1 `main` — MIR Walkthrough

Purpose: TODO: Describe why this walkthrough exists

1.1 Source Context

```rust
fn main () {

    let mut a = MyStruct { a_value: 32, another: false, a_third: 32};

    // mutate field through ref and double-ref
    let r1 = &mut (a.a_value);
    *r1 = 42;
    assert!(a.a_value == 42);
    let mut r1 = &mut (a.a_value);
    let r2 = &mut r1;
    **r2 = 43;
    assert!(a.a_value == 43);

    // create reference-field chain
    let mut e = Enclosing{inner: &mut a};
    let ee = &mut e;

    // read and  write values through chain of ref/field projections
    let vv = (*(*ee).inner).a_value;
    assert!(vv == 43);

    (*(*ee).inner).another = true;

    let r3 = &mut (*ee).inner.a_third;
    *r3 = (*(*ee).inner).a_value as usize;

    assert!(a.another);
    assert!(a.a_third == 43);

}
```

1.2 Function Overview

- **Function:** `main`
- **Basic blocks:** 11
- **Return type:** `() (0 bytes, align 1)`
- **Notable properties:**
  - Contains panic path
  - Introduces borrows
  - Has conditional branches

1.3 Locals

Local	Type	Notes
0	`() (0 bytes, align 1)`	Return place
1	`MyStruct (16 bytes, align 8)`	
2	`&mut i8 (8 bytes, align 8)`	
3	`Int(I8)`	
4	`()`	
5	`&mut i8 (8 bytes, align 8)`	
6	`&mut &mut i8 (8 bytes, align 8)`	
7	`Int(I8)`	

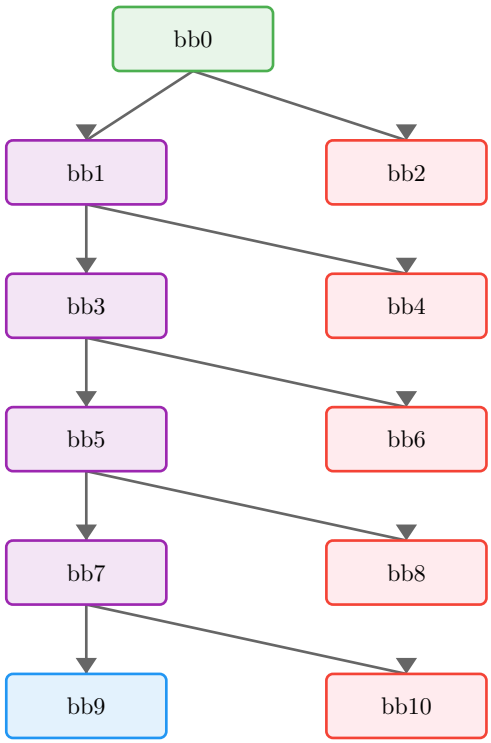
8	()	
9	Enclosing<'_> (8 bytes, align 8)	
10	&mut MyStruct (8 bytes, align 8)	
11	&mut Enclosing<'_> (8 bytes, align 8)	
12	Int(I8)	
13	()	
14	&mut usize (8 bytes, align 8)	
15	Int(I8)	
16	Bool	
17	()	
18	Uint(Usize)	
19	()	
20	&mut i8 (8 bytes, align 8)	
21	&mut MyStruct (8 bytes, align 8)	
22	&mut MyStruct (8 bytes, align 8)	
23	&mut MyStruct (8 bytes, align 8)	
24	&mut MyStruct (8 bytes, align 8)	

1.4 Borrows

#	Borrow	Kind	Created At	Borrowed
0	_6	&mut	bb1[1]	_5
1	_10	&mut	bb3[0]	_1
2	_11	&mut	bb3[2]	_9

Borrows are tracked conservatively: active from creation until reassignment or scope end.

1.5 Control-Flow Overview



1.6 Basic Blocks

1.6.1 bb0 — entry

Entry point of the function.

MIR	Annotation
`_1 = MyStruct(32, 0, 32)`	Construct aggregate
`_2 = &mut _1.0`	Mutable borrow
`(*_2) = 42`	Load constant
`_3 = _1.0`	Copy value
`→ switch(move _3) [42→bb1; else→bb2]`	Branch on move _3

1.6.2 bb1 — branch point

MIR	Annotation
`_5 = &mut _1.0`	Mutable borrow
`_6 = &mut _5`	Mutable borrow
`_20 = copy_deref((*_6))`	
`(*_20) = 43`	Load constant
`_7 = _1.0`	Copy value
`→ switch(move _7) [43→bb3; else→bb4]`	Branch on move _7

1.6.3 bb2 — panic path

Panic/diverging path.

MIR	Annotation
`→ _4 = panic([16 bytes])`	Call panic

1.6.4 bb3 — branch point

MIR	Annotation
`_10 = &mut _1`	Mutable borrow
`_9 = Enclosing(_10)`	Construct aggregate
`_11 = &mut _9`	Mutable borrow
`_21 = copy_deref((*_11).0)`	
`_12 = (*_21).0`	Copy value
`→ switch(_12) [43→bb5; else→bb6]`	Branch on _12

1.6.5 bb4 — panic path

Panic/diverging path.

MIR	Annotation
`→ _8 = panic([16 bytes])`	Call panic

1.6.6 bb5 — branch point

MIR	Annotation
`_22 = copy_deref((*_11).0)`	
`(*_22).1 = 1`	Load constant
`_23 = copy_deref((*_11).0)`	
`_14 = &mut (*_23).2`	Mutable borrow

`_24 = copy_deref((*_11).0)`	
`_15 = (*_24).0`	Copy value
`(*_14) = move _15 as RigidTy(Uint(Usize))`	Integer conversion
`_16 = _1.1`	Copy value
`→ switch(move _16) [0→bb8; else→bb7]`	Branch on move _16

1.6.7 bb6 — panic path

Panic/diverging path.

MIR	Annotation
`→ _13 = panic([16 bytes])`	Call panic

1.6.8 bb7 — branch point

MIR	Annotation
`_18 = _1.2`	Copy value
`→ switch(move _18) [43→bb9; else→bb10]`	Branch on move _18

1.6.9 bb8 — panic path

Panic/diverging path.

MIR	Annotation
`→ _17 = panic([16 bytes])`	Call panic

1.6.10 bb9 — return / success

Normal return path.

MIR	Annotation
`→ return`	Return from function

1.6.11 bb10 — panic path

Panic/diverging path.

MIR	Annotation
`→ _19 = panic([16 bytes])`	Call panic

1.7 Key Observations

TODO: Add bullet points summarizing what this MIR teaches

-
-

1.8 Takeaways

TODO: One or two sentences to generalize this example