1 Pensamento Computacional

O QUÊ?

Pensamento computacional e noções básicas de linguagem de programação.

POR QUÊ?

Os computadores estão presentes em quase todas as atividades humanas na atualidade e, por isso, desenvolver o pensamento computacional e conhecer noções básicas de linguagens de programação pode ser fundamental para sua vida profissional, trajetória acadêmica e participação efetiva na sociedade.

Projeto: LIVRO ABERTO DE MATEMÁTICA

Cadastre-se como colaborador no site do projeto: umlivroaberto.com

Título:	Pensamento Computacional
Ano/ Versão:	2021 / versão 1.0 de 26 de julho de 2021
Editora	Instituto Nacional de Matemática Pura e Aplicada (IMPA-OS)
Realização:	Olimpíada Brasileira de Matemática das Escolas Públicas (OBMEP)
Produção:	Associação Livro Aberto
Coordenação:	Fabio Simas e Augusto Teixeira (livroaberto@impa.br)
Autor (a):	Leonardo Barichello
Revisão:	Diego Lieban Larissa Monzon Kátia Rocha
Design:	Andreza Moreira (Tangentes Design)
Diagramação:	Tarso Caldas
Capa:	Foto de Pankaj Patel no Unsplash https://unsplash.com/photos/yEAOfWSdzgM

Desenvolvido por

Licença:

CC BY NC SA

Patrocínio:

EXPLORANDO UM NOVO JEITO DE RESOLVER PROBLEMAS

Os computadores estão em todos os lugares, seja na forma de celulares, notebooks, computadores de mesa ou dispositivos com funções mais específicas, como máquinas de cobrança. Das atividades mais corriqueiras às atividades mais inovadoras, é possível identificar a influência de um computador, e na matemática não é diferente.

Nas últimas décadas, a comunidade de matemáticos teve que aprender a lidar com o uso intensivo de computadores em uma área do conhecimento que, por muito tempo, aceitava de forma quase única argumentos lógicos na forma textual. Um exemplo ocorreu com o conhecido **problema das quatro cores** que, em resumo, lançava a questão sobre quantas cores são necessárias para pintar um mapa sem que países vizinhos tenham a mesma cor.

Em 1852, uma resposta para esse problema foi sugerida por um matemático: 4 cores seriam suficientes. Porém, ele não demonstrou que essa resposta de fato estava correta. Mais de 100 anos se passaram até que um grupo de matemáticos conseguiu demonstrar que a tal resposta estava de fato correta, mas a demonstração era feita com o auxílio de computadores e tomava mais de mil horas de processamento nos computadores mais rápidos existentes naquele momento.

O que incomodou os matemáticos não foi apenas o tempo quase sobrehumano para checar essa solução, mas principalmente o questionamento sobre a validade ou não de uma demonstração que dependia de um computador para ser realizada.

VOCÊ SABIA?

Você pode saber mais sobre o problema das quatro cores lendo o texto Quatro cores bastam para colorir qualquer mapa.

O que essa história ilustra é que a disponibilidade de computadores faz mais do que agilizar certas tarefas repetitivas, mas cria a possibilidade de resolvermos problemas de maneira diferente do que poderíamos sem eles. A capacidade de utilizar um computador amplifica as nossas possibilidades como resolvedores de problemas.

Neste módulo, vamos desenvolver habilidades relacionadas ao que é chamado de pensamento computacional: as habilidades mentais envolvidas no processo de resolver problemas de maneira que um computador seja capaz de realizar essa resolução. Vamos partir de problemas que se parecem com problemas matemáticos que você já resolveu e buscar chegar até a criação de algoritmos que possam ser executados por um computador!

Boas instruções

Atividade 1

O proprietário de uma empresa que fabrica e vende bottons gostaria de criar uma promoção para incentivar a compra de grandes quantidades, uma vez que isso agiliza a produção. Na sua loja, todos os bottons são vendidos pelo mesmo preço: R$ $2,00$.

A primeira ideia foi oferecer um desconto de 10% no valor total da compra caso o comprador adquirisse mais do que 100 unidades.

a) Com essa promoção, qual seria o valor de uma compra de 95 bottons? E de uma compra de 105 bottons?

Como você deve ter notado, o problema dessa proposta é que algumas compras (com poucas unidades acima de 100) podem ficar mais baratas do que compras que não atingiram o valor que as qualifica para o desconto.

Uma outra ideia do proprietário foi a seguinte: cada unidade além da centésima recebe 15% de desconto no seu valor. Por exemplo, se alguém comprar 105 bottons, o comprador paga o preço normal para 100 deles e depois recebe 15% de desconto no valor dos outros 5.

b) Com essa promoção, qual seria o valor de uma compra de 95 bottons? E de 105? E de 200 bottons?

c) Você acha que essa ideia evita o problema de compras com pouco mais de 100 bottons ficarem mais baratas do que compras com quantidades menores? Justifique a sua resposta.

O proprietário da empresa decide adotar essa segunda ideia, mas notou que ela cria um problema: o cálculo do valor final da compra não pode mais ser feito de maneira imediata na calculadora.

d) Descreva com as suas palavras como um funcionário deve proceder para calcular o valor de uma compra a partir da informação de quantos bottons foram comprados. Dê a sua resposta na forma de um bilhete que será lido por uma pessoa que você não irá encontrar pessoalmente.

e) Troque o seu bilhete com um colega e discuta com ele se vocês seriam capazes de compreender como o preço de uma compra é calculado se vocês tivessem apenas lido o bilhete escrito por cada um.

Uma resolução boa de repetir

Atividade 2

Fonte: Olimpíada Brasileira de Informática

O cometa Halley é um dos cometas de menor período do Sistema Solar, completando uma volta em torno do Sol a cada 76 anos. Na última ocasião em que ele ficou visível do planeta Terra, em 1986, várias agências espaciais enviaram sondas para coletar amostras de sua cauda e assim confirmar teorias sobre suas composições químicas.

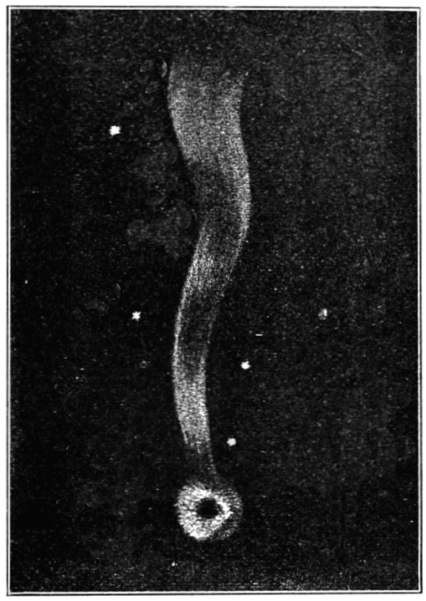


Figura 1.1: Registro da passagem do cometa Halley em 1682

a) 1) Uma pessoa que nasceu em 2020 vai ter a oportunidade de avistar o cometa Halley pela primeira vez em que ano? E uma que venha a nascer no ano 2200? E no ano 3000?

b) Qual foi o ano em que o cometa Halley pode ter sido avistado pela primeira vez para uma pessoa que nasceu em 1900? E uma que nasceu em 1500?

c) Em uma folha de papel à parte, descreva como obter a resposta para questões como as anteriores para um amigo fictício que tenha terminado o Ensino Médio, mas não tenha resolvido essas questões. Tenha em mente que o objetivo não é explicar como resolver o problema, mas descrever um procedimento que permita ao seu colega obter respostas. Use texto, expressões matemáticas, esquemas visuais ou outros recursos que possam ajudar a deixar as suas instruções o mais claras possível.

d) Troque as instruções com um colega, e seguindo os passos que ele descreveu, obtenha o ano do primeiro possível avistamento do cometa Halley para alguém que tenha nascido nos anos: 2500, 1000 e 2290.

e) Antes de devolver a resolução para o seu colega, avalie-a considerando os seguintes aspectos: Ele seguiu a mesma estratégia que você? Na sua opinião, qual dos dois conjuntos de instruções é mais fácil de seguir e porquê?

ORGANIZANDO BOAS INSTRUÇÕES

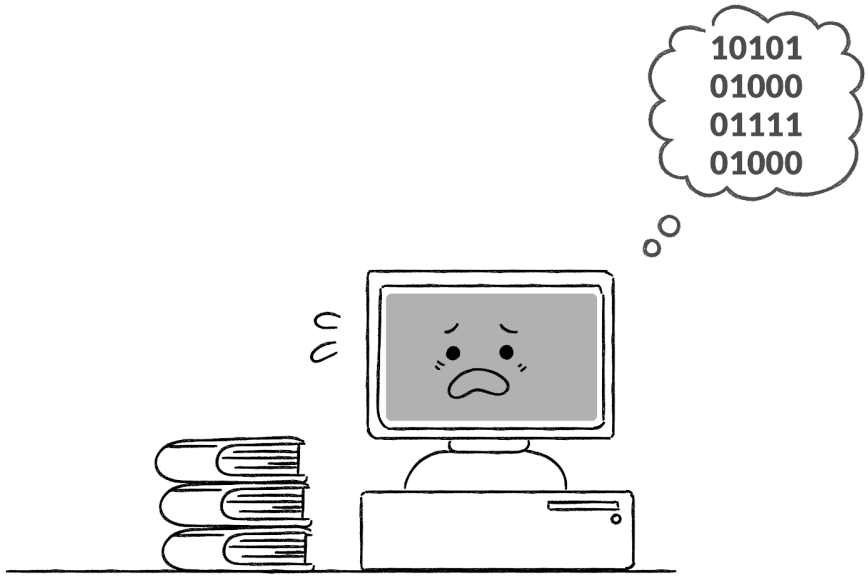


Figura 1.2: Fonte:Computação Desplugada

As atividades anteriores mostraram que não é simples criar um conjunto de instruções que resolve um determinado problema de modo que uma pessoa seja capaz de compreendê-las e segui-las sem poder fazer mais perguntas ou pedir esclarecimentos. Por isso é importante que fique claro tanto o significado de cada instrução, quanto a ordem em que elas devem ser executadas.

