

# Системне Програмування

З використанням мови програмування **Rust.**  
**Fundamentals. Primitive Types**

# Базові типи даних

- цілі числа
  - u8 / i8
  - u16 / i16
  - u32 / i32
  - u64 / i64
  - u128 / i128
  - usize / isize
- дрібні
  - f32
  - f64
- bool
- char UTF8
- unit
- never

Назва	Розмір, байтів
i8 / u8	1
i16 / u16	2
i32 / u32	4
i64 / u64	8
i128 / u128	16
isize / usize	2...16
f32	4
f64	8
bool	1
char	1...4
unit	0
never	0



# Діапазони

MIN			MAX
-----			
i8:	-128		127
i16:	-32768		32767
i32:	-2147483648		2147483647
i64:	-9223372036854775808		9223372036854775807
i128:	-170141183460469231731687303715884105728		170141183460469231731687303715884105727
isize:	-9223372036854775808		9223372036854775807
-----			
u8:	0		255
u16:	0		65535
u32:	0		4294967295
u64:	0		18446744073709551615
u128:	0		340282366920938463463374607431768211455
usize:	0		18446744073709551615
-----			
f32:	-3.4028235e38		3.4028235e38
f64:	-1.7976931348623157e308		1.7976931348623157e308



# Бінарна система обчислення

0	00000000
1	00000001
2	00000010
3	00000011
4	00000100
.....	
124	01111100
125	01111101
126	01111110
127	01111111

42							
0	0	1	0	1	0	1	0
$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
128	64	32	16	8	4	2	1

Використовуються  
тільки цифри **0** та **1**

5	00000101
7	00000111
12	00001100



# Представлення від'ємних чисел

```
-3 11111101
-2 11111110
-1 11111111
 0 00000000
 1 00000001
 2 00000010
 3 00000011
 4 00000100
.....
124 01111100
125 01111101
126 01111110
127 01111111
-128 10000000
-127 10000001
-126 10000010
-125 10000011
```

```
-5 11111011
 7 00000111
12 00001100
```

[https://en.wikipedia.org/wiki/Two's\\_complement](https://en.wikipedia.org/wiki/Two's_complement)

# Переповнення

```
let a: u8 = 252;  
let b: u8 = 5;  
let c = a + b; // panic here
```

```
let a: u8 = 1;  
let c = a - 2; // panic here
```



# Перевірка переповнення

```
let x = u8::checked_add(251_u8, 10);  
// what type of is here ???
```

# Конвертація типів

```
let x: u8 = 5;  
let y: i16 = 10;  
let z = x + y; // will not compile
```

```
let z = x as i16 + y;  
let z = x + y as u8;
```

# Але...

```
let x = 1234_i32;  
let y = 10_u8;  
let z = x as u8 + y;
```

```
let x: i32 = -5;  
let y = x as u32;
```

