1. **08 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 para o 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 língua 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 experiência. Não é à 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 mensagens:

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.

# 09 Instalação do ruby

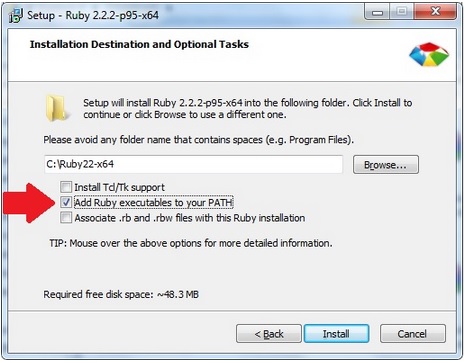
**Windows:**

1 - Comece a instalar o Ruby, de acordo com a versão do seu Windows:

<http://rubyinstaller.org/downloads>

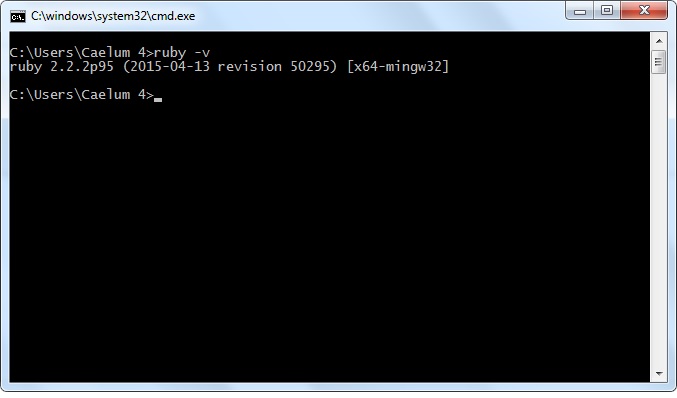
2 - Marque a opção "Add Ruby executable to your PATH" e prossiga a instalação:

2.1 - Marque também a opção *Associate .rb and .rbw files with this Ruby Installation*.



3 - Ao término da instalação, verifique se foi corretamente instalando indo no prompt/terminal e dando o comando:

ruby -v

4 - Deve aparecer a versão que foi instalada do Ruby, algo como: 

Possíveis problemas:

Caso no passo **4** o prompt/terminal retornou uma mensagem como: "ruby is not recognized as an internal or external command (...)" Isso quer dizer que o Ruby não foi instalado corretamente, certifique-se que você:

• instalou a versão de acordo com o seu sistema operacional e sua respectiva versão;

• marcou a opção "Add Ruby executable to your PATH" (passo 2).

**Linux**

Você pode instalar o Ruby com o seguinte comando no terminal:

\curl -sSL https://get.rvm.io | bash -s stable

Se a sua distribuição de Linux possui o gerenciador apt-get, pode utilizar alternativamente:

sudo apt-get install ruby

Depois dê o comando "ruby -v" para ver se ele está instalado corretamente.

**Mac**

O Ruby já vem instalado, mas verifique se está tudo ok dando o comando "ruby -v" no terminal.

# 10 Nosso primeiro programa

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.

Criamos um arquivo chamado maior\_ou\_menor.rb.

ruby maior\_ou\_menor.rb

Diga para o computador imprimir a mensagem "Bem-vindo ao jogo da adivinhação":

puts "Bem-vindo ao jogo da adivinhação"

**Exercício**

Queremos perguntar ao usuário qual é o nome dele, para personalizarmos sua experiência.

Leia o valor na variável nome.

Imprima uma mensagem que começaremos o jogo para o usuário, e mostre o nome dele, como em: Começaremos o jogo para você, Guilherme.

Lembre-se de pular algumas linhas entre a entrada do nome e o resultado final.

### Opinião do instrutor

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 para o 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 língua 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": Caso apresente algum problema na ortografia por causa dos acentos e caracteres especiais do Português acrescente ´"# encoding: UTF-8"`na primeira linha, assim seus problemas com a codificação da língua portuguesa acabam

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.

No nosso caso, queremos perguntar ao usuário qual é o nome dele, para personalizarmos sua experiência. Não é à 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 (get) 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 mensagens:

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

Que saída horrível. 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

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 que 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 ::String::, 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

# 11 Entrada e saída básica

Crie um arquivo chamado maior\_ou\_menor.rb. O conteúdo dele? Vazio.