No caso de criarmos instruções para um computador seguir (um conjunto finito de instruções com fluxo bem definido também pode ser chamado de algoritmo), essa exigência fica ainda maior, pois o computador não consegue usar um pouco de bom senso ou conhecimentos adicionais sobre a situação para complementar as instruções ou tomar decisões, caso isso seja necessário. Para um computador, todas as instruções devem ser dadas de modo que ele seja capaz de executar e o fluxo das instruções deve estar absolutamente claro na descrição.

EXPLORANDO A REPRESENTAÇÃO VIA FLUXOGRAMAS

Ao longo deste módulo, vamos conhecer várias ferramentas que nos permitam criar instruções que possam ser realizadas por um computador. Afinal, estamos procurando desenvolver habilidades e ferramentas que nos permitam criar algoritmos, e algoritmos podem ser usados para criar programas e aplicativos que realizem certas tarefas para nós!

Uma dessas ferramentas é o **fluxograma**. Trata-se de uma forma de representar visualmente a maneira como as instruções (ou comandos) se relacionam e qual a ordem em que devem ser realizadas (ou executados).

Você já deve ter visto esquemas visuais como o mostrado abaixo, que esquematiza o processo, de decisão nesse caso, sobre a pertinência de compartilhar uma notícia.

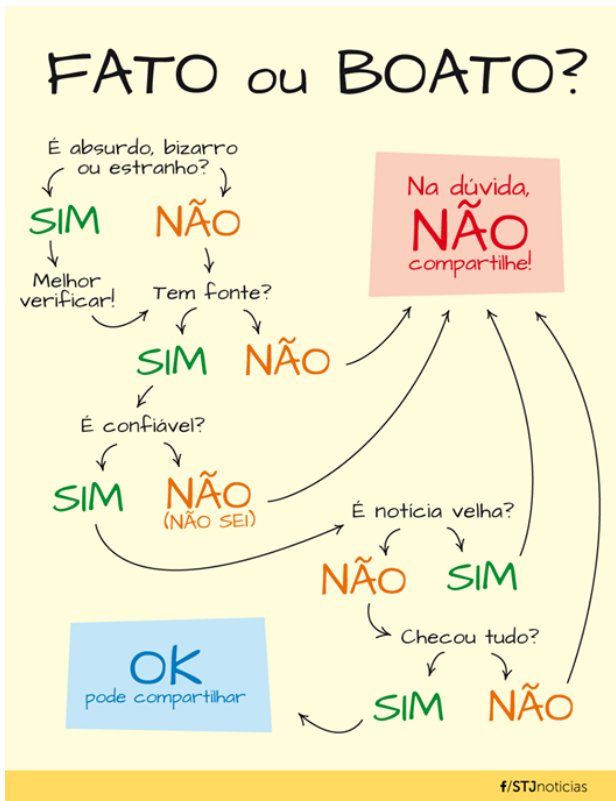


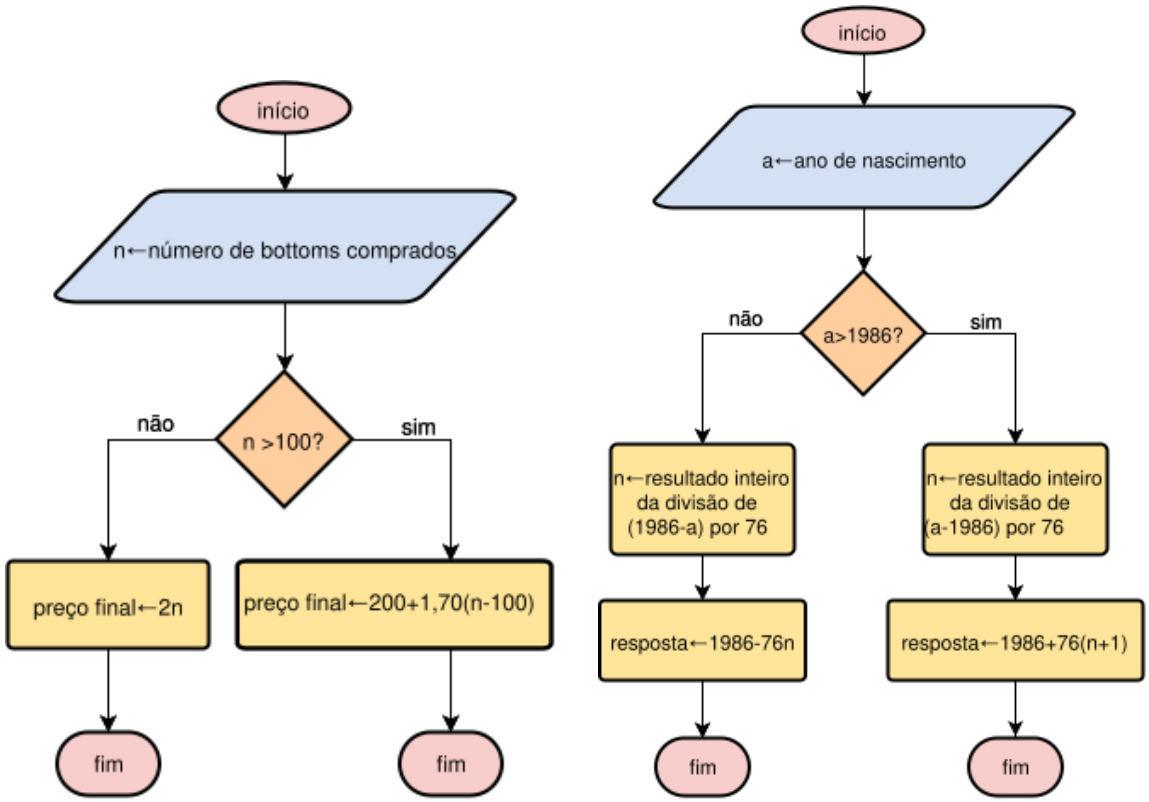
Figura 1.3: campanha do Superior Tribunal de Justiça

Neste módulo, faremos um uso mais específico e rigoroso desse recurso, mas a intenção é a mesma da imagem acima: representar visualmente um processo com múltiplas etapas e ações.

Fluxogramas

Atividade 3

A imagem a seguir mostra dois fluxogramas que resolvem as atividades anteriores. Leia-os com atenção e, se necessário, volte às atividades 1 e 2 para relembrar.



a) Utilize os fluxogramas para resolver os problemas das atividades 1 e 2 para os seguintes casos:

- **i)** Qual seria o valor da compra para 60 bottons? E para 150?
- **ii)** Em que ano o cometa Halley poderá ser avistado pela primeira vez por alguém que tenha nascido no ano de 2300?

b) Descreva a sua solução para a Atividade 2 na forma de um fluxograma.

PARA REFLETIR

O que você achou dos fluxogramas como forma de apresentar um conjunto de instruções?

ORGANIZANDO FLUXOGRAMAS

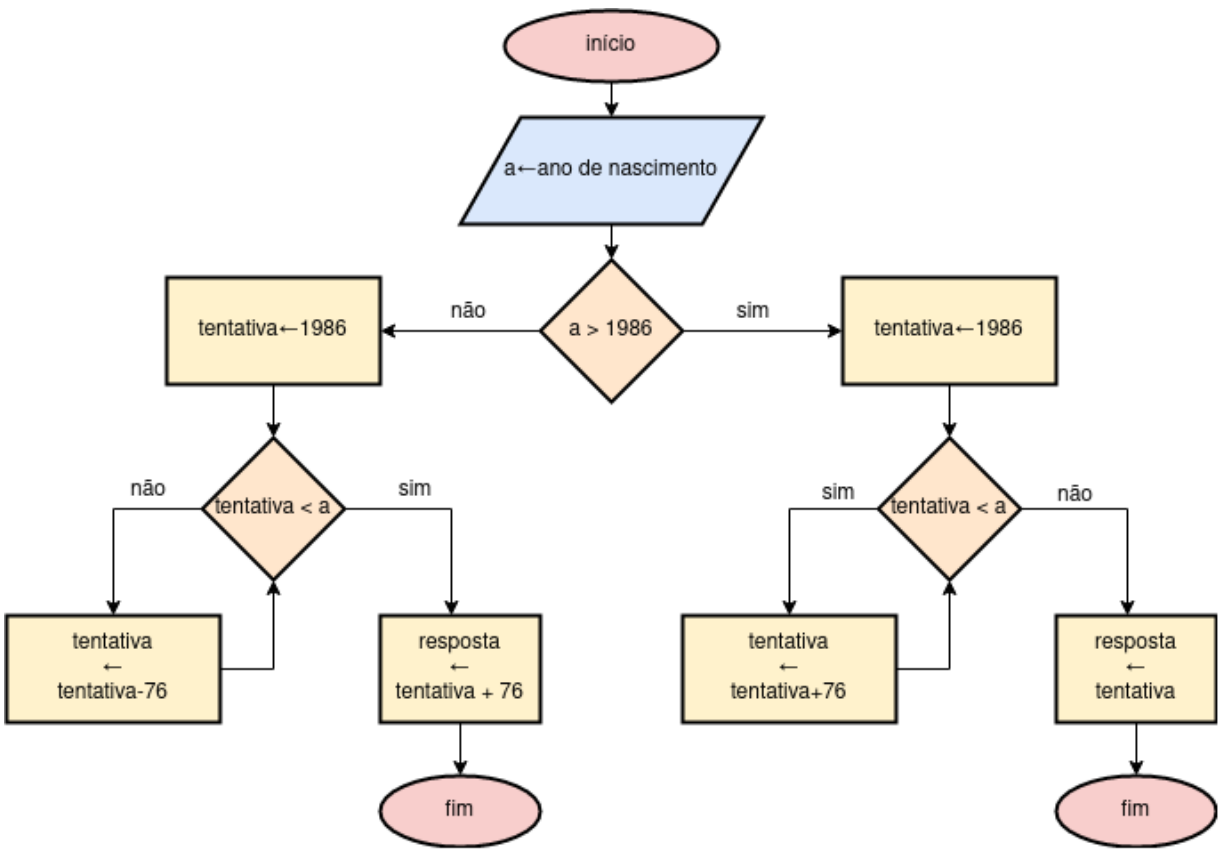
Os fluxogramas são uma maneira de representar algoritmos que fazem uso de um arranjo visual para deixar claro o fluxo, ou seja, a sequência em que comandos devem ser realizados. Embora seja conveniente em muitas situações, eles não são recomendados para algoritmos muito complexos ou longos. Uma outra limitação dos fluxogramas é que eles não podem ser interpretados diretamente por um computador.

