Rust | A Programming language

ارائه دهندگان: نوید رزمان و دانیال عزیزآمان

استاد مربوطه: دکتر صادق سلیمانی

تاریخ ارائه: ۱۳۹۸/۳/۱۳

دانشگاه کردستان

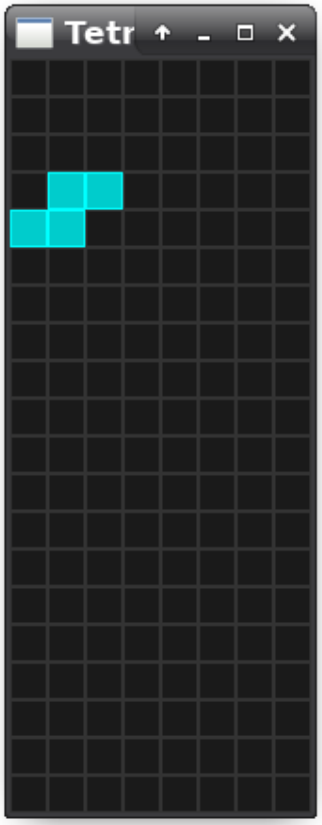
رئوس مطالب

- مقدمه
- معرفی
- کاربرد
- جزئیات برنامه‌نویسی با Rust
- مقایسه با سایر زبان‌ها
- کامپایل و اجرای نمونه برنامه

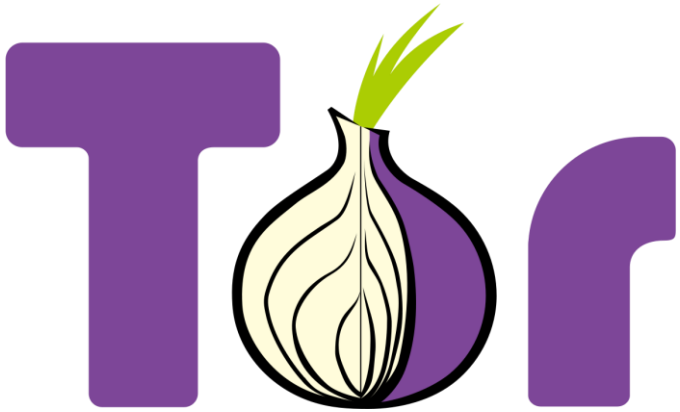
- آغاز در سال ۲۰۰۶ توسط آقای Graydon Hoare
- مطرح شدن در شرکت موزیلا در سال ۲۰۰۹
- کامپایلر اولیه با زبان OCaml
- انتشار نسخه اولیه آن در سال ۲۰۱۱
- نسخه پایدار در سال ۲۰۱۵

- زبانی که توسط خود کامپایل می‌شود (Bootstrapping)
- سیستم مدیریت حافظه نو
- تمرکز بر روی امنیت و برنامه نویسی Concurrent
- محبوب ترین زبان طبق نظرسنجی سالانه Stack Overflow
- مناسب برای پردازش موازی در اینترنت اشیا

- موتور مرورگر Servo
  - ❖ افزایش سرعت تا دو برابر
- Grin
  - ❖ ارز دیجیتال



- Piston
  - ❖ یک موتور بازی‌سازی ماژولار
- Tock
  - ❖ سیستم‌عامل مناسب برای اینترنت اشیا



- Tor
  - ❖ یک شبکه برای پنهان سازی هویت در فضای مجازی

- Stratis
  - ❖ سیستم مدیریت فایل در لینوکس

- exa

❖ ابزار خط فرمان

❖ جایگزین مدرن ls

- Wargroove

❖ یک بازی ویدیویی

جزئیات برنامه نویسی RUST

- داده
- عملیات اصلی
- کنترل ترتیب
- کنترل داده
- مدیریت حافظه
- محیط عملیاتی

داده (اولیه)

- اسکالر (عدد صحیح)

» Attribute: i8 , i16 , i32 , i64 , u8 , u16 , u32 , u64

» Value: arch -> isize , usiz

» Operation: + , - , * , / , % , = , < , <= ,

› let mut x:i8 = 256;

» Storage representation

» Algorithm

Number literals	Example
Decimal	`98_222`
Hex	`0xff`
Octal	`0o77`
Binary	`0b1111_0000`
Byte (`u8` only)	`b'A'`

داده (اولیه)

- عدد اعشاری (Floating point)

» Attribute: f32 , f64

» Value:

» Operation: + , - , * , / , = , < , <= , == , !=

› let y:f32 = 3.14

» Storage representation

» Algorithm

```
fn main() {
    let x = 2.0; // f64

    let y: f32 = 3.0; // f32
}
```

داده (اولیه)

- بول (bool)

» Attribute: bool

» Value: true , false

» Operation: && , || , & , | , ! , == , != , =

› let z:bool = false;

» Storage representation

» Algorithm

داده (اولیه)

- کاراکتر (char)

» Attribute: char

» Value:

› U+000 – U+D7FF (Exclusive)

› U+E000 – U+10FFFF (NonExclusive – chinese)

» Operation: = is_digit() , is_alphabetic() ,

is_alphanumeric(),..

› let ch:char = ’☺’;

» Storage representation → 1 to 4 Byte (unicode)

» Algorithm → hardware

داده (اولیه)

Enumeration -

» Attribute: enum

» Value:

» Operation: = , :: (accessing) , successor , predecessor , comparison , …

› Enum IpVersion { v4, v6, }

› let four = IpVersion::V4

داده (ساختار یافته)

- آرایه

» let a = [1,2,3,4,5];

» Accessing and Selection: A[0] = 2;

» Storage representation → sequential

» Algorithm → software

داده (ساختار یافته)

- تاپل (Tuple)

let tup: (i32, f64, u8, char) = (-12, 3.14 ,2, ’☺’);

» Accessing and Selection:

tup.0 → -12

» let tup = (500, 6.4, 1);

let (x, y, z) = tup;

عملیات اصلی

» & → Borrow

» : → loop label , variable type assign

» :: → object definition

» → → function return

» . → member access

» .. → subrange

» … → subrange (includes the higher bound

کنترل ترتیب

- If expression

```
1   fn main() {
2       let number = 3;
3       if number < 5 {
4           println!("condition was true");
5       }
6       else {
7           println!("condition was false");
8       }
9   }
10
```

کنترل ترتیب

- Loop expression

```
1  fn main(){
2      loop{
3          print!("loop for ever !");
4      }
5  }
6
```

کنترل ترتیب

- while expression

```
fn main() {
    let mut number = 3;
    while number != 0 {
        println!("{}!" , number);
        number-=1;
    }
}
```

کنترل ترتیب

- for expression

```
fn main() {
    for number in (1..4).rev() {
        println!("{}!", number);
    }
println!("LIFTOFF!!!"); }
```

کنترل ترتیب

Match expression -

```
enum Coin { Penny, Nickel, Dime, Quarter, }
fn value_in_cents(coin: Coin) -> u32 {
    match coin {
        Coin::Penny => 1,
        Coin::Nickel => 5,
        Coin::Dime => 10,
        Coin::Quarter => 25,
        _ => 0,
    }
}
```

کنترل ترتیب

- کنترل خطا

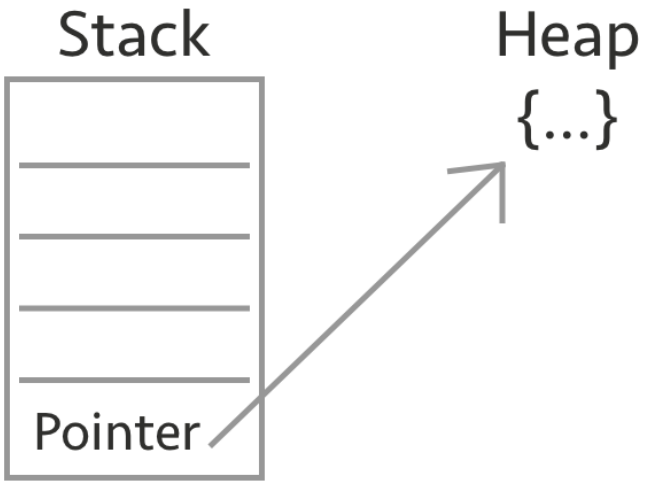
```
io::stdin().read_line(&mut guess).expect("Failed to read line");

let guess_num: u32 = match guess.trim().parse() {
    Ok(num) => num,
    Err(_) => {
        println!("you idiot input A NUMBER !");
        continue;
    }
};
```

کنترل داده

```rust
fn main() {
    let mut s = String::from("hello");

    change(&mut s);

    println!("{}", s);
}

fn change(some_string: &mut String) {
    some_string.push_str(", world");
}
```

