Rust Basics

CS128 Honors

Slides by Matt Geimer (FA21)
Presented 9/13/2021

Functions

```
int add(int firstNumber, int secondNumber)  {
    return firstNumber + secondNumber;
}
```

A function that returns

```
void doesNothing() {
    System.out.println("...");
}
```

A function that doesn’t return

Functions - In Rust

```
fn add(a: i32, b: i32) -> i32 {
    return a + b;
}
```

For reference:

```
int add(int firstNumber, int secondNumber)  {
    return firstNumber + secondNumber;
}
```

Functions - In Rust

```
fn add(a: i32, b: i32) -> i32 {
    a + b;
}
```

For reference:

```
int add(int firstNumber, int secondNumber)  {
    return firstNumber + secondNumber;
}
```

Functions

```rust
fn main() {
    // function call the doesn't return anything
    hello_world_v2();

    let n1 = 120;
    let n2 = 8;

    // two different ways of implementing the same function
    let result1 = add_v1(n1, n2);
    let result2 = add_v2(n1, n2);

    println!("{}", result1);
    println!("{}", result2);
}

fn add_v1(first_number:i32, second_number:i32) -> i32 {
    first_number + second_number
}

fn add_v2(first_number:i32, second_number:i32) -> i32 {
    return first_number + second_number
}

fn hello_world_v2() {
    println!("Hello, Students!");
}
```

Functions

```
Hello, Students!
128
128
```

```rust
fn main() {
    // function call the doesn't return anything
    hello_world_v2();

    let n1 = 120;
    let n2 = 8;

    // two different ways of implementing the same function
    let result1 = add_v1(n1, n2);
    let result2 = add_v2(n1, n2);

    println!("{}", result1);
    println!("{}", result2);
}

fn add_v1(first_number:i32, second_number:i32) -> i32 {
    first_number + second_number
}

fn add_v2(first_number:i32, second_number:i32) -> i32 {
    return first_number + second_number
}

fn hello_world_v2() {
    println!("Hello, Students!");
}
```

Comments

Comments

```
// function call the doesn't return anything
hello_world_v2();
```

- Single-line comments can be added by doing two consecutive slashes
- In Rust, it's also customary for multi-line comments to all begin with consecutive slashes

Comments

```
// function call the doesn't return anything
// however, it does print “Hello students”
hello_world_v2();
```

- Single-line comments can be added by doing two consecutive slashes
- In Rust, it’s also customary for multi-line comments to all begin with consecutive slashes

Sidebar: Self-documenting code

- Self-documenting code is code that is inherently readable
- This doesn't mean you shouldn't write comments
- You should strive for self-documenting code, but…
    - If someone can't understand it by looking at the function, write documentation

Compound Types

Tuples in Rust

- **Tuples** are a compound type which can hold several values in a single variable
- We can create a tuple using a comma separated list in parentheses
- Tuples have **fixed length**

```
let my_tuple = ('a', 2, 42.35);
```

Tuples in Rust

- **Tuples** can also be **destructured**, turning their values into variables

```rust
let my_tuple = ('a', 2, 42.35);
let (char_var, int_var, float_var) = my_tuple;
```

Tuples in Rust

```rust
fn main() {
    // this is how we declare a tuple
    let tup_1 = ('a', 2, 42.35);

    // this is known as destructuring – we put the values
    // in the tuple in three different variables
    let (char_var, int_var, float_var) = tup_1;

    // print all the tuple values
    println!("The character (first value) in the tuple is: {}”, char_var);
    println!("The integer (second value) in the tuple is: {}", int_var);
    println!("The float (third value) in the tuple is: {}", float_var);
}
```

Standard Output

```
The character (first value) in the tuple is: a
The integer (second value) in the tuple is: 2
The float (third value) in the tuple is: 42.35
```

Arrays in Rust

- Arrays are another compound data type in Rust
- Arrays are:
    - **Fixed length** (different from other programming languages)
    - **Can only hold values with the same data type**
- We can create an array using a comma separated list in square brackets
- We will cover **vectors** in later lectures which can grow/shrink in size

Arrays in Rust

```rust
fn main() {
    // we can declare an array like so:
    let array_1 = ['a', 'b', 'c', 'd', 'e'];

    // we can also specify a type and size beforehand
    let array_2: [i32; 5] = [1, 2, 3, 4, 5];

    // if you want to initialize an array with the
    // same value for every index you do this:
    // the first value is the value you want
    // and the second is the array size
    let array_3 = ['🤠'; 5];

    // print arrays
    println!("array_1: {:?}", array_1);
    println!("array_2: {:?}", array_2);
    println!("array_3: {:?}", array_3);
}
```

Arrays in Rust

```rust
fn main() {
    // we can declare an array like so:
    let array_1 = ['a', 'b', 'c', 'd', 'e'];

    // we can also specify a type and size beforehand
    let array_2: [i32; 5] = [1, 2, 3, 4, 5];

    // if you want to initialize an array with the
      // same value for every index you do this:
    // the first value is the value you want
      // and the second is the array size
    let array_3 = ['🤠'; 5];

    // print arrays
    println!("array_1: {:?}", array_1);
    println!("array_2: {:?}", array_2);
    println!("array_3: {:?}", array_3);
}
```

```
array_1: ['a', 'b', 'c', 'd', ‘e']
array_2: [1, 2, 3, 4, 5]
array_3: ['🤠', '🤠', '🤠', '🤠', '🤠']
```

Indexing Arrays in Rust

```
fn main() {
    // we can declare an array like so:
    let array_1 = ['a', 'b', 'c', 'd', 'e'];

    // print arrays
    println!("array_1: {}", array_1[2]);
}
```

```
Output:
   c
```

Indexing Arrays in Rust

```
fn main() {
    // we can declare an array like so:
    let array_1 = ['a', 'b', 'c', 'd', 'e'];

    // print arrays
    println!("array_1: {}", array_1[8]);
}
```

```
Compiles, but crashes
```

Fancy for loops

```rust
fn main() {
    let a = [10, 20, 30, 40, 50];

    for element in a.iter() {
        println!("the value is: {}", element);
    }
}
```

```
the value is: 10
the value is: 20
the value is: 30
the value is: 40
the value is: 50
```

Summary

- Functions
- Comments
  - Self-Documenting Code
- Compound Types
  - Tuples
  - Arrays

Rust Basics

CS128 Honors

Slides by Matt Geimer (FA21)
Presented 9/13/2021