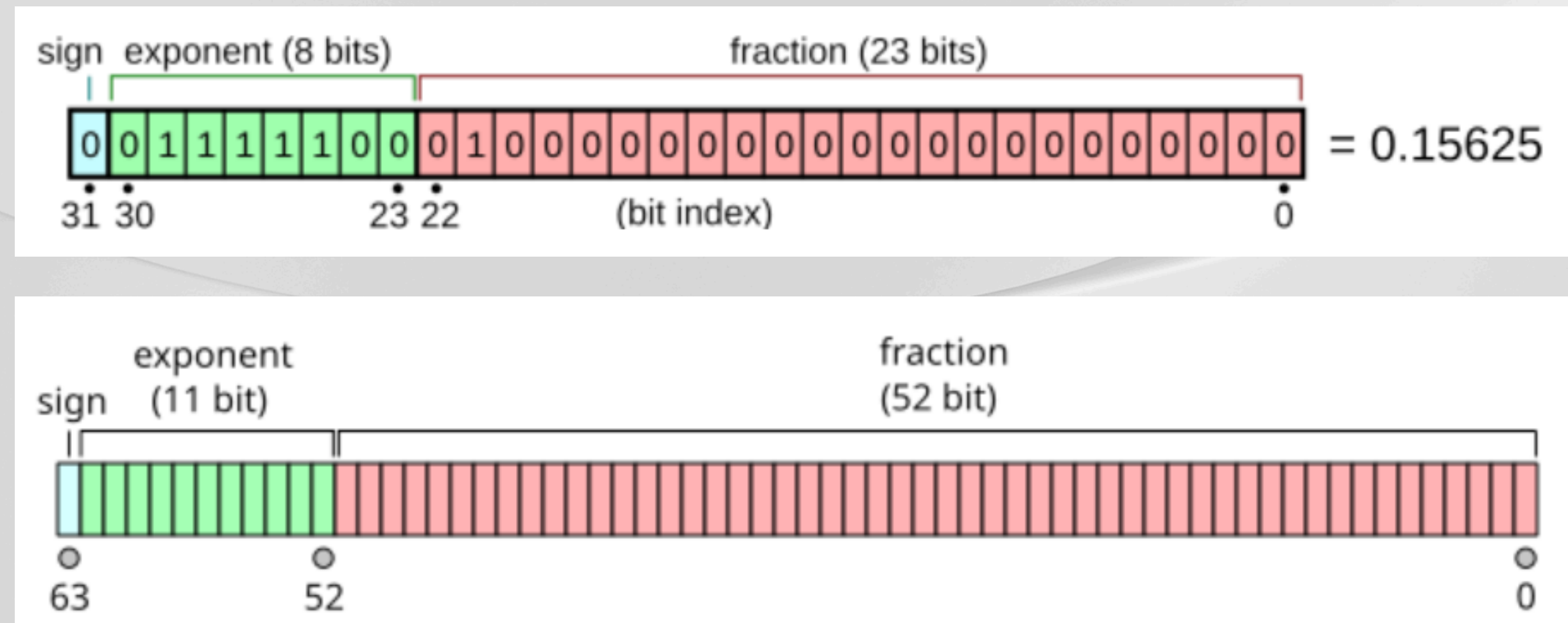
**Безпечно:** u8 -> u16 -> u32 -> u64 -> u128  
i8 -> i16 -> i32 -> i64 -> i128



# Тип float. Репрезентація

$$(-1)^S * 1.M * 2^{(E-127)}$$

# Тип float. Репрезентація



[https://en.wikipedia.org/wiki/Double-precision\\_floating-point\\_format](https://en.wikipedia.org/wiki/Double-precision_floating-point_format)



# Тип float. Репрезентація

[illegible]



# Тип float. Точність

decimal		float		float binary
16777216		16777216		01001011100000000000000000000000
16777217		16777216		01001011100000000000000000000000
16777218		16777218		01001011100000000000000000000001



# Тип float. Точність 2

```
let x = 0.2 + 0.1;  
println!("{}", x);
```

```
// 0.30000000000000004
```

0.1		<del>0</del> 0111101110011001100110011001101
0.2		<del>0</del> 0111110010011001100110011001101

# Тип float. Чи існують числа які можуть бути точно представлені?



# Тип float. Чи існують числа які можуть бути точно представлені?

1. Цілі числа  $-16777216 \dots 16777216$
2. Дрібні ступені двійки  
 $1/2, 1/4, 1/8 \dots$  та їх довільні комбінації





# Різні системи обчислення

Decimal: 0,1,2,3,4,5,6,7,8,9 (default)

let x = 5;

Binary: 0, 1

let x = 0b111;

Octal: 0, 1, 2, 3, 4, 5, 6, 7

let x = 0o71;

Hexadecimal: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f

let x = 0xF0AC;

# Можливість “виводити тип” та задавати

```
let x = 5; // i32
```

```
let x = 5u8;
```

```
let x = -5u8; // ???
```

```
let x = 5u16;
```

```
let x: u16 = 5u16;
```



# Можливість зручно задавати літерали

```
let x = 4441211541383370;
```

```
let x = 4441_2115_4138_3370;
```

# Типи даних Unit, never

```
let x = println!("hello");
```

```
let x = loop {  
    print!(".")  
};
```



**Приклади коду з лекції  
знаходяться на GitHub:**

<https://github.com/djnzx/rust-course>