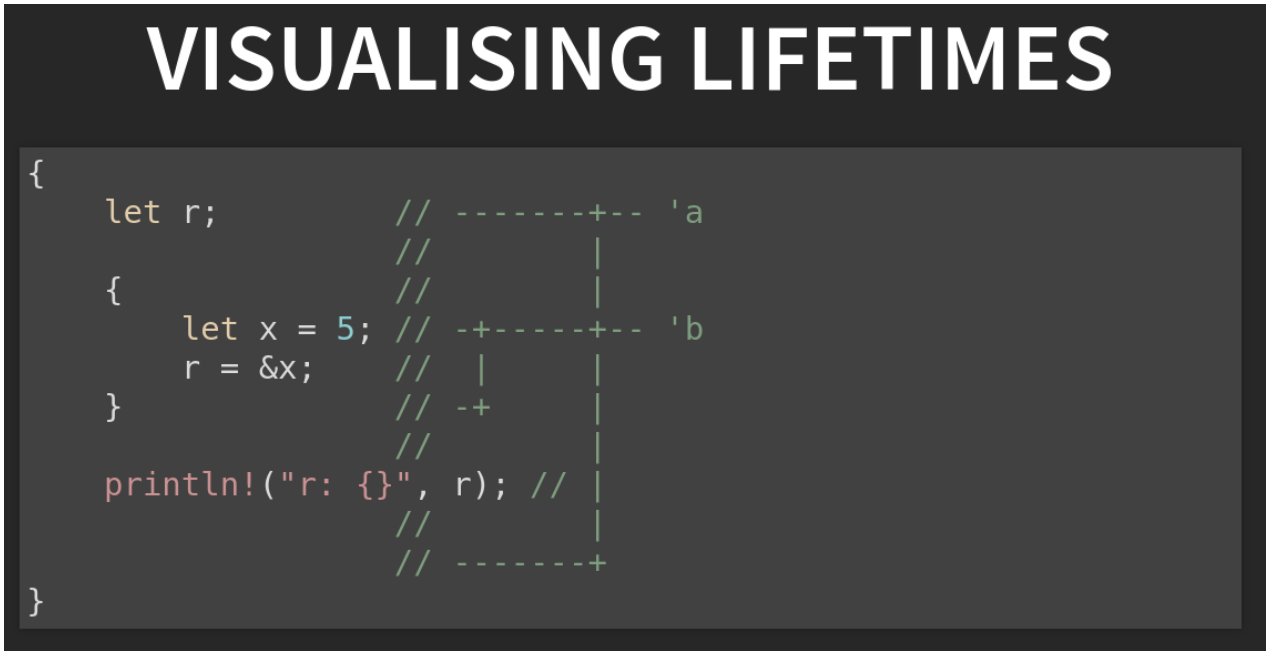
مدیریت حافظه

resource acquisition is initialization (RAII)
+ reference counting

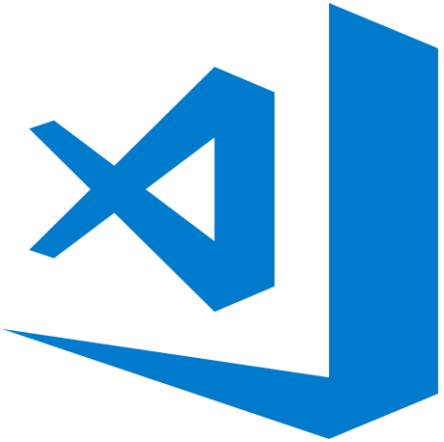


مدیریت حافظه

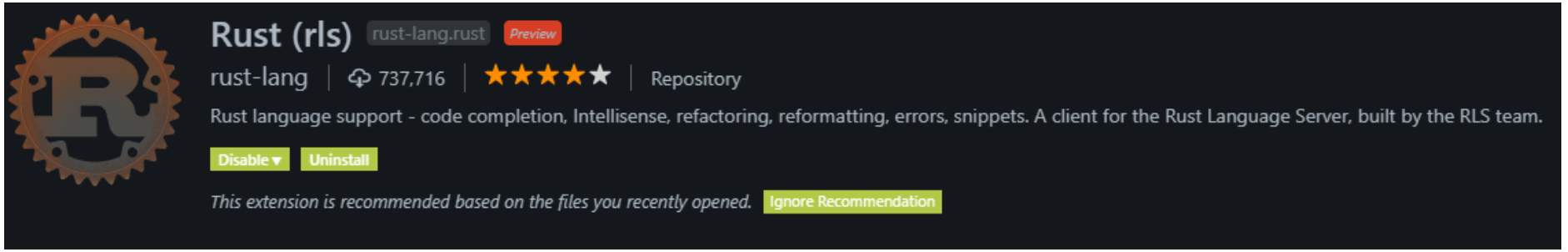
- Dealing with dangling reference example

