RUST
Programming language

Lucas Castro (PED) MC346 2021.2
lucas.castro@ic.unicamp.br

PED CLASSES

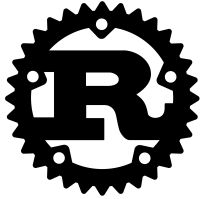
01 RUST LANGUAGE
09/11

02 GOLANG
11/11

03 CLOJURE
16/11

04 SCALA
18/11

Rust applications



Web Assembly	System programming	Embedded systems

RUST TIMELINE

- Graydon Hoare - 2006
  - Mozilla
- Brendan Eich
  - JavaScript
    - NPM
  - Mozilla

- 2010 - Releasing
  - Mozilla Research Labs

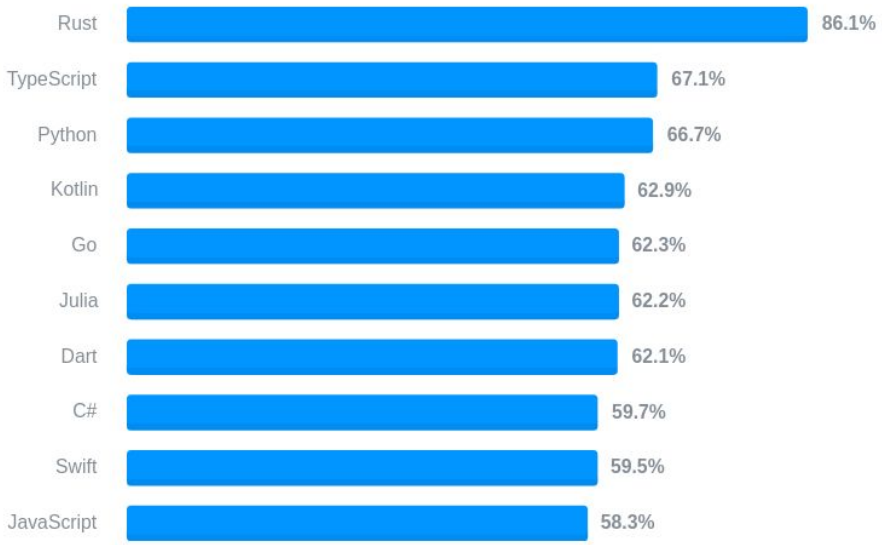
- 2014 - Public

RUST FEATURES

NO
GARBAGE
COLLECTOR

NO
RUNTIME

YES
OWNERSHIP



% of developers who are developing with the language or technology and have expressed interest in continuing to develop with it

LINK

Energy Efficiency across Programming Languages

How Does Energy, Time, and Memory Relate?

Rui Pereira
HASLab/INESC TEC
Universidade do Minho, Portugal
ruipereira@di.uminho.pt

Marco Couto
HASLab/INESC TEC
Universidade do Minho, Portugal
marco.l.couto@inesctec.pt

Francisco Ribeiro, Rui Rua
HASLab/INESC TEC
Universidade do Minho, Portugal
fribeiro@di.uminho.pt
rrua@di.uminho.pt

Jácome Cunha
NOVA LINCS, DI, FCT
Univ. Nova de Lisboa, Portugal
jacome@fct.unl.pt

João Paulo Fernandes
Release/LISP, CISUC
Universidade de Coimbra, Portugal
jpf@dei.uc.pt

