Estruturas de controle de fluxo

Estruturas de controle de fluxo

Expressões lógicas

- Expressões cujos operadores são lógicos ou relacionais e cujos operandos são relações ou variáveis ou constantes do tipo lógico

- Operadores lógicos
  - E, OU e NÃO

- Como representamos em Rust?

E: &&

OU: ||

NÃO: !

Estruturas de controle de fluxo

Expressões e operadores aritméticos

Operador	Símbolo
Soma	+
Subtração	-
Multiplicação	*
Divisão	/
Resto (mod)	%

Estruturas de controle de fluxo

Operadores lógicos

Operador	Símbolo
Igual a	==
Diferente de	!=
Maior que	>
Menor que	<
Maior ou igual a	>=
Menor ou igual a	<=

Estruturas de controle de fluxo

Controle de fluxo

```
fn main() {
    let numero = 3;

    if numero < 5 {
        println!(“condição verdadeira!”);
    } else {
        println!("condição falsa...”);
    }
}
```

Estruturas de controle de fluxo

Controle de fluxo

```
fn main() {
    let icone: char = '🐸';

    if icone == '🐶' {
        println!("Cachorro");
    } else if icone == '🐸'  {
        println!("Sapo");
    } else {
        println!("Não conheço esse animal...");
    }
}
```

Estruturas de controle de fluxo

Exercícios

- Crie um programa em Rust que recebe como entrada a idade de um usuário e, se for maior de idade, mostre na tela que ele é maior de idade. Caso contrário, mostre que ele é menor de idade.

- Crie um programa para autenticar um usuário. Solicite o nome de usuário e senha e, em seguida, verifique se o nome de usuário é “admin” e a senha é “@dm1n”. Caso a senha ou o usuário não corresponderem, mostre na tela que o usuário ou a senha estão incorretos. Caso contrário, mostre que o usuário foi autenticado.

Laços de repetição

Laços de repetição

- Em Rust temos três tipos de laços de repetição:
  - `loop`
  - `while`
  - `for`

Laços de repetição

- O `loop` executa infinitamente até que **explicitamente** eu peça para que ele pare

```
fn main() {
    loop {
        println!("Executando infinitamente!!!");
    }
}
```

Laços de repetição

- O `while` executa até que uma condição seja satisfeita

```
fn main() {
    let n: u8 = 0;
    while n < 10 {
        println!("Iteração {}", n);
        n += 1;
    }
}
```

Laços de repetição

- O `for` irá iterar por todos os elementos de uma sequência

```
fn main() {
    let a: [char; 4] = ['a', 'b', 'c', 'd'];

    for ch in a {
        println!("{}", ch);
    }
}
```

Laços de repetição

Exercícios

- Escreva um programa em Rust para mostrar a sequência de Fibonacci até n termos
  - Em seguida, some os termos **pares** da sequência e, ao final, mostre a soma

- Escreva um programa em Rust que calcule a soma dos números ímpares em um intervalo de 0 até 100

- Escreva um programa em Rust que leia números inteiros do teclado. Quando digitado o valor 0, o programa deverá encerrar e mostrar a média dos números fornecidos de entrada.