Dado esse programa, vazio, rode o mesmo:

ruby maior\_ou\_menor.rb

Adicione instruções do tipo puts e gets para imprimir as mensagens a seguir para o usuário, ao pedir o seu nome:

Bem-vindo ao jogo da adivinhação

Qual é o seu nome?

Guilherme

Começaremos o jogo para você, Guilherme

Lembre-se que o puts é usado como em:

puts "Mensagem que desejo imprimir"

E o gets para ler:

nome = gets

### Opinião do instrutor

# Jogo da adivinhação

## O Jogo: entrada e saída básica

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 língua 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.

No nosso caso, queremos perguntar ao usuário qual é o nome dele, para personalizarmos sua experiência. Não é à 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 (get) 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 mensagens:

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

Que saída horrível. 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

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 que 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 ::String::, 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

# 12 Lendo um chute

1 - **Desejamos escolher um número secreto.**

Mostre que está "Escolhendo um número secreto entre 0 e 200..." e fixe o numero\_secreto como 175 por enquanto.

Imprima a mensagem que foi "Escolhido... que tal adivinhar hoje nosso número secreto?".

2 - **Precisamos agora perguntar um número, um chute, que o usuário deseja escolher.**

Peça pela "Tentativa 1", "Entre com o número" e leia o chute do usuário.

Após ler o chute, mostre a mensagem com o chute que ele acabou de fazer, como no exemplo a seguir, onde chutei 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 número

100

Será que acertou? Você chutou 100

### Opinião do instrutor

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 número"

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 número"

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 número

100

Será que acertou? Você chutou 100

# 13 Operador ==

Vamos praticar um pouco!

1 - Crie um novo arquivo de teste.rb.

2 - Nele use o puts para testar se o operador == funciona comparando 175 com 174.

E 175 com 175?

puts 175 == 175

puts 175 == 174

### Opinião do instrutor

## Será que acertou? O operador de comparação ==

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 usamos o = antes para dizer que um valor estava sendo utilizado. Como então verificar se um valor é igual a outro? Usaremos o ==. Será que o chute é igual 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 é:

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 é:

true

false

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

# 14 Tipos e comparações

Tente usar agora o operador == para comparar o chute do usuário com o número secreto escolhido.

Imprima true se ele acertou, ou false caso contrário.

Teste seu jogo com um número errado, como 100.

1. O que acontece?
2. E com o número certo, 175?
3. O que aconteceu?
4. Qual o motivo disso ter acontecido?

### Opinião do instrutor

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 número"

chute = gets

puts "Será que acertou? Você chutou " + chute

puts 175 == chute

E eu errei:

...

Tentativa 1

Entre com o número

100

Será que acertou? Você chutou 100

false

Mas e se eu chutar 175?

...

Tentativa 1

Entre com o número

175

Será que acertou? Você chutou 175

false

O que aconteceu? 175 é igual igual a 175, como já sabíamos, mas ele imprimiu false. Acontece que o número 175 é igual ao número 175, verdade. Mas lembra que o texto 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. Um é um texto, outro é um número. Antes mesmo de analisar seus valores, eles são duas coisas de tipos diferentes, seus tipos são diferentes: um é uma String, o outro é um Int (número inteiro).

puts "175" == 175

puts "175" == "175"

puts 175 == 175

Com o resultado:

false

true

true

# 15 Transformando string em número inteiro

Use o método to\_i para converter uma String em um número inteiro (integer).

Para isso pegue nosso chute (uma String lida do usuário) e a transforme em um número (integer).

Teste seu jogo, que agora deve funcionar.

Compartilhe seu código aqui.

### Opinião do instrutor

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 número"

chute = gets

puts "Será que acertou? Você chutou " + chute

puts 175 == chute.to\_i

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

# 16 Operadores 1

**Exercício**

1. Dê exemplo de variáveis que usamos até agora e de valores válidos para elas.
2. Qual o operador que usamos para atribuir um valor a uma variável?

### Opinião do instrutor

## Operador de atribuição e variáveis

Vimos que podemos utilizar o = para atribuir um valor a algo.

Por exemplo, utilizamos nome = gets para ler do usuário um texto.

A esse algo damos o nome de variável. No nosso caso a variável nome recebe o valor que foi lido do jogador.