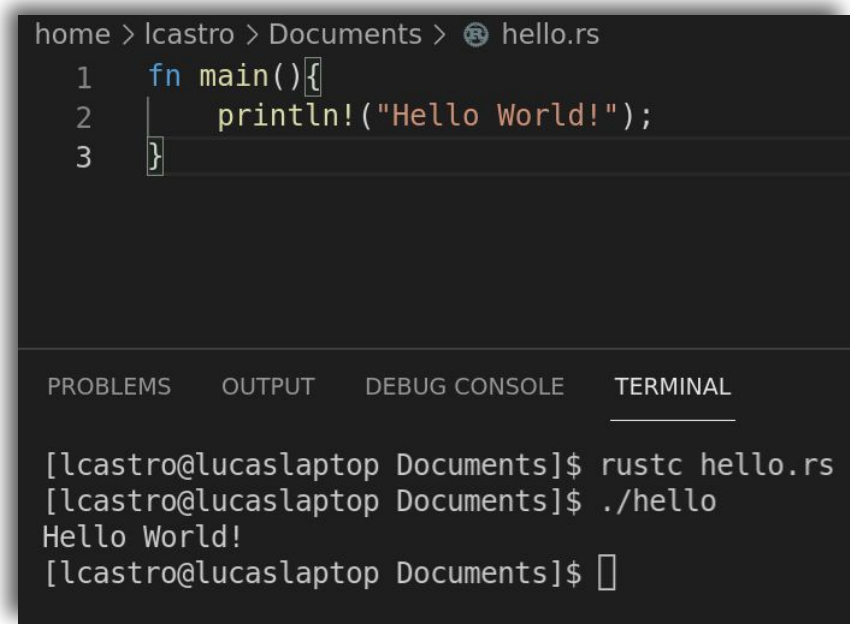
João Saraiva
HASLab/INESC TEC
Universidade do Minho, Portugal
saraiva@di.uminho.pt

	fasta			
	Energy	Time	Ratio	Mb
(c) Rust ⇓9	26.15	931	0.028	16
(c) Fortran ↓6	27.62	1661	0.017	1
(c) C ↑1 ⇓1	27.64	973	0.028	3
(c) C++ ↑1 ⇓2	34.88	1164	0.030	4
(v) Java ↑1 ⇓12	35.86	1249	0.029	41
(c) Swift ⇓9	37.06	1405	0.026	31
(c) Go ↓2	40.45	1838	0.022	4
(c) Ada ↓2 ⇑3	40.45	2765	0.015	3
(c) Ocaml ↓2 ⇓15	40.78	3171	0.013	201
(c) Chapel ↑5 ⇓10	40.88	1379	0.030	53
(v) C# ↑4 ⇓5	45.35	1549	0.029	35
(i) Dart ⇓6	63.61	4787	0.013	49
(i) JavaScript ⇓1	64.84	5098	0.013	30
(c) Pascal ↓1 ⇑13	68.63	5478	0.013	0
(i) TypeScript ↓2 ⇓10	82.72	6909	0.012	271
(v) F# ↑2 ⇑3	93.11	5360	0.017	27
(v) Racket ↓1 ⇑5	120.90	8255	0.015	21
(c) Haskell ↑2 ⇓8	205.52	5728	0.036	446
(v) Lisp ⇓2	231.49	15763	0.015	75
(i) Hack ⇓3	237.70	17203	0.014	120
(i) Lua ⇑18	347.37	24617	0.014	3
(i) PHP ↓1 ⇑13	430.73	29508	0.015	14
(v) Erlang ↑1 ⇑12	477.81	27852	0.017	18
(i) Ruby ↓1 ⇑2	852.30	61216	0.014	104
(i) JRuby ↑1 ⇓2	912.93	49509	0.018	705
(i) Python ↓1 ⇑18	1,061.41	74111	0.014	9
(i) Perl ↑1 ⇑8	2,684.33	61463	0.044	53

Reference

RUST NA SUA MÁQUINA

- Official guide
  - Linux and MAC*
  - $ curl --proto '=https' --tlsv1.2 https://sh.rustup.rs -sSf | sh
  - rustup update
  - code hello.rs
  - rustc hello.rs
  - ./hello



*Windows - Follow the Official guide

RUST - CARGO

- Official documentation
  - cargo new <projectName>
  - cd <projectName>
  - cargo build