Mesmo assim, eles ainda podem ser uma forma rápida para organizar o seu raciocínio quando estiver pensando em um algoritmo. Por isso, consideramos que vale a pena mostrar mais um exemplo que ilustra uma situação não mostrada anteriormente: um processo repetitivo.

Considere a seguinte solução (descrita na forma textual) para a Atividade 2:

Se o ano de nascimento for maior que 1986, então some 76 a 1986 até que o resultado seja maior ou igual ao ano de nascimento. Quando isso ocorrer, o resultado da última soma é a resposta. Se o ano de nascimento for menor do que 1986, então subtraia 76 a 1986 até que o resultado seja menor que ano de nascimento. Quando isso ocorrer, some 76 ao resultado da última soma e essa será a resposta.

Como fluxograma, ela pode ser representada como mostrado a seguir.



Três coisas merecem atenção neste fluxograma. Primeiro, a variável tentativa, que não apareceu nos fluxogramas anteriores. Ela é usada para armazenar em qual ano estamos à medida que somamos ou subtraímos 76 até encontrar a resposta. Na descrição verbal não é necessário mencioná-la, mas em uma linguagem de programação precisamos armazenar todos os resultados que desejamos usar posteriormente em alguma variável. Em um fluxograma também poderíamos evitar o seu uso, se escrevêssemos algo como "some 76 ao ano", mas a prática de

armanzenar o resultado em uma variável é boa para não deixar dúvidas nas instruções seguintes.

Segundo, a ocorrência de ciclos: esses ciclos representam processos iterativos, ou seja, que envolvem repetição sucessiva, são repetitivos. Focando no que está mais à esquerda, ele diz o seguinte: cheque se a tentativa é menor do que a, se não for, subtraia 76 e volte para a checagem. Isso faz com que o algoritmo repita essa subtração até que o resultado encontrado seja menor do que a. No ciclo do lado direito, ocorre o mesmo mas com adições (veja que o lado direito do fluxograma é acionado quando a é maior do que 1986).

O terceiro aspecto que merece atenção são os comandos que são realizados dentro dos ciclos. Vamos focar novamente no que está mais à esquerda: tentativa = tentativa + 76. Note que essa expressão, do ponto de vista matemático, representa uma equação sem solução. Mas em grande parte das linguagens de programação, o sinal de igual significa "guarde o resultado da direita na variável à esquerda". Nesse caso, o computador primeiro calcula o valor do lado direito (somando 76 ao valor atual da variável tentativa) e então guarda na mesma variável (tentativa) o resultado (perdendo o valor anterior, o que não é um problema, pois já testamos e o descartamos).

É importante ter muito cuidado com expressões como "`t=t+76`". Em um contexto matemático, o sinal de igual nessa expressão significa uma equivalência entre os valores do seu lado direito e esquerdo. Nesse caso específico, a expressão é uma equação sem solução (pois não existe valor de t que satisfaça a igualdade proposta).

Em linguagens de programação, o sinal de igual é normalmente usado para designar uma atribuição: a variável à esquerda do sinal deve receber o valor à direita. Inclusive, nesse contexto, a expressão `t+76=t` é totalmente diferente de `t=t+76` (na verdade, a primeira expressão está incorreta pois o conteúdo à esquerda do sinal de igual não é uma variável). Em algumas linguagens, a atribuição não é feita com o símbolo `=`, mas com `:=` ou `<-` (imitando a seta ←).

Esse uso computacional do símbolo = deve ser evitado fora do contexto de algoritmos, pois provavelmente será interpretado em termos matemáticos.

Processos repetitivos são muito comuns em algoritmos e voltaremos a eles em breve.

PARA SABER + INSTRUÇÕES PARA UM COMPUTADOR

Para que um computador seja capaz de seguir instruções que realizem alguma tarefa, é necessário que o algoritmo seja escrito em uma linguagem de programação. Uma linguagem de programação nada mais é do que uma maneira muito estruturada de descrever um algoritmo. A necessidade de clareza e estrutura para que um computador consiga interpretar um algoritmo faz com que certos elementos aparentemente estranhos sejam usados para organizar o fluxo das instruções.

VOCÊ SABIA?

Existem centenas de linguagens de programação em uso na atualidade, cada uma com certas vantagens e desvantagens e com diferentes níveis de popularidade. Atualmente, Python é uma linguagem muito utilizada em vários contextos e conta com muitos materi-

ais para autoestudo na internet. Javascript é um outro exemplo, mas seu uso é mais comum em páginas de internet. A linguagem C é muito usada, mas normalmente para programas mais técnicos que precisem de alto rendimento.

A imagem abaixo mostra dois algoritmos escritos em uma linguagem de programação chamada **Portugol**, que tem seus comandos em português. Apesar de não ser usada comercialmente, essa é a linguagem que sugerimos para você neste capítulo.

As instruções descritas abaixo em Portugol são as mesmas descritas anteriormente com fluxogramas. Leia com calma e tente compreender como as duas representações se relacionam.

```
programa {
    funcao inicio() {
        inteiro n
        real preco
        leia(n)
        se (n<=100) {
            preco=2*n
        }
        senao {
            preco=200+1.70*(n-100)
        }
        escreva(preco)
    }
}
```

```
programa {
    funcao inicio() {
        inteiro a, n, resposta
        leia(a)
        se (a<=1986) {
            n = (1986-a)/76
            resposta = 1986-76*n
        }
        senao {
            n = (a-1986)/76
            resposta = 1986+76*(n+1)
        }
        escreva(resposta)
    }
}
```

Você pode acessar um ambiente online que permite a execução destes algoritmos em `portugol-webstudio.cubos.io`. Caso você não esteja familiarizado com esse tipo de recurso, sugerimos o vídeo `youtu.be/6OIADpFImtc` como ponto de partida.

EXPLORANDO ALGORITMOS PARA PROBLEMAS MATEMÁTICOS

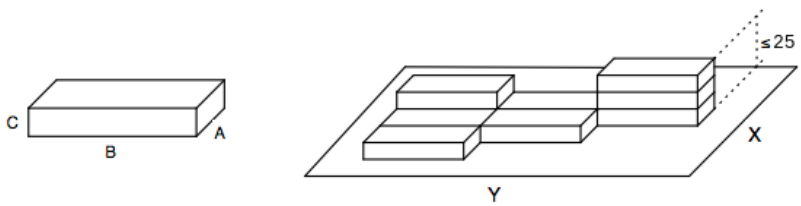
Dois recursos são muito comuns na criação de algoritmos, não importando se descrevemos na forma textual, com fluxograma ou em uma linguagem de programação: condicionais e repetições. Além desses dois recursos, o conceito de variável é especialmente importante quando pensamos em linguagens de programação, seja ela qual for.

Já conhecemos esses três elementos nas atividades anteriores e nas atividades a seguir vamos ter a chance de explorá-los com mais profundidade.

Contêineres

Atividade 4

O transporte em grandes navios de carga se dá através de contêineres, que são caixas metálicas grandes, dentro das quais as empresas acomodam seus produtos da maneira que desejar. Porém, a maneira como os contêineres são colocados nos navios costuma ser determinada pelo sistema de carregamento disponível nos portos. Imagine um porto que, por restrição nos equipamentos disponíveis, carregue seus contêineres todos alinhados, na mesma direção e com os lados paralelos aos lados da área de carregamento do navio, como mostrado na figura abaixo.



A administradora desse porto está com problemas para determinar quantos contêineres de dimensões A, B e C (em metros) podem ser colocados em um navio que tenha área de carregamento nas dimensões X e Y (em metros). Observe que a dimensão A dos contêineres deve ser carregada paralelamente à dimensão X dos contêineres, o mesmo ocorre para as dimensões B e Y.

