



Rust 编程语言入门



Microsoft®
Most Valuable
Professional

杨旭，微软MVP
Rust、Go、C#开发者

3.2 数据类型

- 标量和复合类型
- Rust 是静态编译语言，在编译时必须知道所有变量的类型
 - 基于使用的值，编译器通常能够推断出它的具体类型
 - 但如果可能的类型比较多（例如把 String 转为整数的 parse 方法），就必须添加类型的标注，否则编译会报错（例子）

标量类型

- 一个标量类型代表一个单个的值
- Rust 有四个主要的标量类型：
 - 整数类型
 - 浮点类型
 - 布尔类型
 - 字符类型

整数类型

- 整数类型没有小数部分
- 例如 u32 就是一个无符号的整数类型，占据 32 位的空间
- 无符号整数类型以 u 开头
- 有符号整数类型以 i 开头
- Rust 的整数类型列表如图：
 - 每种都分 i 和 u，以及固定的位数
 - 有符号范围：
 - $-(2^n - 1)$ 到 $2^{n-1} - 1$
 - 无符号范围：
 - 0 到 $2^n - 1$

Length	Signed	Unsigned
8-bit	i8	u8
16-bit	i16	u16
32-bit	i32	u32
64-bit	i64	u64
128-bit	i128	u128
arch	isize	usize

isize 和 usize 类型

- isize 和 usize 类型的位数由程序运行的计算机的架构所决定：
 - 如果是 64 位计算机，那就是 64 位的
 - ...
- 使用 isize 或 usize 的主要场景是对某种集合进行索引操作。

整数字面值

- 除了 `byte` 类型外，所有的数值字面值都允许使用类型后缀。
 - 例如 `57u8`
- 如果你不太清楚应该使用那种类型，可以使用 Rust 相应的默认类型：
- 整数的默认类型就是 `i32`：
 - 总体上来说速度很快，即使在 64 位系统中

Number literals	Example
Decimal	<code>98_222</code>
Hex	<code>0xff</code>
Octal	<code>0o77</code>
Binary	<code>0b1111_0000</code>
Byte (u8 only)	<code>b'A'</code>

整数溢出

- 例如：u8 的范围是 0 – 255，如果你把一个 u8 变量的值设为 256，那么：
 - 调试模式下编译：Rust 会检查整数溢出，如果发生溢出，程序在运行时就会 panic
 - 发布模式下（--release）编译：Rust 不会检查可能导致 panic 的整数溢出
 - 如果溢出发生：Rust 会执行“环绕”操作：
 - 256 变成 0，257 变成 1...
 - 但程序不会 panic

浮点类型

- Rust 有两种基础的浮点类型，也就是含有小数部分的类型
 - f32, 32位, 单精度
 - f64, 64位, 双精度
- Rust 的浮点类型使用了 IEEE-754 标准来表述
- f64 是默认类型，因为在现代 CPU 上 f64 和 f32 的速度差不多，而且精度更高。
- （例子）

数值操作

- 加减乘除余等（例子）

布尔类型

- Rust 的布尔类型也有两个值：true 和 false
- 一个字节大小
- 符号是 bool
- （例子）

字符类型

- Rust 语言中 `char` 类型被用来描述语言中最基础的单个字符。
- 字符类型的字面值使用单引号
- 占用 4 字节大小
- 是 Unicode 标量值，可以表示比 ASCII 多得多的字符内容：拼音、中日韩文、零长度空白字符、emoji 表情等。
 - U+0000 到 U+D7FF
 - U+E000 到 U+10FFFF
- 但 Unicode 中并没有“字符”的概念，所以直觉上认为的字符也许与 Rust 中的概念并不相符
- （例子）

再见