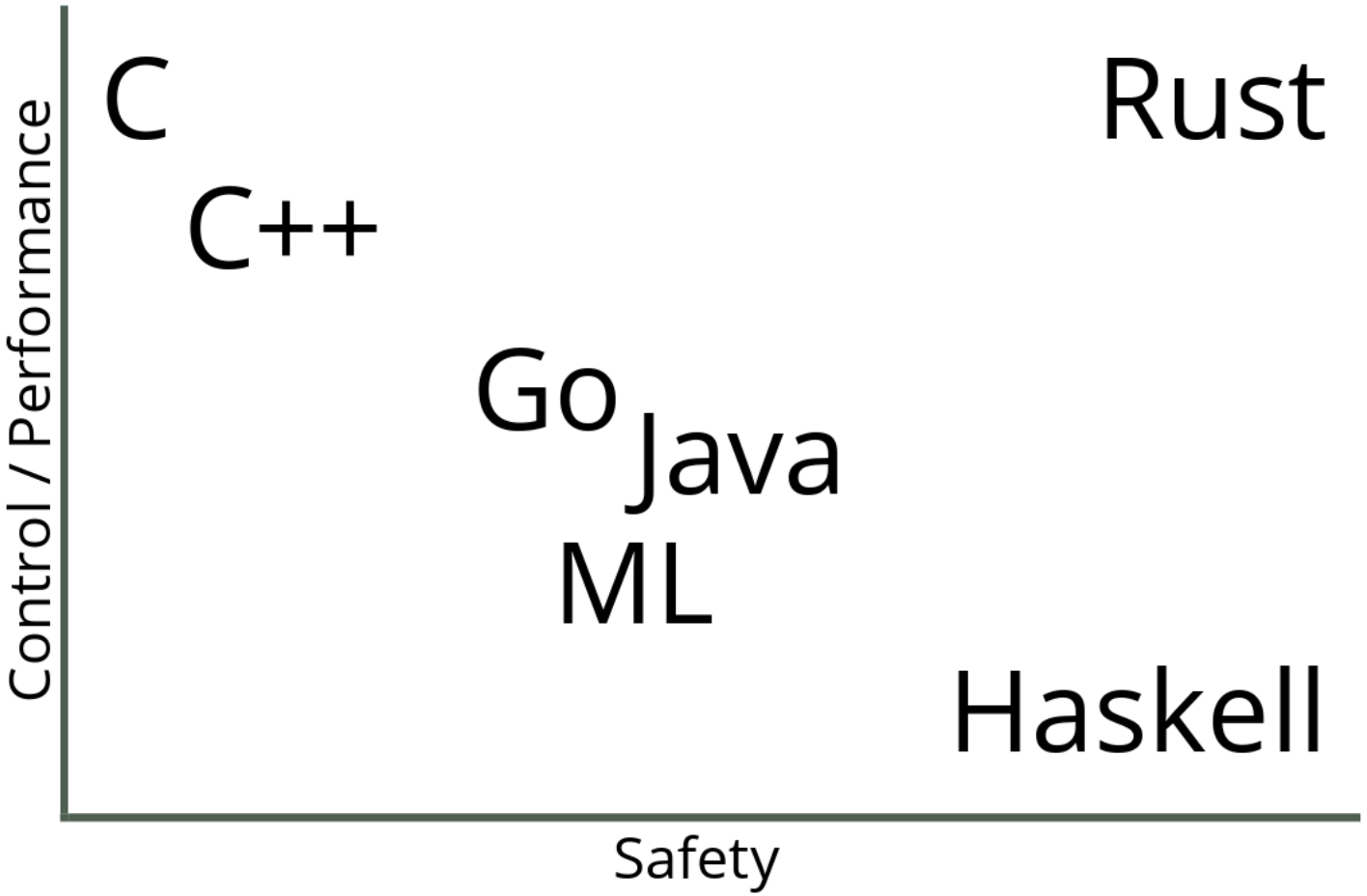


محیط عملیاتی



VS Code →
Rust extension




C
C++
Go
Java
ML
Rust
Haskell
Control / Performance
Safety

بازی حدس عدد !

با تشکر از توجه شما