Além dessas restrições, há uma limitação de altura imposta pelas autoridades portuárias que diz que a pilha de contêineres não pode ultrapassar 25 metros de altura.

a) Quantos contêineres de dimensão (em metros) $A = 3$, $B = 4$, $C = 1$ cabem em um navio com área de carregamento com dimensões $X = 30$ e $Y = 60$?

b) Quantos contêineres de dimensão (em metros) $A = 4$, $B = 8$, $C = 2$ cabem em um navio com área de carregamento com dimensões $X = 30$ e $Y = 60$?

c) Descreva, com suas palavras, como determinar quantos contêineres de dimensões A, B e C cabem em um navio com espaço de carregamento igual a X e Y.

Um professor em cada van

Atividade 5

Uma escola costuma organizar passeios com frequência e para levar os alunos conta com vários motoristas de vans. Essas vans possuem 14 lugares, além do assento do motorista. Por questão de segurança, a direção da escola exige que sempre haja um professor em cada van, não importando o número de alunos. Felizmente, a escola tem contato com muitos motoristas de vans.

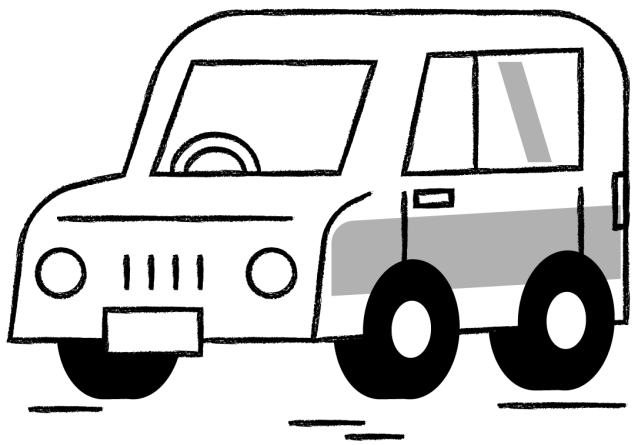


Figura 1.4: Fonte: Computação Desplugada

a) Quantos professores serão necessários para acompanhar os estudantes em um passeio em que 200 estudantes desejem participar? E em um passeio em que 65 estudantes desejem participar?

b) Descreva com suas palavras, e de forma clara, como obter a quantidade mínima de professores necessários para acompanhar um passeio em que n estudantes estejam interessados em participar.

Em algumas ocasiões, porém, há mais professores dispostos a acompanhar os estudantes do que o necessário. Em geral, a direção gosta dessa situação pois cada professor fica responsável por menos estudantes. Porém, já aconteceu de não haverem lugares disponíves nas vans para os professores adicionais, e a escola não pretende contratar vans adicionais.

c) Se houverem 200 estudantes interessados e 20 professores disponíveis, haverá lugar para todos os professores nas vans sem que seja necessário contratar vans adicionais? E se forem 75 estudantes e 10 professores?

d) Descreva com suas palavras e de forma clara como descobrir o número máximo de professores que podem acompanhar n estudantes em um passeio sem que seja necessário contratar mais vans.

ORGANIZANDO O COMPUTADOR SABE MENOS DO QUE VOCÊ

Para resolver as duas atividades anteriores você deve ter utilizado apenas algumas operações: soma, subtração, multiplicação, divisão e arredondamentos. As quatro primeiras estão diretamente disponíveis em qualquer linguagem de programação, mas a quinta delas nem sempre. Vamos discuti-la em mais detalhes.

Na atividade Um professor em cada van você deve ter utilizado o recurso de arredondar para cima para descobrir o número mínimo de professores necessários para acompanhar a turma. Por exemplo, se 30 alunos quiserem participar de uma viagem, calculamos . Logo, precisamos de mais de 2 professores, ou seja, 3. Matematicamente, dizemos que obtivemos o menor número inteiro maior ou igual ao resultado da divisão. Informalmente, dizemos que estamos "arredondando para cima". Porém, o Portugol (e outras linguagens de programação) não entende o comando "arredonde para cima" ou "obtenha o menor número inteiro maior ou igual". Então, como poderíamos descrever para o computador esse processo?

Para fazer isso, podemos utilizar duas operações que a maioria das linguagens de programação oferece: quociente da divisão (representada por "/") e resto da divisão entre dois números inteiros (representada por "%").

A operação / retorna um número real igual ao quociente da divisão se os valores envolvidos forem números reais, caso todos sejam números inteiros, ela retorna a parte inteira do quociente. Por exemplo, `7.2/4.8` terá como resultado 1.5 e `9/4` terá como resultado 2.

Já a operação % só pode ser realizada entre dois números inteiros e também retorna como resultado um número inteiro. Já . , `9%4` terá como resultado 1 e `7.2/4.8` terá como resultado 1.5 e a expressão `7.2%4` não será compreendida.

No caso do problema posto na atividade, devemos adicionar uma van se a divisão do número de interessados por 13 tiver resto maior do que zero. Em Portugol, uma possibilidade seria essa:

```
programa
{
    funcao inicio()
    {
        inteiro n, interessados,maximoprofs
        leia(interessados)
        n = interessados/13
        se (interessados%13 > 0) {
            n=n+1
        }
        maximoprofs = 14*n-interessados
        escreva("Sao necessários no mínimo ", n)
        escreva(" e podem ir no máximo ", maximoprofs)
    }
}
```

Veja que esse algoritmo primeiro considera que o número mínimo de professores é igual ao resultado inteiro (veja que a variável n é do tipo inteiro) da divisão do número de interessados por 13. Depois, se o resto da divisão por 13 for maior do que zero, o algoritmo soma 1 ao número mínimo de professores.

EXPLORANDO CONDICIONAIS

As sentenças que fazem uso do condicional (aqui chamadas simplesmente de condicionais) são um componente central em qualquer linguagem de programação e, na verdade, em boa parte dos recursos computacionais que utilizamos explícita ou implicitamente.

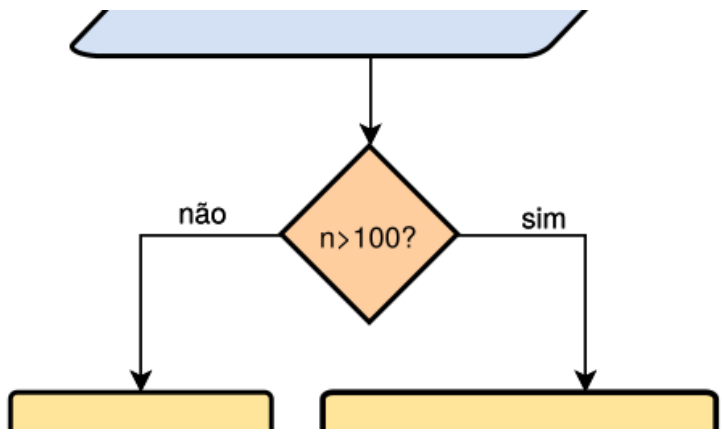
No caso das planilhas eletrônicas, por exemplo, elas podem ser usadas para que uma célula mostre um conteúdo que depende do conteúdo de uma outra célula. Um exemplo simples é mostrado abaixo. Imagine que um professor tenha uma planilha onde registra as notas dos seus estudantes em duas provas. Além de calcular a média das duas notas, ele deseja que a planilha mostre se cada estudante está, ou não, de recuperação de acordo com o seguinte critério: se a média for menor do que 6 então o estudante está de recuperação, senão está aprovado.

	A	B	C	D
1	**Prova 1**	**Prova 2**	**Média**	**Situação**
2	6,5	4,5	=(A2+B2)/2	=SE(C2<6;"Recuperação";"Aprovado")

	A	B	C	D
1	**Prova 1**	**Prova 2**	**Média**	**Situação**
2	6,5	4,5	5,5	Recuperação

Note que o comando descrito na frase "se a média for menor do que 6, então o estudante está de recuperação, senão está aprovado" é composto por três partes: uma condição ($C2 < 6$), o que deve ser feito se a condição for verdadeira (escrever a palavra "Recuperação" na célula) e o que deve ser feito se a condição for falsa (escrever "Aprovado" na célula). Isso está condensado no conteúdo da célula $D2$ mostrada acima.

Nas atividades anteriores, usamos o mesmo raciocínio para resolver o problema do cometa Halley (se o ano de nascimento fosse maior do que 1986 a resolução seguia por um caminho, se fosse menor seguia por outro) e o problema sobre o número de professores (se houvesse resto na divisão, era necessário contratar mais uma van). Quando representamos um algoritmo na forma de um fluxograma, condicionais são representadas como bifurcações no fluxo das instruções, de modo que um ou outro lado deve ser seguido conforme a condição estipulada.



A próxima atividade vai exigir um uso mais sofisticado de condicionais do que as atividades que resolvemos até agora.

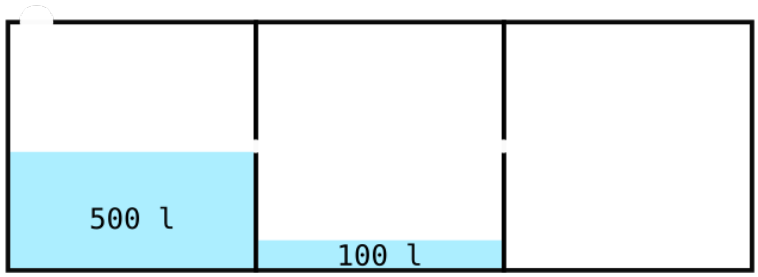
Cubos conectados

Atividade 6

Uma empresa que produz detergente transporta o produto bruto em cubos metálicos com lados iguais a 1 metro. Para facilitar o enchimento destes cubos, eles possuem uma abertura pequena, como mostrado na figura abaixo, que permite conectar dois cubos durante o enchimento. Assim, à medida que o líquido é despejado, ao atingir a altura da abertura, o líquido começa a escoar para o segundo cubo. Se houver líquido suficiente, o terceiro cubo também poderá receber uma parte.