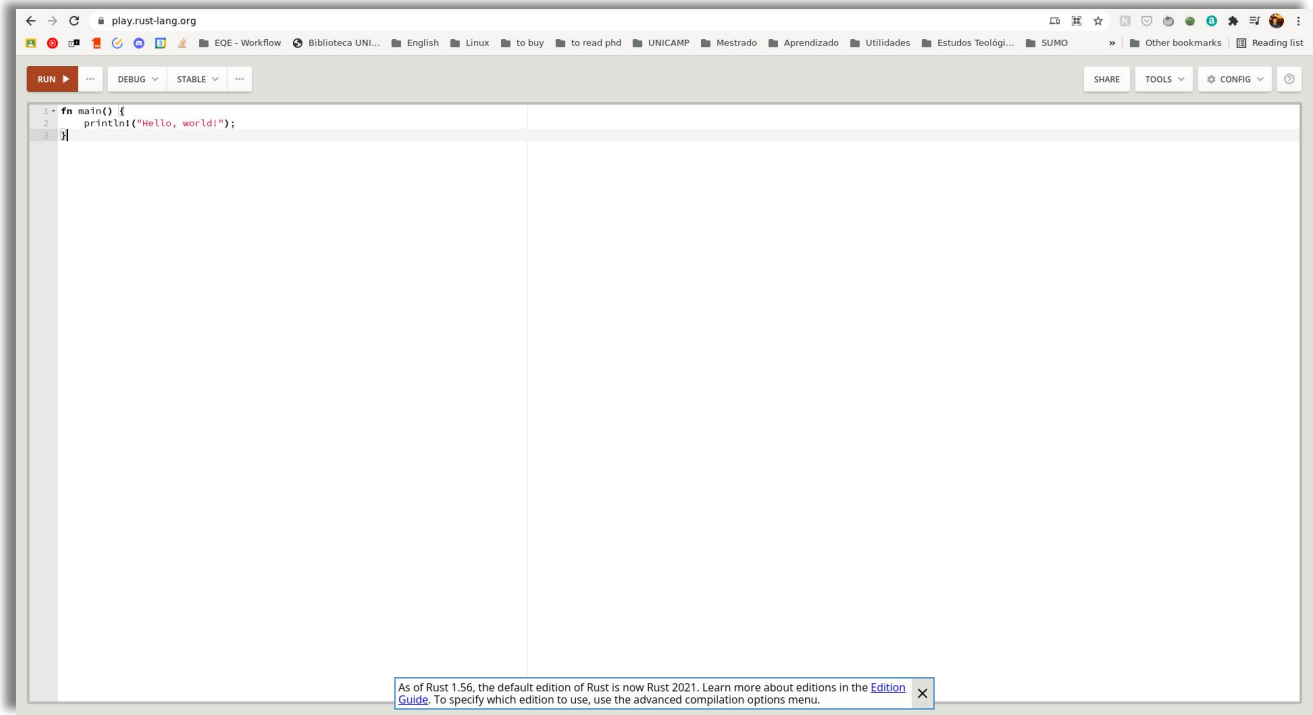
```
1  fn main() {
2      println!("Hello, world!");
3  }
4
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
[lcastro@lucaslaptop person]$ cargo build
    Finished dev [unoptimized + debuginfo] target(s) in 0.02s
[lcastro@lucaslaptop person]$ ls
Cargo.lock  Cargo.toml  src  target
[lcastro@lucaslaptop person]$ ./target/debug/
build/        deps/         examples/     .fingerprint/ incremental/  person
[lcastro@lucaslaptop person]$ ./target/debug/person
Hello, world!
[lcastro@lucaslaptop person]$
```

RUST ONLINE

- https://play.rust-lang.org/



RUST DATATYPES

- Rust datatypes
  - Infere os tipos
  - Tipagem estática

Length	Signed	Unsigned
8-bit	`i8`	`u8`
16-bit	`i16`	`u16`
32-bit	`i32`	`u32`
64-bit	`i64`	`u64`
128-bit	`i128`	`u128`
arch	`isize`	`usize`

Tuple Type

home > lcastro > examples.rs

```rust
fn main() {
    let tup = (500, 6.4, 1);

    let (x, y, z) = tup;

    println!("The value of y is: {}", y);
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
[lcastro@lucaslaptop ~]$ ./examples
The value of y is: 6.4
[lcastro@lucaslaptop ~]$
```

