụ thuộc vào kiến trúc máy tính = i32 hoặc i64.
  - Vùng giá trị : $-(2^{n-1})$ to $2^{n-1}-1$

- Số nguyên không dấu (+): u8, u16, u32, u64, u128, usize
  - usize: phụ thuộc vào kiến trúc máy tính = u32 hoặc u64.
  - Vùng giá trị: 0 to 2^n-1

Scalar - Float (Dấu phẩy động)

- Có 2 kiểu: f32 và f64(Mặc định)
- Ví dụ : let x = 3.2; // kiểu f64

  let y = 3.2f32

Scalar - Char, bool

- Ký tự char : lưu trữ 1 ký tự, hỗ trợ ký tự Unicode

```
let x = 'x';

let y: char = '😎';
```

- Kiểu bool: true/false

Compound - Array

- Array: là kiểu dữ liệu mảng, có cố định số lượng phần tử cùng kiểu dữ liệu

  let arr : [i32; 3] = [1, 2, 3];

- Mặc định là immutable => không thay đổi giá trị của phần tử
- println!(“{:?}”, arr);
- println!(“{}”, arr[0]);
- Destrusture array: let [a, b, c] = arr;

Compound - Tuple

- let tup : (i32, u64, &str) = (-10, 5, “a”);
- Lưu trữ tập hợp các giá trị
  - có kích thước cố định,
  - có kiểu dữ liệu khác nhau
- println!(“{:?}”, tup);
- println!(“{}”, tup.0);
- destructure tuple:
  - let (a, b, c) = tup;

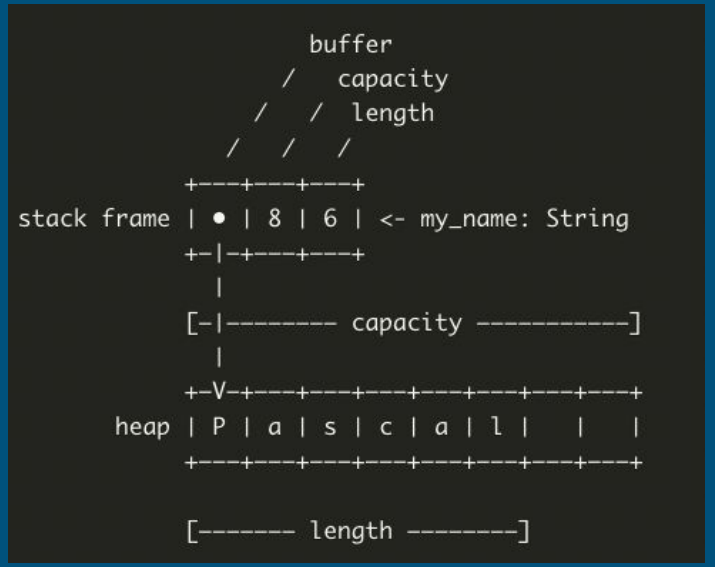
Slice

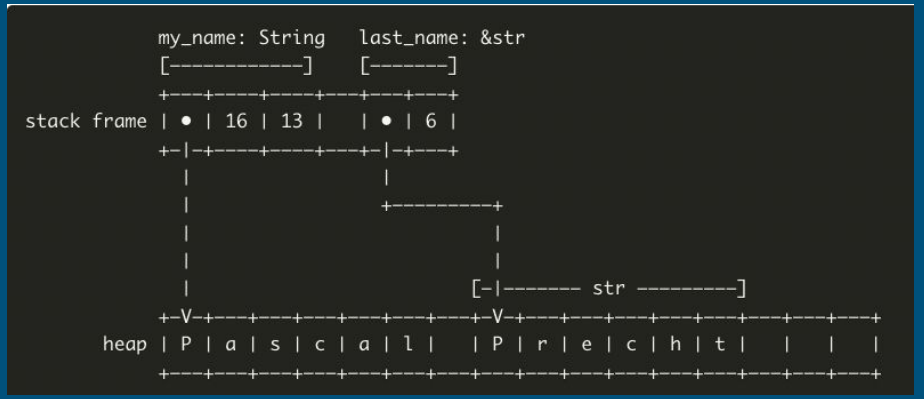
- Là 1 tham chiếu đến 1 phần hoặc toàn bộ vùng nhớ của đối tượng khác.
- Mặc định chỉ có quyền đọc - immutable.
- Ví dụ :

```
let a: [i32; 4] = [1, 2, 3, 4]; // Parent Array

let b: &[i32] = &a; // Slicing whole array
let c = &a[0..4]; // From 0th position to 4th(excluding)
let d = &a[..]; // Slicing whole array

let e = &a[1..3]; // [2, 3]
let f = &a[1..]; // [2, 3, 4]
let g = &a[..3]; // [1, 2, 3]
```

String với &str

- String
  - Lưu trữ 1 chuỗi các ký tự, có kích thước động
  - let mut name = String::from(“Tom”);
  - Hỗ trợ hàm thay đổi nội dung của chuỗi:
    - name.push_str(“a”);
- &str: String slice
  - let str1 = “Hello world”;
  - let str2 = &name[..];
  - Chỉ có quyền đọc





Control Flow

- Điều kiện

  if else

  match
- Vòng lặp

  loop

  for

  while

Thực hành

- Đảo 1 chuỗi str Slice

Bài tập

- Bài tập 1: Cho 2 mảng có các phần tử là số nguyên dương, kiểm tra mảng này có phải là mảng con của mảng kia không ?(yêu cầu đúng thứ tự của các phần tử)

  let org_arr = [1, 2,3,5,6,8, 10, 11];

  let sub_arr = [6,8,10];

Bài tập 2

Cho 1 chuỗi ký tự, nhập 1 ký tự từ bàn phím trả về số lần xuất hiện của từ đó trong chuỗi đã cho, và chuỗi không chứa ký tự nhập từ bàn phím. Lưu ý: khong phân biệt viết hoa, viết thường

Ví dụ: let input = “adbcdaDd”.

- Nhập s = ‘a’ => in ra kết quả : 2, “dbcdDd”
- Nhập s = ‘d’ => in ra kết quả : 4, “abca”