- 所谓类型，就是对表示信息的值 进行细粒度的区分。

- 不同的类型占用的内存不同。

- 不同类型如何计算？需要转化。这种类型间的转化，形成了类型系统。

- 类型行为：基于类型定义的一系列组合、运算和转换等方法。
- 类型行为决定了类型该如何计算，同时这也是一种约束。

类型系统通用概念

类型系统的 作用

- 排查错误：编译期间，排查类型不一致、未定义等情况，避免运行后崩溃。
- 抽象：抽象能力有助于强化编程规范，比如面向对象中的类型就可以作为一种类型。
- 文档：在阅读代码时，明确的类型声明，可以表明程序的行为。
- 优化效率：这一点是针对静态语言来说的，在编译器通过类型检查来优化一些操作，节省运行时的时间。
- 类型安全
  - 类型安全的语言可以避免 类型间的无效计算
  - 类型安全的语言可以避免 语义上的逻辑错误
  - 类型安全的语言可以保证 内存安全

类型系统的 分类

- 分类
  - 静态类型：在编译期 进行类型检查
  - 动态类型：在运行期 进行类型检查

类型系统与 多态性

多态类型系统：允许一段代码在不同的上下文中具有不同的类型。

- 参数化多态 范型：函数、数据类型都需要适用于多种类型。
- Ad-hoc 多态 trait：是指同一种行为定义，不同的上下文中，会响应不同的行为实现。
- 子类型 多态 面向对象中的继承：代表一种包含关系
  - rust中没有继承概念，所以也没有子类型多态

静多态：发生在编译期，有参数化多态，Ad-hoc多态，不灵活，性能高。
动多态：发生在运行期，有子类型多态，灵活，性能低。