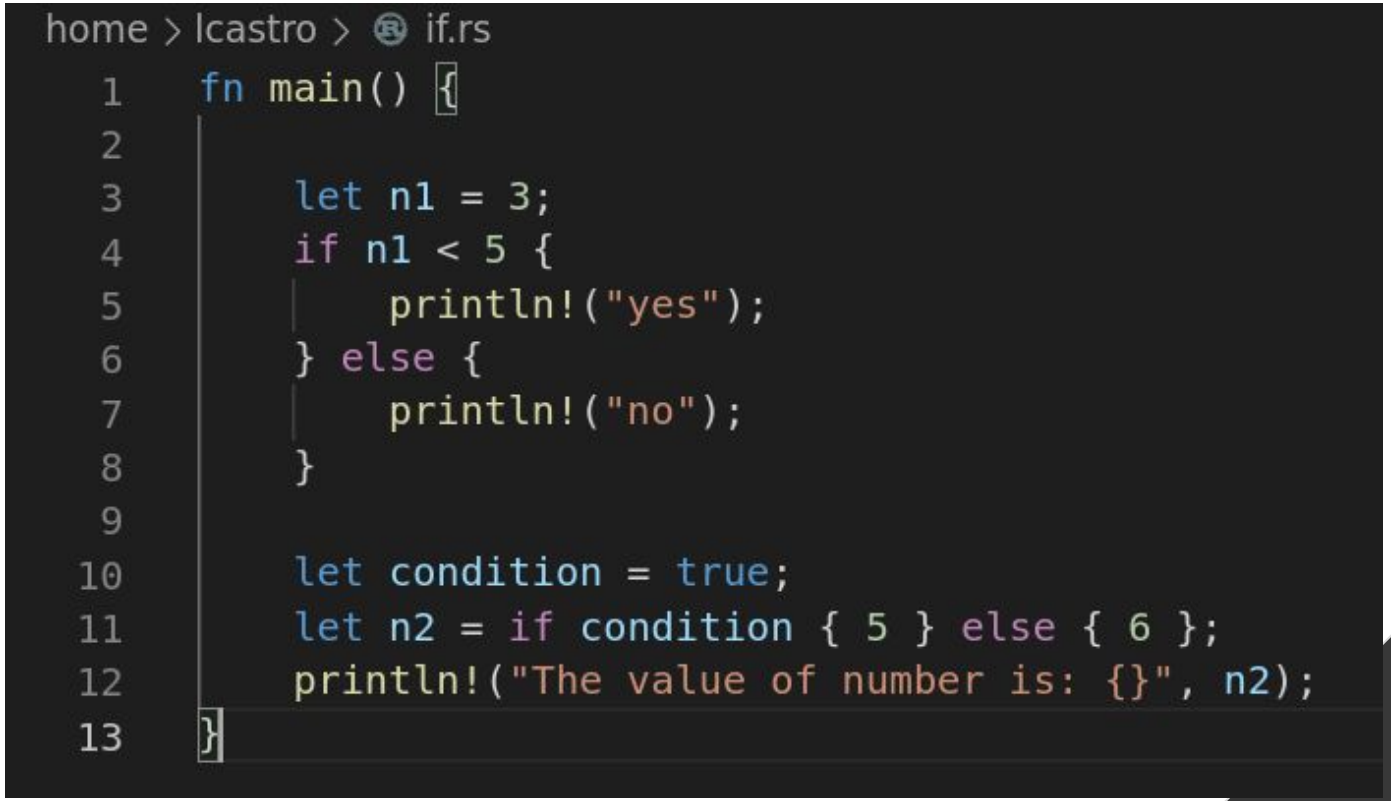
RUST DATATYPES

- Array type

home > lcastro > examples.rs

```rust
fn main() {
    let a = [1, 2, 3, 4, 5];
    let months = ["January", "February", "March", "April", "May", "June", "July",
                  "August", "September", "October", "November", "December"];
    let b: [i32; 5] = [1, 2, 3, 4, 5];

    let first = a[0];
    let second = a[1];

    println!("{}",second);



}
```

RUST IF STATEMENT



RUST IMMUTABLE - SECURITY

home > lcastro > examples.rs

```rust
fn main() {
    let name = "Maria";
    name = "Joana";

    println!("Nome: {}",name);

}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
  |         ^^^^
  |
  = note: `#[warn(unused_assignments)]` on by default
  = help: maybe it is overwritten before being read?

error[E0384]: cannot assign twice to immutable variable `name`
 --> examples.rs:3:5
  |
2 |     let name = "Maria";
  |         ----
  |         |
  |         first assignment to `name`
  |         help: consider making this binding mutable: `mut name`
3 |     name = "Joana";
  |     ^^^^^^^^^^^^^^ cannot assign twice to immutable variable

error: aborting due to previous error; 1 warning emitted
```

home > lcastro > examples.rs

```rust
fn main() {
    let mut name = "Maria";
    name = "Joana";

    println!("Nome: {}",name);

}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
[lcastro@lucaslaptop ~]$ ./examples
Nome: Joana
[lcastro@lucaslaptop ~]$
```

