From C++ till Rust

Lodz, 16 March 2022

From C++ till Rust

In C++ we can

copy data,

share by passing by reference (or pointer)

or even move

From C++ till Rust

C++ is designed around being able to copy

From C++ till Rust

Copying is pushed to such an extreme level

that passing by pointer is in fact copying pointer

From C++ till Rust

References and pointers are designed to allow sharing data

From C++ till Rust

However, there are no safety mechanisms whatsoever built in

From C++ till Rust

```
{
    DayOfWeek wednesday(“Wednesday”);
    make_it_friday(wednesday);
    std::cout << wednesday→day << std::endl;
    DayOfWeek friday = create_friday(wednesday);
    std::cout << wednesday→day << std::endl;
    std::cout << friday→day << std::endl;
}
```

```
class DayOfWeek {
    std::string day
};
```

From C++ till Rust

```
{
    DayOfWeek wednesday(“Wednesday”);
    make_it_friday(wednesday);
    std::cout << wednesday→day << std::endl;
    DayOfWeek friday = create_friday(wednesday);
    std::cout << wednesday→day << std::endl;
    std::cout << friday→day << std::endl;
}
```

```
class DayOfWeek {
    std::string day
};
```

From C++ till Rust

```
make_it_friday(const DayOfWeek &dow)

DayOfWeek create_friday(DayOfWeek dow)
```

From C++ till Rust

Regardless, there was no good way to express uniqueness

From C++ till Rust

Move is like passing by reference but.. different

From C++ till Rust

```
DayOfWeek tomorrow = std::move(today)
```

or

```
create_friday(std::move(wednesday))
```

From C++ till Rust

C++

- Copy Constructor
- Assignment Operator
- Move Constructor // new s**t
- Move Operator // new s**t

```
class A {

    A(const A &rhs)

    A& operator=(const A &rhs)

    A(A &&rhs)

    A& operator=(const A &&rhs)

}
```

From C++ till Rust

A quick demo

From C++ till Rust

In Rust things are slightly different: we can

borrow data,

or move data

From C++ till Rust

Although there is still a way to copy/clone

From C++ till Rust

C++	Rust
Move	Move
Pass by reference / pointer	Borrow
Copy	Copy / Clone

From C++ till Rust

It looks familiar and thus simple but actually it is not

From C++ till Rust

All of this is backed up by the language itself,

checked (mostly) at the compile time

and there are rules

From C++ till Rust

Each value (thing on the right side)

is owned

by exactly one variable (thing on the left side) at a time

From C++ till Rust

Each value (thing on the right side)

is owned

by exactly one variable (thing on the left side) at a time

```
{

    let x = vec![1, 2, 3];

    // do things with x

    let y = x;

    // from now on x is invalid

}
```

From C++ till Rust

```
error[E0382]: borrow of moved value: `x`
 --> src/main.rs:5:25
  |
2 |     let x = vec![1, 2, 3];
  |         - move occurs because `x` has type `Vec<i32>`, which does not implement the `Copy` trait
3 |     let y = x;
  |             - value moved here
4 |     print!("Value: {}", y[0]);
5 |     print!("Value: {}", x[0]);
  |                         ^ value borrowed here after move

For more information about this error, try `rustc --explain E0382`.
```

From C++ till Rust

Borrowing is passing by reference, but…

From C++ till Rust

Variables can be mutable or not

References can be mutable or not

From C++ till Rust

Variables can be mutable or not

References can be mutable or not

```rust
fn main() {

    let mut x = vec![1, 2, 3];

    let y = &mut x;

    // use x and y

}
```

From C++ till Rust

```rust
fn this_is_moving(v: Vec<i32>)

fn this_is_borrowing(v: &Vec<i32>)
```

From C++ till Rust

I’ve said earlier that you can also Copy / Clone

From C++ till Rust

```
error[E0382]: borrow of moved value: `x`
 --> src/main.rs:5:25
  |
2 |     let x = vec![1, 2, 3];
  |         - move occurs because `x` has type `Vec<i32>`, which does not implement the `Copy` trait
3 |     let y = x;
  |             - value moved here
4 |     print!("Value: {}", y[0]);
5 |     print!("Value: {}", x[0]);
  |                         ^ value borrowed here after move

For more information about this error, try `rustc --explain E0382`.
```

From C++ till Rust

The Copy trait makes a type copyable

It comes together with Clone trait

From C++ till Rust

Another demo

From C++ till Rust

A Trait is like a well.. a trait