Entrada, saída, Strings e números inteiros

Jogo da adivinhação

O Jogo: entrada e saída básicas

Nosso primeiro grande projeto será a criação de um jogo que escolhe um número aleatório, e nos desafia a adivinhá-lo. O jogo permite a escolha do nível de dificuldade, nos dá feedback constante sobre nossos erros e acertos.

Portanto nosso primeiro programa nos diz se o número é maior ou menor do que o escolhido pelo computador. Nosso programa é um arquivo escrito na linguagem Ruby, que nome escolher para ele? Como o jogo brinca com maior ou menor, vamos chamar de `maior_ou_menor`, mas qual a extensão para ele? O padrão do ruby é utilizar a extensão `rb`, portanto criamos um arquivo chamado `maior_ou_menor.rb`. O conteúdo dele? Vazio.

Dado esse programa, vazio, queremos mandar o computador executá-lo. Para isso dizemos pro `ruby` rodar nosso arquivo:

```
ruby maior_ou_menor.rb
```

E nada acontece. Claro, não havia nada para ser feito.

Um programa é uma lista de comandos que o computador obedece. Somos os donos do computador não só fisicamente, mas somos o dono de suas "mentes". Somos capazes de dizer comandos que o computador deve obedecer. Esses comandos serão escritos aqui na linguagem ruby, e um tradutor, um intérprete, será capaz de traduzir esses comandos para um código "maluco" que nossa máquina entende. Afinal, quem hoje em dia quer aprender a falar a lingua das máquinas? Muito complicado, principalmente para começar a programar. Então aprendemos uma linguagem mais próximo de nós do que dos computadores (uma *linguagem de alto nível*). Qual o primeiro comando que quero dar para meu computador? Por favor, imprima uma mensagem de boas vindas.

Para isso dizemos para o computador colocar uma mensagem na saída, coloque, `puts`, a mensagem `"Bem vindo ao jogo da adivinhação"`:

```
puts "Bem vindo ao jogo da adivinhação"
```

E a saída ao rodar novamente `ruby maior_ou_menor.rb` é agora o que esperamos:

```
Bem vindo ao jogo da adivinhação
```

Já somos os donos. Ele nos obedece, e a mágica agora está em aprender mais comandos e formas de juntar comandos para criar programas complexos que fazem as mais diversas tarefas, desde um jogo até mesmo um piloto automático de avião. Todos eles são códigos escritos por seres humanos, e o computador obedece.

Lendo o nome de um jogador

No nosso caso, queremos perguntar ao usuário qual é o nome dele, para personalizarmos sua experência. Não é a toa que os jogos de hoje em dia perguntam o nome, no fim é possível lembrar quem foi o melhor jogador, criar um rank etc. No nosso caso começaremos pedindo o nome dele, usando novamente o `puts` para colocar uma mensagem na saída do computador:

```
puts "Bem vindo ao jogo da adivinhação"
puts "Qual é o seu nome?"
```

E o inverso? Queremos que nosso programa leia um dado, pegue (`gets`) uma informação do usuário, a entrada de dados mais simples é feita com a função `gets` , que devolve um texto, um valor que o usuário digitou, junto com o Enter (o *Return*):

```
nome = gets
```

Imprimimos então o nome da mesma maneira que imprimimos outras mensagem:

```
puts "Bem vindo ao jogo da adivinhação"
puts "Qual é o seu nome?"

nome = gets
puts "Começaremos o jogo para você, "
puts nome
```

E após executarmos com `ruby maior_ou_menor.rb` temos:

```
Bem vindo ao jogo da adivinhação
Qual é o seu nome?
Guilherme
Começaremos o jogo para você,
Guilherme
```

A saída está feia, com tudo muito grudado. Vamos separar o momento que o nome foi lido do resto do programa, onde notificamos o usuário que começaremos o jogo. Primeiro colocamos algumas linhas em branco, isto é, não imprimimos nada, somente uma quebra de linha:

```
puts "Bem vindo ao jogo da adivinhação"
puts "Qual é o seu nome?"
nome = gets
puts
puts
puts
```

```
puts
puts
puts
puts "Começaremos o jogo para você, "
puts nome
```

E a saída agora fica um pouco melhor:

```
Bem vindo ao jogo da adivinhação
Qual é o seu nome?
Guilherme




Começaremos o jogo para você,
Guilherme
```

String e concatenação

A saída ainda está feia, gostaríamos de jogar na tela a mensagem `Começaremos o jogo para você, Guilherme`. Para isso precisamos juntar dois textos, o primeiro é o `nome`. Como fazer isso? Em linguagens de programação em geral, chamamos de *String* um valor que é um conjunto de caracteres, como uma palavra, um texto, uma placa de carro, etc. Portanto, queremos juntar duas *Strings*, uma depois da outra. Como fazer isso? Usaremos a soma de duas `Strings`, chamada de *concatenação*:

```
puts "Começaremos o jogo para você, " + nome
```

Ficando com o código final:

```
puts "Bem vindo ao jogo da adivinhação"
puts "Qual é o seu nome?"
nome = gets
puts
puts
puts
puts
puts
puts
puts "Começaremos o jogo para você, " + nome
```

Agora sim nossa saída é bonita:

```
Bem vindo ao jogo da adivinhação
Qual é o seu nome?
```

```
Guilherme
```

```
Começaremos o jogo para você, Guilherme
```