RUST O.O

- Structs
- Impl (key-feature)

home > lcastro > examples.rs

```rust
struct Brand{
    name: String
}
struct Car{
    name: String,
    brand: Brand,
    plate: u64
}



fn main(){
    let fiat = Brand {name: "Fiat".to_string()};
    let uno = Car {name: String::from("Uno"), brand: fiat, plate: 999222};

    println!("Eu tenho um {} - placa: {}",uno.name,uno.plate);

}

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
[lcastro@lucaslaptop ~]$ ./examples
Eu tenho um Uno - placa: 999222
[lcastro@lucaslaptop ~]$
```

RUST O.O

- Structs
- Impl (key-feature)

home > lcastro > examples.rs

```rust
struct Brand{
    name: String
}
struct Car{
    name: String,
    brand: Brand,
    plate: u64
}

impl Car{
    fn saymodel(&self) -> String{
        format!("Eu tenho um {} - placa: {}",self.name,self.plate)
    }
}


fn main(){
    let fiat = Brand {name: "Fiat".to_string()};
    let uno = Car {name: String::from("Uno"), brand: fiat, plate: 999222};

    println!("{}",uno.saymodel());

}
```

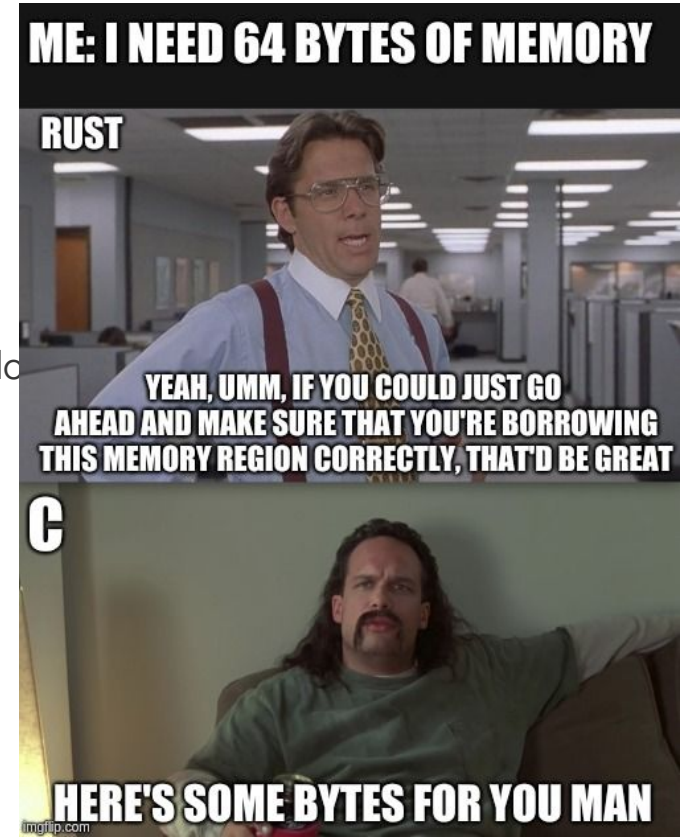
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
[lcastro@lucaslaptop ~]$ ./examples
Eu tenho um Uno - placa: 999222
```

RUST - OWNERSHIP

- Habilita o Rust a não precisar do Garbage Collector
  - Borrowing
  - Slices

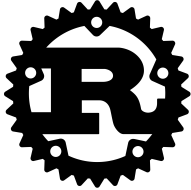
- 3 regras básicas
  - Toda variável possui um dono
  - Só existe um dono por vez
  - Quando o dono sai do escopo do valor, o valor é descartado
    - string.rs
    - Scope.rs
  - Copy, move, &reference



RUST - TRAITS

- Semelhante ao comportamento de interface
- Comportamento compartilhado
- Generic datatypes
    - trait.rs

RUST - THREADS - CHANNEL



home > lcastro > threads.rs

```rust
use std::sync::mpsc;
use std::thread;

fn main() {
    let (tx, rx) = mpsc::channel();

    thread::spawn(move || {
        let val = String::from("hi");
        tx.send(val).unwrap();
    });

    let received = rx.recv().unwrap();
    println!("Got: {}", received);
}
```