

# Tipe Data Rust

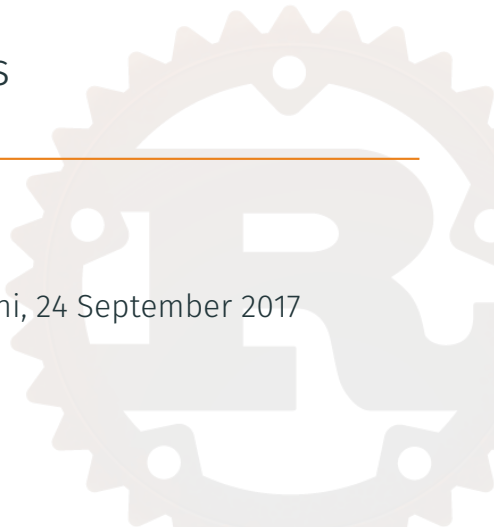
## Primitive data types

---

Muhamad Aldo Ridhoni

Padepokan ASA Wedomartani, 24 September 2017

RUST ID Meetup #2



# Membahas

- Tipe Data Dasar



# Membahas

- Tipe Data Dasar
- Penggunaan



# Membahas

- Tipe Data Dasar
- Penggunaan
- Konversi



# Tipe Data Dasar

Scalar *single value* :

- Boolean
- Integer dan floating-point
- Character



# Tipe Data Dasar

## Scalar *single value* :

- Boolean
- Integer dan floating-point
- Character

## Compound *multiple value* :

- Tuples
- Arrays
- Slices
- String



# Boolean



Tipe boolean dengan nilai **true** atau **false**.

# Boolean

```
let benar: bool = true;  
let salah: bool = false;
```

```
if benar {  
    println!("Berhasil")  
}
```

```
assert_eq!(benar as i8, 1);  
assert_eq!(salah as i8, 0);
```



# Integer

Integer adalah tipe data yang merepresentasikan bilangan bulat seperti dalam matematika.

**Signed** Cakupan integer bertanda dari  $-(2^{n-1})$  hingga  $2^{n-1} - 1$ .

**Unsigned** Integer tidak bertanda dari 0 hingga  $2^n - 1$ .

# Integer

## Cakupan

Panjang	<i>Signed</i>	<i>Unsigned</i>
8-bit	i8	u8
16-bit	i16	u16
32-bit	i32	u32
64-bit	i64	i64
arch	isize	usize

- **i8** = -128 .. 127  
**u8** = 0 .. 255
- **i16** = -32.768 .. 32.767  
**u16** = 0 .. 65.535
- **i32** = -2.147.483.648 .. 2.147.483.647  
**u32** = 0 .. 4.294.967.295
- **i64** = -9.223.372.036.854.775.808 ..  
9.223.372.036.854.775.807  
**u64** = 0 .. 18.446.744.073.709.551.615

# Integer Literals

Jenis	Contoh
Desimal	45_678
Hexadesimal	0xbb
Oktal	0o77
Binary	0b1100_1010
Byte (u8)	b'Z'

# Floating-Point


Floating-point adalah tipe numerik dengan titik desimal.

**f32** 32bit single precision float.  
32bit dengan presisi 24bit.

**f64** 64bit double precision float.  
64bit dengan presisi 53bit.

Tipe default untuk float adalah **f64**.

# Floating-Point



```
let f1: f32 = 1.1234567;  
let f2: f32 = 0.000000006;  
let f3: f32 = 1.1234568;  
  
assert_eq!(f1 + f2, f3); // true
```

# Character

Char adalah tipe data yang memuat nilai **Unicode Scalar**. Satu karakter dimuat dalam petik `' '`.

# Character

```
let c = 'z';  
let z = 'Z';  
let black_chess_knight = '♞';
```

```
// hanya u8
```

```
let y: char = char::from(0x79);  
let x: char = char::from(b'x');
```

# Tuple

Tuple (**T**, **U**, ...) adalah rangkaian yang dapat terdiri dari elemen dengan tipe berbeda dan dimuat dalam tanda kurung **()**.

Elemen tuple dapat diakses dengan indeks *'tuple indexing'*.



