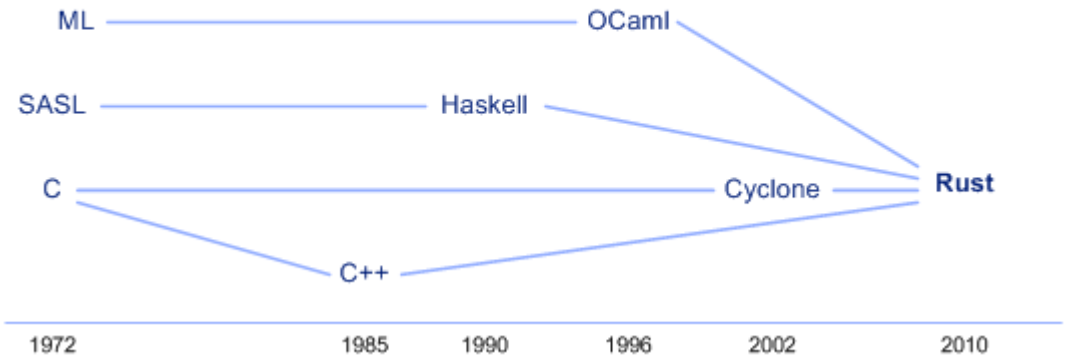
História e Influências

Figure 1. Rust and its family tree



Sobre a linguagem

- Multiparadigma
- Compilada
- Inferência de Tipo
- Tagged Union nativo

```
enum Color {
     Red,
     Blue,
     Green,
     RGB(u32, u32, u32),
     HSV(u32, u32, u32),
     HSL(u32, u32, u32),
     CMY(u32, u32, u32),
     CMYK(u32, u32, u32, u32),
 }
```

```
let i = 20; // Isso será inferido pelo compilador como um inteiro 32bits
let i : i32 = 20; // No momento de criação da variável o programador já definiu o tipo da mesma
```

Expressividade em relação à linguagem C/C++

Ownership

- Regras:
  - Todo valor em Rust possui uma variável que é sua dona.
  - Só pode existir um dono por vez.
  - Quando o dono sai do escopo o valor será liberado da memória.

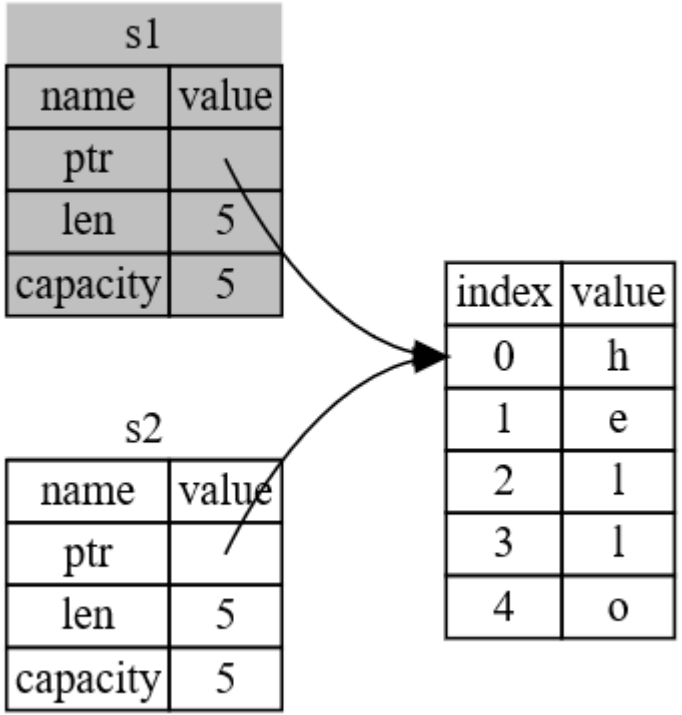
```
{ // A variável s não existe nesse momento
    let s = "hello";   // Agora s é valida

    // Já que s é valida podemos fazer o que quisermos com ela
} // Agora s saiu do escopo e portando não é mais válida.
```

```
    let s1 = String::from("hello"); // Criamos uma String e a dona dela é
s1
    let s2 = s1; //Agora movemos o valor de s1 para s2 e portanto o novo
dono é s2 e s1 não poderá mais ser usada enquanto o ownership não voltar
para ela
```

Expressividade em relação à linguagem C/C++

Ownership

- Regras:
  - Todo valor em Rust possui uma variável que é sua dona.
  - Só pode existir um dono por vez.
  - Quando o dono sai do escopo o valor será liberado da memória.



Expressividade em relação à linguagem C/C++

Pattern Match e Enum

```
enum Color{
  Red,
  Green,
  Blue,
  RgbColor(u8, u8, u8),
  Cmyk{cyan:u8, magenta:u8, yellow:u8, black:u8}
}
fn main(){

  let c:Color = Color::RgbColor(1,100,255);
  match c {
      Color::Red => println!("r"),
      Color::Green => println!("g"),
      Color::Blue => println!("b"),
      Color::RgbColor(0,0,0) => println!("black"),
      Color::RgbColor(r, g, b) => println!("rgb({},{},{})", r, g, b),
      Color::Cmyk{cyan:_,magenta:_,yellow:_,black:255} => println!("black"),
      _ => ()
  }
}
```

```
#include <stdio.h>
union Color {
    enum {
    C_primaria, C_RGB, C_CMYK
    } ctype;

    enum cores_primarias {
        Red,
        Yellow,
        Blue
    }cor_primaria;

    struct RgbColor {
        int r;
        int g;
        int b;
    }RGB;

    struct Cmyk {
    int cyan;
    int magenta;
    int yellow;
    int black;
    }CMYK;

};

int main(void){
    union Color color;
    color.RGB.r = 1;
    color.RGB.g = 100;
    color.RGB.b = 255;
    color.ctype = 1;
    switch (color.ctype)
    {
        case C_primaria:
            switch (color.cor_primaria)
            {
                case Red:
                    printf("RED");
                    break;
                case Yellow:
                    printf("Yellow");
                    break;
                case Blue:
                    printf("Blue");
                    break;
            }
        case C_RGB:
            if(color.RGB.r == 0 && color.RGB.g == 0 && color.RGB.b == 0){
                printf("BLACK");
            }
            else
                printf("%d %d %d",color.RGB.r,color.RGB.g,color.RGB.b);
            break;
        case C_CMYK:
            if(color.CMYK.cyan == 0 && color.CMYK.magenta == 0 && color.CMYK.yellow == 255 && color.CMYK.black == 0){
                printf("YELLOW");
            }
            break;

        default:
            printf("");
            break;
    }
    return 0;
}
```

- Bibliografia e Links
- Site oficial: https://www.rust-lang.org/
- Página com referências oficiais para aprendizado: https://www.rust-lang.org/learn
- Wikipédia: https://en.wikipedia.org/wiki/Rust_(programming_language)
- Compilador online oficial: https://play.rust-lang.org/