复合类型

可以将**多个值放在一个类型里**
Rust 提供了两种基础的复合类型：**元组（Tuple）、数组**

创建 Tuple

- 在小括号里，将值用逗号分开
- Tuple 中的每个位置都对应一个类型，Tuple 中各元素的类型不必相同

Example：

```
fn main() {
    let tup: (i32, f64, u8) = (500, 6.4, 1);

    println!("{}, {}, {}, ", tup.0, tup.1, tup.2);
}

```

在 tuple 中分别定义了不同类型的变量
利用解构来获取 tuple 中的元素值

```
fn main() {
    let tup: (i32, f64, u8) = (500, 6.4, 1);

    let (x, y, z) = tup;

    println!("{}, {}, {}, ", x, y, z);
}

```

把变量分别赋值到 tup 中（也是之间想不明白为什么 c 语言不能这样用变量）

访问 tuple 的元素

在 tuple 变量使用电标记法，后接元素的索引号（和 c 语言数组访问非常相似）

```
fn main() {
    let tup: (i32, f64, u8) = (500, 6.4, 1);

    println!("{}, {}, {}, ", tup.0, tup.1, tup.2);
}

```

数组

- 数组也可以将多个值放在一个类型里
- 数组中每个元素的类型必须相同
- 数组的长度也是固定的

中括号中 各变量分开

数组的用处

- 如果想让你的数据存放在 stack（栈）上而不是 heap（堆）上，或者想保证有固定数量的元素，这时使用数组更有好处
- 数组没有 Vector 灵活（以后再讲）。
  - Vector 和数组类似，它由标准库提供
  - Vector 的长度可以改变
  - 如果你不确定应该用数组还是 Vector，那么估计你应该用 Vector。

数组的类型

- 数组的类型以这种形式表示：[类型；长度]
  - 例如： let a: [i32; 5] = [1, 2, 3, 4, 5];

另一种声明数组的方法

- 如果数组的每个元素值都相同，那么可以在：
  - 在中括号里指定初始值
  - 然后是一个“；”
  - 最后是数组的长度
- 例如： let a = [3; 5]; 它就相当于：let a = [3, 3, 3, 3, 3];

访问数组元素和 c 语言一样，直接用索引值访问就可以
如果访问索引超过数组范围 虽然编译会通过但是运行中报错 Rust **不会允许其继续访问相应地址的内存（安全性）**