# Tuple

```
type Pair<T1, T2> = (T1, T2);  
let p: Pair<i8, f32> = (10, 3.14);  
let q: Pair<&str, i8> = ("ID", 9);  
let (a, b) = p;
```

```
assert_eq!(a, 10);  
assert_eq!(b, 3.14);  
assert_eq!(p.0, 10);  
assert_eq!(p.1, 3.14);
```

# Array

Array (**[T; N]**) adalah koleksi padat elemen-elemen dengan tipe data yang sama dan memiliki panjang yang tidak bisa berubah setelah dideklarasikan.

Sintaks array baru dengan membuat daftar nilai yang terpisah oleh koma didalam kurung siku **[ ]**.

# Array

*// dengan tipe annotation*

```
let array: [i32; 3] = [1, 2, 3];
```

*// repeat expression*

```
let mut array: [i32; 3] = [0; 3];
```

```
let bulan = ["Januari", "Pebruari",  
    ↪ "Maret", "April", "Mei", "Juni",  
    ↪ "Juli", "Agustus", "September",  
    ↪ "Oktober", "Nopember", "Desember"];
```

```
let jan = bulan[0];
```

```
let dec = bulan[11];
```

# Slice

Slice (**&[T]**) adalah 'view' untuk koleksi seperti array dengan ukuran yang *dinamis*. Dinamis dalam arti ukuran tidak diketahui saat kompilasi.

Slice menampilkan blok memori dengan representasi sebagai sebuah pointer serta ukuran panjangnya.

# Slice

```
let str_slice: &[&str] = &["one", "two"];

let arr = [1, 2, 3, 4, 5, 6];
let arr_slice: &[i32] = &arr[1..4];

println!("{:?}", arr_slice); // [2, 3, 4]

assert_eq!(Some(&3), arr_slice.get(1));
// atau
assert_eq!(3, arr_slice[1]);
```

# String slices



Tipe dasar **str** untuk menyimpan kalimat *string*.

Nilai dari strings slices selalu valid UTF-8.

# String slices

```
let hello = "Hello, world!";
```

```
// dengan tipe annotation
```

```
let hello: &'static str = "Hello, world!";
```

# Tipe-tipe Lain

Rust Standard Library menyediakan banyak tipe antara lain:

**Vector** `Vec<T>`, Tipe Array yang dapat bertambah dan dialokasikan di *heap*.

**Box** `Box<T>`, Tipe pointer untuk alokasi data di *heap*.

**String** Tipe kalimat *string* yang dapat bertambah dengan encoding UTF-8



# Konversi Antar Tipe (Casting)

to \ from	i32	u32	f64	String
i32	n/a	x as <b>u32</b>	x as <b>f64</b>	x.to_string()
u32	x as <b>i32</b>	n/a	x as <b>f64</b>	x.to_string()
f64	x as <b>i32</b>	x as <b>u32</b>	n/a	x.to_string()

String ke tipe numerik :

`x.parse()`

Secara explicit dengan type annotation :

`x.parse::i32()`

# Konversi Antar Tipe String

to from	String	&str
String	n/a	&*x
&str	x.to_string()	n/a

# Kesimpulan

- Pilih tipe data yang tepat.
- Cobalah kasus dengan *test*.
- Tangani *error*.



Sekian & Terima Kasih



# Referensi

## Pustaka

▶ <https://doc.rust-lang.org/std/index.html#primitives>

▶ <https://doc.rust-lang.org/reference/types.html>

▶ <https://doc.rust-lang.org/book/second-edition/ch03-02-data-types.html>

▶ <http://carols10cents.github.io/rust-conversion-reference>

▶ <https://www.manning.com/books/rust-in-action>

▶ <https://www.doc.ic.ac.uk/~eedwards/compsys/float/>



# Referensi

## Kode

▶ <https://github.com/rust-lang/rust/blob/master/src/libcore/num>

▶ <https://github.com/rust-lang/rust/blob/master/src/test/parse-fail/lex-bad-numeric-literals.rs>

## Lain-lain

▶ [https://en.wikipedia.org/wiki/Integer\\_\(computer\\_science\)](https://en.wikipedia.org/wiki/Integer_(computer_science))

▶ <https://unicode-table.com/en/#control-character>

▶ <https://play.rust-lang.org/>

▶ <https://rust.godbolt.org/>