Por questões de logística, a empresa atualmente enche sempre 3 cubos conectados por essas aberturas, e isso é feito despejando-se o líquido sempre a partir da tampa do cubo mais à esquerda, como mostrado na vista lateral abaixo.

Por exemplo, se forem injetados 600 litros de detergente, os cubos ficarão como mostrado a seguir, com 500 litros no primeiro, 100 litros no segundo e 0 no terceiro.



a) Quantos litros haverá em cada cubo se forem injetados 900 litros de detergente?

b) Quantos litros haverá em cada cubo se forem injetados 1250 litros de detergente?

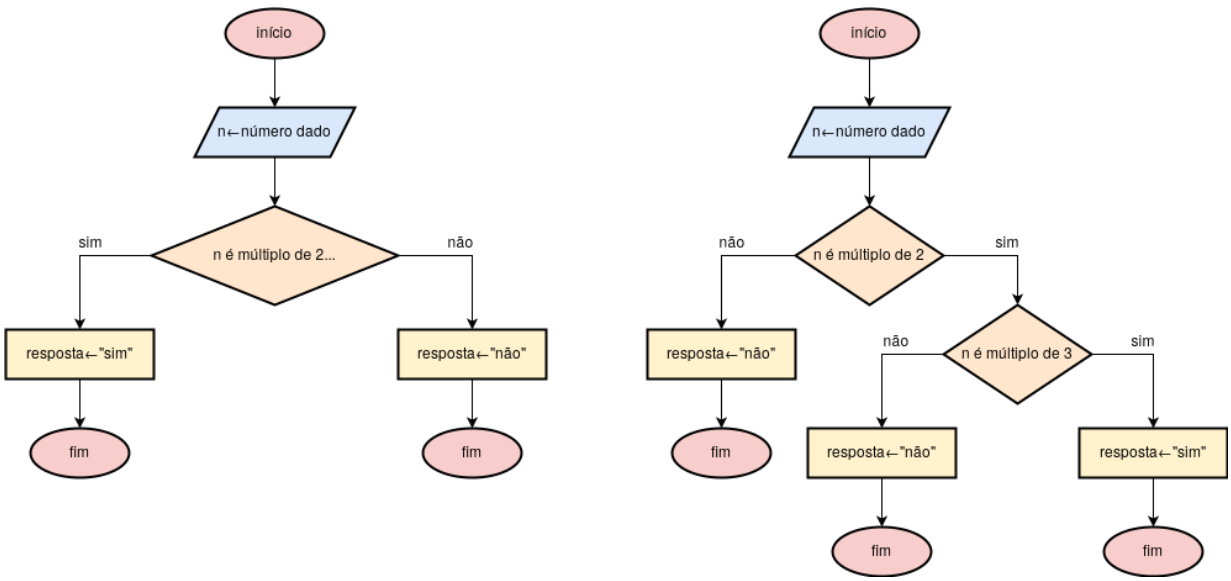
c) Quantos litros haverá em cada cubo se forem injetados 1800 litros de detergente?

d) Quantos litros haverá em cada cubo se forem injetados 2400 litros de detergente?

e) A empresa quer que você crie um algoritmo que permita saber qual é o volume de detergente em cada um dos três cubos quando uma quantidade N (em litros) é despejada no conjunto. Apresente a sua solução de maneira esquemática, isto é, sem usar muito texto.

ORGANIZANDO COMBINANDO CONDIÇÕES

Leia com atenção os dois fluxogramas abaixo. Ambos foram criados para determinar se um número dado é múltiplo de 6 a partir do seguinte resultado: um número é múltiplo de 6 se for múltiplo de 2 e de 3.

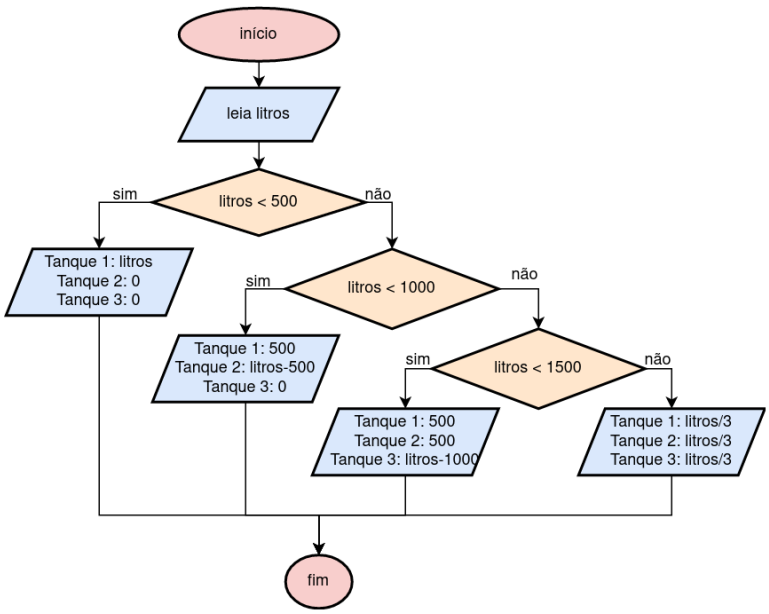


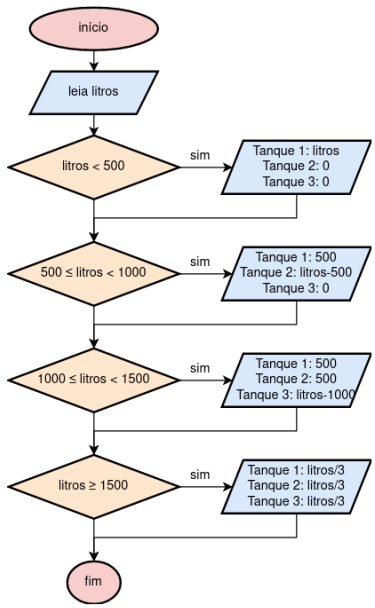
PARA REFLETIR

Os dois fluxogramas são visualmente diferentes, mas o que você pode dizer sobre os resultados que cada um deles produz para diferentes valores de N?

Veja que no fluxograma da esquerda, temos uma única condição e ela testa, de uma só vez, se N é múltiplo de 2 e se N é múltiplo de 3. No fluxograma da direita temos duas condições e a segunda está "dentro" da primeira, sendo acionada apenas se a primeira for verdadeira. Apesar de estruturalmente diferentes, ambas produzem os mesmos resultados para qualquer valor de N. Como ambas estão corretas, a escolha por uma ou outra solução deve ser feita por outros critérios, como a clareza para o leitor, os recursos disponíveis na linguagem de programação que você esteja usando ou até mesmo considerações sobre a eficiência de cada algoritmo em termos de velocidade de processamento computacional (mas essa discussão é muito mais avançada do que os nossos objetivos no momento).

Agora, vejamos dois exemplos de flouxogramas que resolvem a atividade Cubos Conectados corretamente e que são visualmente muito diferentes.





Note que no fluxograma da esquerda, as condições estão encadeadas. Se o total de litros colocados for menor do que 500, o fluxo do algoritmo passa pelo bloquinho azul mas à esquerda e depois já vai para o fim, sem sequer considerar as demais condições.

No fluxograma da direita, as quatro condições são sempre analisadas, não importando se alguma delas já foi satisfeita e produziu a resposta corretamente.

PARA REFLETIR

Você consegue pensar em algum argumento que permita concluir se um dos fluxogramas é melhor do que o outro?

PARA SABER + CONDICIONAIS

Condições em linguagens de programação costumam ser escritas com o comando se, que você já viu anteriormente. O uso de várias condicionais em sequência, ou de uma dentro da outra, acompanhado ou não pelo comento senao, permite criar muitas variações para um mesmo algoritmo, como vimos com fluxogramas para a atividade dos Cubos Conectados.

Tente usar o comando se para escrever um algoritmo em Portugol que se comporte como cada um dos dois fluxogramas mostrados acima. E você não precisa parar por aí: é possível criar outras soluções corretas e diferentes para o problema. Que tal tentar?

EXPLORANDO REPETIR E REPETIR

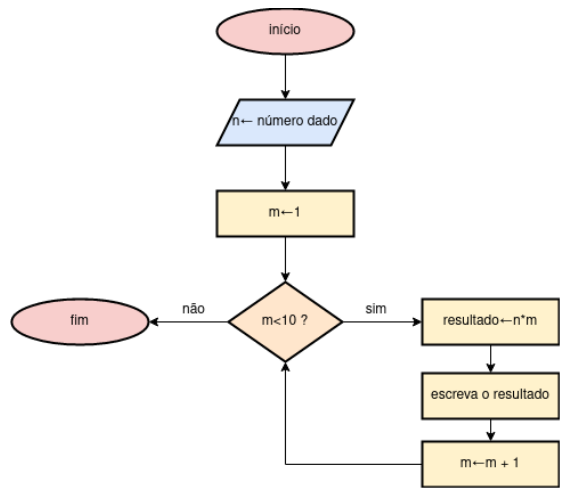
Repetir muitas vezes uma mesma ação é algo considerado cansativo para a maioria das pessoas. Porém, para muitas tarefas isso é necessário para se resolver algum problema ou atingir algum objetivo. Esse é um dos motivos que fazem dos computadores uma ferramenta tão útil: eles podem repetir comandos sem se cansar, sem perder a motivação ou a atenção e sem se distrair a ponto de cometerem erros. Historicamente, essa foi a motivação de diversos cientistas que, a partir do século 17, começaram a propor máquinas que fossem capazes de realizar cálculos aritméticos e que podem ser consideradas as avós dos computadores modernos.

