

猜数游戏 --- 一次猜测

知识点: let、match 等方法

相关函数

外部的 crate 第三方包、库

猜数游戏 - 目标

- 生成一个 1 到 100 间的随机数
- 提示玩家输入一个猜测
- 猜完之后，程序会提示猜测是太小了还是太大了
- 如果猜测正确，那么打印出一个庆祝信息，程序退出

```
c > main.rs > ...
1 use std::io; // prelude
2
3 fn main() {
4     println!("猜数!");
5
6     println!("猜测一个数");
7
8     // let mut foo = 1;
9     // let bar = foo; // imm
10
11     // foo = 2;
12
13     let mut guess = String::new();
14 }
```

pub const fn new() → String
Creates a new empty `String`.
Given that the `String` is empty, this will not allocate any initial buffer. While that means this initial operation is very inexpensive, it may cause excessive allocation later when you add data. If you have an idea of how much data the `String` will hold, consider the `with_capacity` method to prevent excessive re-allocation.

Examples

Basic usage:

```
let foo = 1;
```

```
let bar = foo;
```

在 rust 中默认所有变量是不可变的 **immutable** 如果想要生成可变的变量 就在前边加 **mut**

```
let mut guess = String::new();
```

使用 **let** 声明了一个变量 **guess** 将其绑定到一个空的字符串上边 字符串 **String** **::** 表明 **new** 是 **String** 的一个关联函数，具体函数内容如上 **new** 函数相当于创建了一个空白的字符串 创建类型实例的惯用函数名称->**new**

```
c > main.rs > ...
1 use std::io; // prelude
2
3 fn main() {
4     println!("猜数!");
5
6     println!("猜测一个数");
7
8     // let mut foo = 1;
9     // let bar = foo; // imm
10
11     // foo = 2;
12
13     let mut guess = String::new();
14
15     io::stdin().read_line(&mut guess).expect("无法读取行");
16
17     println!("你猜测的数是: {}", guess);
18 }
```

pub fn stdin() → Stdin
<https://doc.rust-lang.org/nightly/std/io/stdio/stdin.v.html>
Constructs a new handle to the standard input of the current process.
Each handle returned is a reference to a shared global buffer whose access is synchronized via a mutex. If you need more explicit control over locking, see the `[Stdin::lock]` method.

Note: Windows Portability Consideration
When operating in a console, the Windows implementation of this stream does not support non-UTF-8 byte sequences. Attempting to read bytes that are not valid UTF-8 will return an error.

io 库有一个函数叫 `stdin` 返回一个 `S` 的实例 用作句柄处理终端中的函数

通过 `read_line` 的方法 我们可以获得用户的输入 `read_line` 的方法就是把输入放进一个字符串中,所以需要 一个字符串类型作为 `read_line` 的参数 而 `guess` 是可变的 因为要输入

取地址符号表示一个引用 表明这个方法是按照引用传递的,引用在 rust 中是一个很复杂的特性,rust 的核心竞争力之一就是安全的使用引用功能

`result` 是一个枚举类型 有几个固定的值 `io::result` 就有两个值

`io::result Ok,Err` 两个变体 `Ok` 表示操作成功 返回值; `Err` 代表失败

`expect` 是定义的方法之一 对潜在的错误进行处理 会出波浪线

```
use std::io; // prelude

fn main() {
    println!("猜数!");
    println!("猜测一个数");

    // let mut foo = 1;
    // let bar = foo; // immutable

    // foo = 2;

    let mut guess = String::new();
    match io::stdin().read_line(&mut guess) {
        Ok(_) => {}
        Err(_) => {}
    }
    io::stdin().read_line(&mut guess).expect("无法读取行");
}
```

Examples

`pub fn read_line(&self, buf: &mut String) -> io::Result<usize>`
Locks this handle and reads a line of input, appending it to the specified buffer.
For detailed semantics of this method, see the documentation on [BufRead::read_line].

`use std::io;`
`let mut input = String::new();`
`match io::stdin().read_line(&mut input) {`
`io::stdin().read_line(&mut guess).expect("无法读取行");`

```
fn main() {
    println!("猜数!");
    println!("猜测一个数");

    // let mut foo = 1;
    // let bar = foo; // immutable

    // foo = 2;

    let mut guess = String::new();
    match io::stdin().read_line(&mut guess) {
        Ok(_) => {}
        Err(_) => {}
    }
    io::stdin().read_line(&mut guess).expect("无法读取行");

    println!("你猜测的数是: {}", guess);
}
```

Examples

`pub fn read_line(&self, buf: &mut String) -> io::Result<usize>`
Locks this handle and reads a line of input, appending it to the specified buffer.
For detailed semantics of this method, see the documentation on [BufRead::read_line].

`use std::io;`
`let mut input = String::new();`
`match io::stdin().read_line(&mut input) {`
`io::stdin().read_line(&mut guess).expect("无法读取行");`

```
c> main.rs > ...
1 use std::io; // prelude
2
3 fn main() {
4     println!("猜数!");
5     println!("猜测一个数");
6
7     // let mut foo = 1;
8     // let bar = foo; // immutable
9
10    // foo = 2;
11
12    let mut guess = String::new();
13    match io::stdin().read_line(&mut guess) {
14        Ok(_) => {}
15        Err(_) => {}
16    }
17    io::stdin().read_line(&mut guess).expect("无法读取行");
18 }
```

`pub fn expect(self, msg: &str) -> T`
Returns the contained [Ok] value, consuming the self value.

Panics
Panics if the value is an [Err], with a panic message including the passed message content of the [Err].

Examples
Basic usage:
`io::stdin().read_line(&mut guess).expect("无法读取行");`

总体代码预览

```
main.rs > ...  
// use std::io; // prelude  
  
fn main() {  
    println!("猜数! ");  
  
    println!("猜测一个数");  
  
    // let mut foo = 1;  
    // let bar = foo; // immutable  
  
    // foo = 2;  
  
    let mut guess = String::new();  
  
    std::io::stdin().read_line(&mut guess).expect("无法读取行");  
  
    println!("你猜测的数是: {}", guess);  
}
```

在电脑运行的结果

```
Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.60  
s  
Running `target\debug\guessing_game.exe`  
猜数!  
猜测一个数  
2  
你猜测的数是:2
```