No nosso jogo temos três variáveis: o chute, o numero\_secreto e o nome do jogador.

Cada variável possui um valor. O numero\_escolhido está fixo como 175, enquanto tanto o nome do jogador enquanto o chute são valores lidos a partir da entrada do usuário (via gets).

# 17 Operador de igualdade e diferença

Dois operadores muito comuns de comparação são o igual (==) e o diferente (!=). Faça uso deles para verificar se diversos números são iguais ou diferentes.

### Opinião do instrutor

## Operadores de comparação

Como o símbolo = foi utilizado pela linguagem para atribuir um valor a uma variável, teremos que adotar algum outro símbolo para a comparação de valores. O == é o padrão mais comum adotado pelas linguagens de programação para tal situação. Como vimos antes, o código a seguir imprime verdadeiro e depois falso:

puts 175 == 175

puts 175 == 174

Mas será que existem outros comparadores? Outro muito comum é o diferente, isto é, não (!) igual (=), como no exemplo a seguir, que imprime verdadeiro duas vezes:

puts 175 != 375

puts 175 != 174

# 18 Operadores de comparação

Use os operadores >, <, >= e <= para fazer comparações entre números, o que cada um significa?

### Opinião do instrutor

Outros operadores utilizados comumente em operações matemáticas são os de maior (>), menor (<), maior ou igual (>=) ou menor ou igual (<=). Os exemplos a seguir todos imprimem verdadeiro:

puts 175 > 174

puts 174 < 175

puts 170 >= 170

puts 175 >= 170

puts 170 <= 170

puts 170 <= 175

# 19 Entrada e saída

O que é entrada e saída? Dê exemplos de dispositivos de entrada e saída em um computador moderno (seja ele desktop, notebook, um celular ou até mesmo um relógio ou uma TV!).

### Opinião do instrutor

## Entrada e saída

Aprendemos a ler do teclado (entrada) invocando o gets, e a imprimir resultado na saída com o puts. Entrada e saída é vital para um programa pois é a maneira padrão dele se comunicar com o mundo exterior.

Por enquanto a leitura é feita do teclado (na verdade, da entrada padrão), e a saída para o console (na verdade, a saída padrão). No decorrer deste material veremos como ler de e escrever para um arquivo, por exemplo.

No futuro é possível, por exemplo, ler dados ou enviar informações para a internet, ler dados de um joystick ou se comunicar com uma caixa de som via bluetooth. Tudo isso é entrada e saída: o joystick é uma entrada de dados, uma caixa de som é uma saída.

# 20 Refatoração

Em dois momentos do nosso jogo usamos uma série de invocações ao puts para imprimir cada uma das linhas em branco, o que ficou bem estranho. Refatore, altere a implementação sem alterar o resultado final, o nosso código para que ele use o caractere especial \n para imprimir linhas em branco. Compartilhe o código final aqui.

### Opinião do instrutor

## Refatoração: \n

Se nosso objetivo é escrever código melhor, para aqui Guilherme. Você escreveu seis puts só para pular seis linhas, poxa. Verdade. É hora de parar, olhar para trás e melhorar nosso código, o processo que chamamos de refatorar. Queremos alterar nossas seis linhas de puts para um código mais simples que alcance o mesmo resultado.

O que queremos é imprimir seis linhas em branco? Seria muito legal se fosse possível então escrever:

puts "proxima\_linha,proxima\_linha,proxima\_linha,proxima\_linha,proxima\_linha,proxima\_linha"

Claro, Ruby não entende isso. Mas existe uma sequência mágica (não, não é magia) que significa uma nova linha, ::new line::, a sequência \n. Portanto podemos substituir nosso código por:

puts "\n\n\n\n\n\n"

O resultado é o mesmo. Se isso funciona, podemos substituir também o puts de quatro linhas por quatro \ns. Maravilha, nosso código agora fica mais simples:

puts "Bem-vindo ao jogo da adivinhação"

puts "Qual é o seu nome?"

nome = gets

puts "\n\n\n\n\n\n"

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 "\n\n\n\n"

puts "Tentativa 1"

puts "Entre com o número"

chute = gets

puts "Será que acertou? Você chutou " + chute

puts 175 == chute.to\_i

Refatoramos nosso código.