Escolhendo um número e lendo o chute do jogador

Desejamos escolher um número secreto. Nesse instante deixaremos um número fixo e depois alteramos para que cada vez o número tenha um valor diferente. Primeiro imprimimos a mensagem de anúncio do sorteio, algo que já conhecemos:

```
puts "Escolhendo um número secreto entre 0 e 200..."
```

Depois queremos definir um novo valor. O valor `175` será nosso número secreto, então falo que `numero_secreto` deve receber o valor `175`:

```
puts "Escolhendo um número secreto entre 0 e 200..."
numero_secreto = 175
puts "Escolhido... que tal adivinhar hoje nosso número secreto?"
```

Ficando com o resultado:

```
Bem vindo ao jogo da adivinhação
Qual é o seu nome?
Guilherme
```

```
Começaremos o jogo para você, Guilherme
Escolhendo um número secreto entre 0 e 200...
Escolhido... que tal adivinhar hoje nosso número secreto?
```

Precisamos agora perguntar um número, um chute, que o usuário deseja escolher. Já sabemos ler essa informação, portanto fazemos:

```
puts
puts
puts
puts
puts "Tentativa 1"
```

```
puts "Entre com o numero"
chute = gets
```

Mas o que fazer com o chute? Primeiro avisamos o usuário que processaremos seu chute, será que ele acertou? Novamente, é o código que estamos começando a nos acostumar: `puts` passando *Strings* e concatenações:

```
puts
puts
puts
puts
puts "Tentativa 1"
puts "Entre com o numero"
chute = gets
puts "Será que acertou? Você chutou " + chute
```

Testamos nosso programa, chutando o número 100:

```
Bem vindo ao jogo da adivinhação
Qual é o seu nome?
Guilherme




Começaremos o jogo para você, Guilherme
Escolhendo um número secreto entre 0 e 200...
Escolhido... que tal adivinhar hoje nosso número secreto?




Tentativa 1
Entre com o numero
100
Será que acertou? Você chutou 100
```

Operador de comparação e o *boolean*

Estamos prontos para verificar se o usuário acertou ou errou. Como? Queremos verificar se o valor chutado é igual ao número secreto. Qual o símbolo matemático para igualdade? O igual, `=` . Mas já o usamos antes para dizer que um valor estava sendo utilizado. Como então *verificar se um valor é igual a outro*, fazemos uma *comparação*? Usaremos o `==` . Será que o "chute" é igual ao número secreto escolhido?

Por exemplo, será que 175 é igual igual a 175?

```
puts 175 == 175
```

Levando em consideração que verdadeiro é *true* em inglês, a saída será:

```
true
```

E será que ele é igual a 174?

```
puts 175 == 175
puts 175 == 174
```

Levando em consideração que falso é *false* em inglês, a saída será:

```
true
false
```

Isto é, o operador `==` realmente compara a igualdade entre dois números. Os valores "verdadeiro" e "falso", "*true*" e "*false*" são chamados de *booleanos*.

Agora podemos verificar se o número chutado é igual a `175`:

```
puts 175 == chute
```

Ficando com o programa:

```
puts "Bem vindo ao jogo da adivinhação"
puts "Qual é o seu nome?"
nome = gets
puts
puts
puts
puts
puts
puts
puts "Começaremos o jogo para você, " + nome
puts "Escolhendo um número secreto entre 0 e 200..."
numero_secreto = 175
puts "Escolhido... que tal adivinhar hoje nosso número secreto?"
puts
puts
puts
puts
puts "Tentativa 1"
puts "Entre com o numero"
chute = gets
puts "Será que acertou? Você chutou " + chute
puts 175 == chute
```

E eu errei:

```
...
Tentativa 1
Entre com o numero
100
Será que acertou? Você chutou 100
false
```

Conversão de *String* em número inteiro

Mas e se eu chutar "175"?

```
...
Tentativa 1
Entre com o numero
175
Será que acertou? Você chutou 175
false
```

O que aconteceu? "175" é igual a "175", como já sabíamos, mas ele imprimiu `false`. Mas vamos lembrar que o "175" que o usuário entrou como informação, como seu chute, é um texto. O texto `"175"` não é igual igual ao número `175`, são duas coisas totalmente diferentes. Antes mesmo de analisar seus valores, eles são de tipos diferentes: um é uma *String*, o outro é um *Integer* (número inteiro). Veja a diferença:

```
puts "175" == 175
puts "175" == "175"
puts 175 == 175
```

O resultado:

```
false
true
true
```

Queremos, então, converter nosso chute para um número inteiro (*to an integer*), e adivinha? Existe um método chamado `to_i` que converte a *String* para um inteiro:

```
puts "175".to_i == 175
```

Resultando em:

```
true
```

Mas na verdade 175 é nosso `numero_secreto`, e queremos compará-lo com o `chute` portanto:

```
puts chute.to_i == numero_secreto
```

Terminando a primeira parte de nosso jogo com o código:

```
puts "Bem vindo ao jogo da adivinhação"
puts "Qual é o seu nome?"
nome = gets
puts
puts
puts
puts
puts
puts
puts "Começaremos o jogo para você, " + nome
puts "Escolhendo um número secreto entre 0 e 200..."
numero_secreto = 175
puts "Escolhido... que tal adivinhar hoje nosso número secreto?"
puts
puts
puts
puts
puts "Tentativa 1"
puts "Entre com o numero"
chute = gets
puts "Será que acertou? Você chutou " + chute
puts numero_secreto == chute.to_i
```

Agora sim nosso jogo de uma tentativa funciona: ou você acerta, ou você erra.

Avançar