VOCÊ SABIA?

As primeiras máquinas concebidas para realizar cálculos eram puramente mecânicas, ou seja, baseadas em engrenagens e encaixes físicos. Pesquise na internet pelas máquinas desenvolvidas pelos matemáticos Blaise Pascal e Gottfried Leibniz.

Nas próximas atividades, vamos lidar com problemas que envolvem muitas repetições e que ainda são muito relevantes na atualidade. Para isso, observe como o ciclo apresentado no fluxograma abaixo é representado em Portugol.

```
programa
{
    funcao inicio()
    {
        inteiro n, m, resultado
        leia(n)
        m=1
        enquanto (m<10) {
            resultado = m*n
            escreva(resultado + "\n")
            m=m+1
        }
    }
}
```



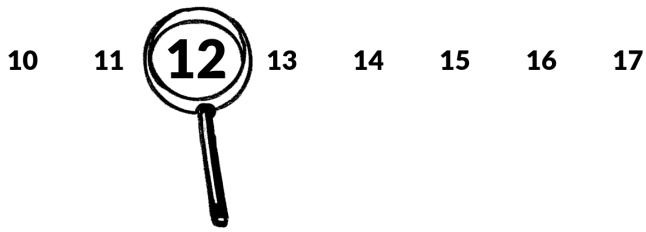
Em Portugol, o ciclo é realizado pelo comando enquanto: quando o computador chega neste comando, ele entende que deve repetir os comandos dentro das chaves seguintes enquanto a condição colocada ($m < 10$) for verdadeira. A cada repetição, a variável m tem o seu conteúdo aumentado em uma unidade, o que faz com que em algum momento o seu valor seja maior ou igual a 10, violando a condição e fazendo com que a interpretação do algoritmo saia do ciclo.

Esse é um exemplo simples de uma estrutura de repetição, que poderia ser feita por uma pessoa sem grande esforço. Mas ela transmite muito bem a ideia de repetir alguma ação enquanto uma condição for verdadeira. Nas atividades a seguir, vamos propor algumas tarefas repetitivas que são muito trabalhosas quando feitas manualmente, por isso é ainda mais relevante utilizar um computador para fazê-las.

É primo ou não?

Atividade 7

Você deve se lembrar que um número é chamado de primo se tiver exatamente dois divisores inteiros positivos diferentes: 1 e ele mesmo. Por exemplo, o número 9 não é primo, pois seus divisores são 1, 3 e 9. Já o número 13 é primo. O número 1 não é primo por que possui um único divisor inteiro positivo.



Até hoje, não existem métodos realmente rápidos para identificar se um número é primo ou não. Basicamente, todos os métodos existentes baseiam-se em testar se o número dado deixa resto zero quando é dividido pelos números menores do que ele. Imagina a quantidade de repetições necessárias para verificar se o número $4.000.037$ é primo!

a) Descreva um algoritmo que verifica se um número é primo ou não.

b) Em conjunto com toda a turma, escrevaa um dos algoritmos criados na questão 1 em Portugol.

c) Utilize esse algoritmo para verificar se o número $4.000.037$ é primo.

Você deve ter notado que o algoritmo responde quase que imediatamente se o número dado é primo ou não, mesmo sendo um número grande.

d) Se você conseguisse verificar um divisor a cada 1 segundo, estime aproximadamente quanto tempo você teria levado para verificar se $4.000.037$ é primo.

PARA REFLETIR

Embora muitas checagens precisem ser feitas, não é necessário dividir um número dado por todos os números naturais menores do que ele para verificar se ele é primo. Várias melhorias podem ser feitas nesse processo para torná-lo mais eficiente. Discuta com a turma essas melhorias.

A sequência de Collatz

Atividade 8

A sequência de Collatz é uma sequência numérica formada por números inteiros que é construída a partir de um valor inicial, que podemos chamar de a_1, por meio da seguinte regra: o próximo termo da sequência é igual à metade do anterior, se o anterior for par, e igual ao tri-

plo do anterior mais um, se o anterior for ímpar. Matematicamente, podemos descrever essa regra de formação dessa maneira:

$$a_n = \begin{cases} a_{n-1}/2, & \text{se } a_{n-1} \text{ é par} \\ 3a_{n-1} + 1, & \text{se } a_{n-1} \text{ é ímpar} \end{cases}$$

Por exemplo, se tomarmos , temos a seguinte sequência: $5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, \ldots$ Note que a sequência entra em um cilo depois que chega em 1 pela primeira vez. Por isso, dizemos que ela termina ao chegar em 1, ou seja, a sequência de Collatz com $a_1 = 5$ tem 6 termos: $5, 16, 8, 4, 2, 1$.

a) Escreva a sequência de Collatz para $a_1 = 12$.

b) Escreva a sequência de Collatz para $a_1 = 18$.

c) Escolha um número entre 5 e 20 e obtenha a sequência de Collatz iniciada por esse número.

PARA REFLETIR

Você consegue notar algumas semelhanças entre as sequências de número obtidas em cada um dos casos anteriore?

d) Descreva um algoritmo que obtenha a sequência de Collatz a partir de um valor dado para a_1.

O comportamento da sequência de Collatz pode ser surpreendente. Por exemplo, se usarmos $a_1 = 26$ a sequência tem 11 termos, mas se usarmos $a_1 = 27$ a sequência precisa de 113 termos para terminar! É claro que repetir esse processo 113 vezes pode ser bem entediante, mas esse é o tipo de tarefa que um computador pode fazer sem dificuldade e muito mais rapidamente.

Seu algoritmo deve escrever os termos que compõem a sequência. Também pode ser útil incluir uma variável que conta o número de termos da sequência.

Esse algoritmo pode ser usado para obter a sequência de Collatz para muitos valores diferentes de a_1, incluindo números bastante grandes. Por exemplo, a sequência obtida a partir de $a_1 = 987654$ tem 182 termos!

Com esse algoritmo em mãos, é mais simples testar casos e eventualmente levantar conjecturas sobre essa sequência: é possível prever valores de que geram sequências bem curtas? Há alguma família de números que, se atingida, encaminha a sequência para o seu final? Será que qualquer que seja o valor de a sequência sempre termina?

Um detalhe interessante é que até hoje os matemáticos não conseguiram responder satisfatoriamente a essa última pergunta! Com o auxílio de computadores, muitos valores de a_1 já

foram testados e a sequência gerada a partir deles sempre chega, cedo ou tarde, em 1. Mas ninguém conseguiu provar que isso é verdadeiro para qualquer número inteiro positivo desde que o problema foi proposto (há mais de 80 anos!).

ORGANIZANDO REPETIÇÕES

Processos que envolvem muitas repetições são um exemplo muito claro de como os computadores podem ser usados como ferramenta: podemos programá-los para executar essa parte da tarefa e focarmos nas partes que envolvem criatividade e o reconhecimento de padrões. No caso da sequência de Collatz, poder analisar e comparar várias sequências rapidamente nos permite levantar conjecturas, testá-las e ajustá-las de acordo com as observações.

Para compreendermos bem o uso de repetição, vamos analisar o algoritmo abaixo. Ele escreve os termos de uma **sequência** que começa com os termos 1 e 1 (veja que as variáveis `a1` e `a2` começam recebendo esses valores). A variável `n`, que é lida no início do algoritmo determina a quantidade de termos que serão escritos. Veja que os dois primeiros são escritos antes do comando enquanto, por isso a variável `i` receber o valor 3 (quando o enquanto começar, já estaremos escrevendo o terceiro termo).

Quais valores serão escritos por esse algoritmo se usarmos `n=6`? E `n=10`?

```
programa
{
    funcao inicio()
    {
        inteiro a1,a2,an,n,i
        a1=1
        a2=1
        escreva(a1+" "+ a2+" ")
        leia(n)
        i=3
        enquanto (i<=n) {
            an=a1+a2
            escreva(an+" ")
            i=i+1
            a1=a2
            a2=an
        }
    }
}
```

PARA REFLETIR

Você conhece essa sequência numérica?

O aspecto que é importante compreender neste algoritmo é o uso das variáveis `i` e `an`.

Note que a variável `i` começa com o valor 3 e ela aparece na condição que determina quantas vezes o comando enquanto vai ser repetido. Ao final dos comandos que são executados dentro do enquanto, o valor de `i` é incrementado em 1 unidade, até que ela fica maior do que n e o enquanto para de ser repetido. O papel dessa variável é servir como um contador para o enquanto.

Já a variável an tem uma função diferente. Para entendê-la, é necessário olhar para os comandos das linhas 12, 15 e 16 em conjunto. Na linha 12, a variável an recebe o valor de a1+a2 e depois o seu conteúdo é escrito, ou seja, `an` é o próximo termo da sequência. Porém, o que ocorre nas linhas 15 e 16?

O que está sendo feito ali é "avançando na sequência". Nesse ponto, as variáveis `a1` e `a2` deixam de ser o primeiro e segundo termos e passam a ser os próximos: `a1` recebe o valor do `a2` e `a2` recebe o valor de `an`, como se estivéssemos "andando" na sequência. Dessa forma, na próxima repetição, `a1` e `a2` serão os dois últimos valores calculados e an será a soma deles, ou seja, o próximo termo.

Você pode adaptar esse algoritmo, e o que aprendeu com as atividades anteriores, para obter outras sequências numéricas, como progressões aritméticas, progressões geométricas e os números triangulares. Pesquise na internet sobre essas sequências e tente construir algoritmos em Portugol para cada uma delas.

PRATICANDO

Nesta seção, vamos praticar o que foi estudado até este momento. Todas as atividades propostas envolvem dois aspectos, o matemático e o computacional. Isso significa que conhecimentos matemáticos serão necessários para compreender e resolver os problemas e você também terá que pensar computacionalmente sobre o processo de resolução, ou seja, pensar sobre como transformar a resolução em um algoritmo.

Imposto de renda retido na fonte

Atividade 9

O imposto de renda é um dos principais impostos no Brasil. Uma das formas em que esse imposto é cobrado ocorre na folha de pagamento, ou seja, quando um funcionário com carteira de trabalho assinada recebe o seu salário. Essa forma de cobrança é chamada de imposto de renda retido na fonte. O cálculo do valor a ser descontado do salário (e imediatamente repassado ao governo federal) é feito de forma que ele seja percentualmente maior à medida que o salário aumenta. No Brasil, são adotados 5 intervalos de salário bruto (isto é, ainda sem o desconto do imposto) que seguem regras de cálculo diferentes. Para cada um desses intervalos, o cálculo é feito da seguinte maneira: calcula-se a porcentagem indicada em "alíquota" do salário bruto e, do resultado, subtrai-se a "parcela a deduzir". O resultado obtido é o valor que será descontado do salário no momento do pagamento.

Base de cálculo (R$)	Alíquota (%)	Parcela a deduzir do IR (R$)
Até $1.903{,}98$	-	-
De $1.903{,}99$ até $2.826{,}65$	7,5	$142{,}80$
De $2.825{,}55$ até $3.751{,}05$	15	$354{,}80$
$3.751{,}06$ até $4.664{,}68$	22,5	$636{,}13$
Acima de $4.664{,}68$	27,5	$869{,}36$

Tabela 1.1: Fonte: Receita Federal

Por exemplo, para um salário de R$ $2.000{,}00$, começamos identificando que ele se encontra na segunda faixa. Logo, devemos calcular $7{,}5\%$ de 2000 e depois subtrair $142{,}80$ do resultado. Portanto, $0{,}075 \times 2000 - 142{,}80 = 7{,}20$ é o valor que será descontado do salário para pagar o imposto de renda desse trabalhador.

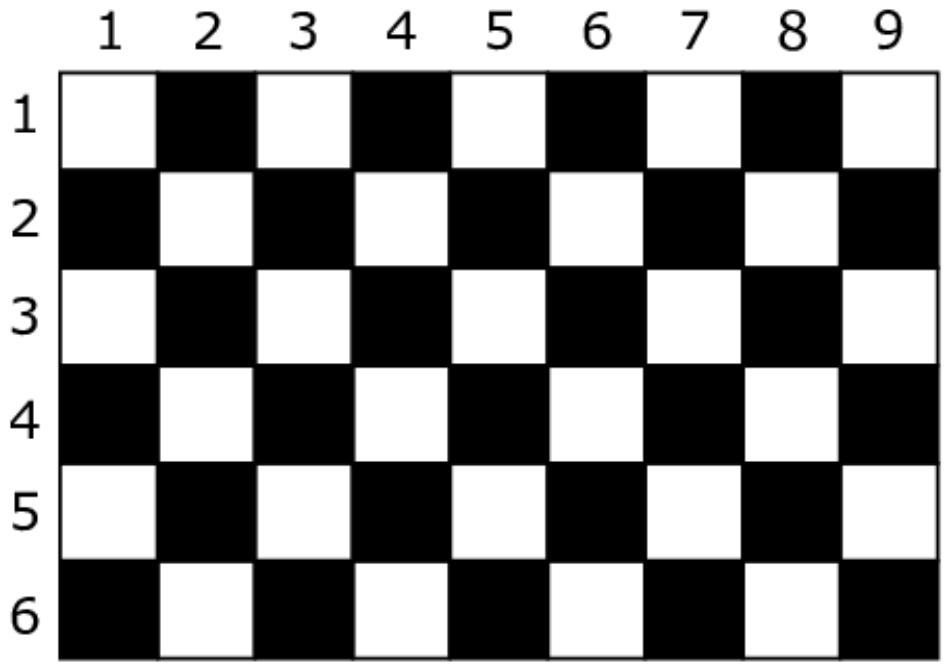
a) Quanto será descontado de uma pessoa que tenha salário bruto igual a R$ $1.500{,}00$? E igual a R$ $3.000{,}00$? E igual a R$ $6.000{,}00$?

b) Crie um algoritmo que calcula o valor do imposto de renda a ser retido na fonte para um dado salário bruto.

Xadrez

Atividade 10

Fonte: Olimpíada Brasileira de Informática No tabuleiro de xadrez, a casa na linha 1, coluna 1 (canto superior esquerdo) é sempre branca e as cores das casas se alternam entre branca e preta. Dessa forma, como o tabuleiro tradicional tem oito linhas e oito colunas, a casa na linha 8, coluna 8 (canto inferior direito) será também branca. Neste problema, entretanto, queremos saber a cor da casa no canto inferior direito de um tabuleiro com dimensões quaisquer: L linhas e C colunas. No exemplo da figura, para L=6 e C=9, a casa no canto inferior direito será preta.



a) Qual seria a cor da casa no canto inferior direito se o tabuleiro tiver 7 colunas e 4 linhas? E se tiver 10 colunas e 8 linhas? E se tiver 23 colunas e 40 linhas?

b) Descreva como descobrir a cor da casa no canto inferior direito de um tabuleiro como esse, sabendo a quantidade de linhas e colunas que o compõe.

c) Escreva um algoritmo que implemente o processo que você descreveu na questão anterior.

MMC

Atividade 11

Existem vários métodos para o cálculo do mínimo múltiplo comum (mmc) entre dois números inteiros. Você deve ter aprendido como fazer isso ainda no Ensino Fundamental e, de vez em quando, ainda deve utilizar esse procedimento para resolver algumas questões de matemá-

tica.

a) Descreva como você procede para obter o mínimo múltiplo comum entre dois números dados. Se quiser, comece obtendo o mínimo múltiplo comum para os números 12 e 18 e depois tente descrever o método que você utilizou de forma genérica, isto é, para dois números quaisquer a e b.

   Embora não seja o método mais eficiente, vamos implementar o método da lista. Ele consiste em listar os múltiplos do primeiro número (começando por ele mesmo) até encontrar um múltiplo que seja divisível pelo segundo número.

   Exemplo: para o caso 12 e 18, vamos listar os múltiplos de 12. Primeiro, o próprio 12, que não é divisível por 18. Depois 24 $(12 * 2)$, que não é divisível por 18. Depois 36 (12*3), que é divisível por 18 e, portanto, é o mínimo múltiplo comum entre esse 12 e 18.

b) Escreva um algoritmo que obtenha o mínimo múltiplo comum entre dois números naturais dados, usando o método da lista.

PARA REFLETIR

Sabendo como calcular o mínimo múltiplo comum entre dois números dados, a e b, você pode utilizar uma propriedade simples para calcular o máximo divisor comum entre esses números. A propriedade diz que o máximo divisor comum entre a e b é igual ao produto dos dois números dividido pelo mínimo múltiplo comum entre eles:

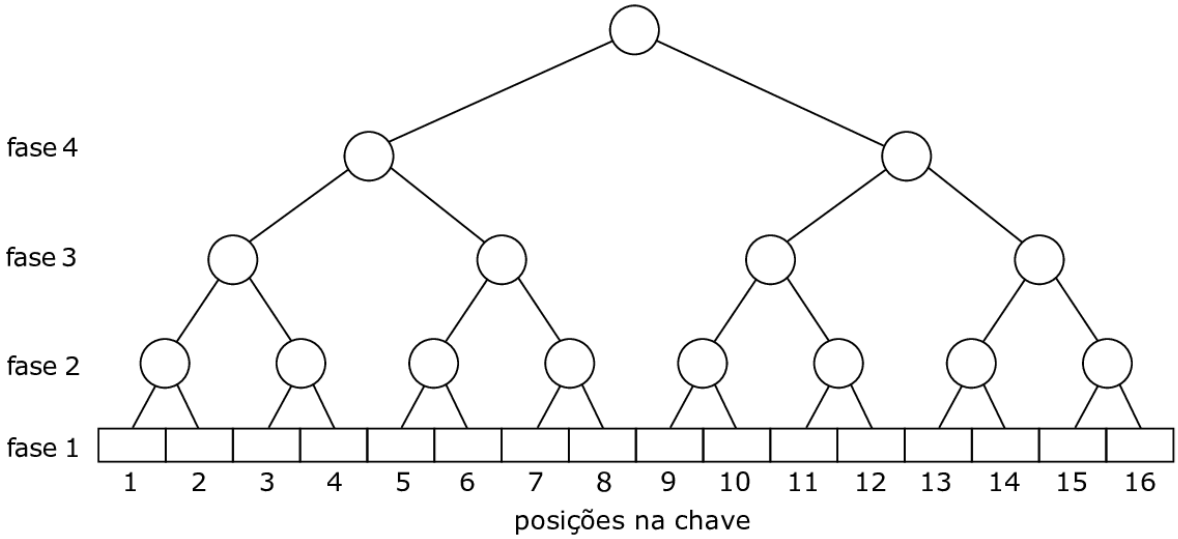
$$\text{mdc}(a, b) = \frac{a \cdot b}{\text{mmc}(a, b)}$$

Como você poderia usar essa propriedade para escrever um algoritmo que obtenha o máximo divisor comum entre dois números naturais dados.

Campeonato

Atividade 12

Um torneio esportivo mundial é organizado no tradicional formato de chaves: dois competidores se enfrentam e quem vence passa para a próxima etapa, até que os dois que venceram todas as partidas se enfrentem na final. Nesta edição, o torneio vai contar com 16 competidores e os confrontos ocorrerão como mostrado abaixo.



O grande adversário do Brasil neste torneio é os Estados Unidos e, por isso, todos querem saber quando os competidores desses dois países poderão vir a se enfrentar. Isso será determinado um dia antes do início do torneio, quando será sorteada uma bolinha com um número de 1 a 16 para cada país, indicado a posição em que eles serão alocados nas chaves.

Escreva um algoritmo que, dadas as duas posições do Brasil e Estados Unidos nas chaves, determina em qual fase do torneio os países poderão se enfrentar.

PARA REFLETIR

Você conseguiria modificar o seu algoritmo de modo que ele possa ser usado para outras quantidades de competidores?

Você pode assumir que essas quantidades sempre serão uma potência de 2 $(2, 4, 8, 16, 32, 64,$ etc).

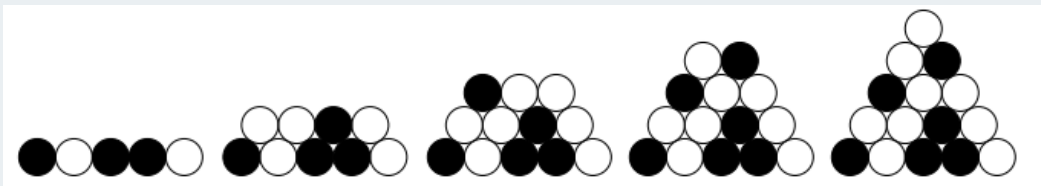
PROJETO APLICADO

Para o encerramento desta parte, escolhemos um problema que foi sugerido na Olimpíada Brasileira de Informática. Este problema, além do desafio matemático envolvido na sua resolução, vai exigir que você aprenda um novo conceito muito importante em todas as linguagens de programação. Mas vamos começar com o problema.

Triângulo de bolinhas: Dois amigos inventaram um passatempo com bolas pretas e brancas. Elas são colocadas uma por vez na mesa, de acordo com uma regra fixa: no início, são colocadas N bolas formando a primeira fileira; em seguida, um triângulo é formado, fileira a fileira, de modo que ao se colocar uma bola na nova fileira, ela ficará encostada em duas bolas da fileira anterior e sua cor será:

a) Preta, se estiver encostada em duas bolas de mesma cor; Branca, se estiver encostada em duas bolas de cores diferentes.

b) O passatempo acaba quando a bolinha do topo é colocada. A figura abaixo ilustra a formação de um triângulo para N=5.



A partir deste contexto, é possível colocar várias perguntas que podem ser respondidas com auxílio da matemática e da computação. Por exemplo:

a) Dadas as cores das bolinhas da primeira linha, qual será a cor da bolinha do topo?

b) É possível prever a cor da última bolinha sem precisar construir todas as linhas?

A criação de algoritmos que permitam a investigação dessas perguntas depende do uso de um recurso existente em todas as linguagens de programação, mas que não discutiremos neste livro. Este recurso é chamado vetor em português, ou array em inglês. Ele tem algumas semelhanças com o conceito de vetor que são estudadas nas aulas de matemática e física, mas seu uso em computação tem algumas peculiaridades que precisam ser compreendidas para seu uso efetivo. De maneira simplificada, um vetor é uma variável que contém várias variáveis de um mesmo tipo dentro e essas variáveis podem ser acessadas de acordo com a sua posição no vetor. Você pode aprender mais sobre vetores em Portugol com o vídeo youtu.be/5oQrDq8qqfg.

Nossa sugestão é que este problema seja encarado como um projeto, ou seja, que não será resolvido em uma ou duas aulas, mas exigirá de você (e de seus colegas) muito estudo, análises e discussões. Bom trabalho!

2 Pensamento Computacional - Aplicações

O QUÊ?

Aplicações do pensamento computacional e de linguagens de programação na exploração de problemas matemáticos.

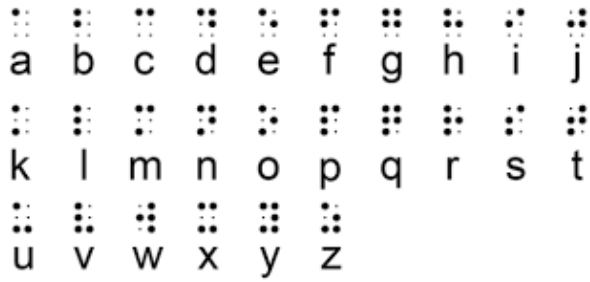
POR QUÊ?

Dominar uma linguagem de programação nos permite explorar e resolver problemas matemáticos de maneiras diferentes. Assim, podemos entender o poder de ferramentas computacionais e porque elas são tão usadas em todas as áreas do conhecimento atualmente.

EXPLORANDO NÚMEROS BINÁRIOS

Dois estados

Na atividade sobre alfabeto Braile no módulo sobre Contagem, você conheceu um pouco sobre esta forma de escrita desenvolvida para cegos. Basicamente, você viu que as letras em Braile são representadas por um conjunto de 6 pontos dispostos em 2 colunas e 3 linhas, de modo que o estado de cada ponto pode ser marcado (pontos maiores na figura abaixo, em relevo quando impressos) ou não marcado (pontos menores na imagem abaixo, sem relevo quando impressos).



A questão resolvida no módulo sobre Contagem foi: quantos caracteres diferentes podemos montar com a estrutura do alfabeto Braile?

Para resolver a esta questão, usamos basicamente a ideia de que cada pontinho é independente e pode assumir dois estados: com relevo ou sem relevo. Dessa forma, para cada um dos 6 pontinhos temos duas possibilidades, o que leva a: $2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 2^6 = 64$ caracteres (63 se excluírmos o caso em que nenhum pontinho foi marcado).

O que vamos explorar nesta seção é uma relação do alfabeto Braile com a maneira como computadores guardam informação.

A relação vem de outra característica: assim como o alfabeto Braile, computadores utilizam apenas dois estados para armazenar informações, os famosos 0 e 1 (ligado e desligado, carregado e descarregado, verdadeiro e falso, etc).

O vídeo Hit dos Bits, da coleção Matemática Multimídia explica como o sistema de representação binário é usado por computadores.

Nesta seção, vamos analisar o sistema de representação binária com o objetivo de escrever um algoritmo que converta um número dado na base decimal para a base binária. Mas para isso, é importante compreendermos bem como essa conversão funciona.

Dois estados

Atividade 1

No sistema de numeração decimal, ou de base 10, cada algarismo de um número, como o 308, representa um múltiplo de uma potência de 10:

$$308 = 300 + 0 + 8$$
$$308 = 3 \cdot 100 + 0 \cdot 10 + 8 \cdot 1$$
$$308 = 3 \cdot 10^2 + 0 \cdot 10^1 + 8 \cdot 10^0$$

No sistema de numeração binário, ou de base 2, temos apenas dois algarismos disponíveis, o 0 e o 1, e a posição de cada algarismo determina qual potência de 2 ele está multiplicando. Por exemplo, o número 1101, na representação binária, pode ser convertido para a base decimal da seguinte forma:

a) Usando essas ideias, obtenha a representação decimal dos seguintes números dados em representação binária: 101, 10000 e 11.

O processo descrito acima é relativamente simples, pois ao escrevermos as potências de 2 já estamos trabalhando na representação decimal, na qual sabemos fazer somas e multiplicações sem dificuldades. Já o processo inverso, de transformar um número dado na representação decimal para a representação binária, é um pouco mais difícil, pois precisamos encontrar quais potências de 2 compõem o número em questão.

Por exemplo, o 20 pode ser escrito com $16 + 4$. Note que 16 e 4 são ambos potências de 2, $16 = 2^4$ e $4 = 2^2$. Logo:

$$20 = 16 + 4$$
$$20 = 2^4 + 2^2$$
$$20 = \cdot 2^4 + 1 \cdot 2^2$$
$$20 = \cdot 2^4 + 1 \cdot 2^2$$
$$20 = \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$$

Assim, o número 20 é representado na base binária por 10100.

b) Escreva os números 10, 25 e 50 na base binária, como mostrado no exemplo anterior e seguindo a estrutura da tabela abaixo.

Representação decimal	Soma de potências de 2	Soma completa de potências de 2	Representação binária
20	$16 + 4$	$1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$	10100
10			
25			
50			

Agora, vamos pensar sobre esse procedimento de forma a sistematizá-lo em um algoritmo.

c) Tente descrever textualmente, ou com auxílio de um fluxograma, como você pode proceder para converter um número natural dado na representação decimal para a representação binária. Faça essa descrição em uma folha de papel avulsa com a frente e o verso em branco.

d) Troque com um colega a sua descrição e avalie se você consegue seguir as instruções criadas por ele e se elas funcionam corretamente. Faça anotações para o seu colega, destroque e reescreva a sua descrição no verso da folha levando em conta as anotações feitas na sua descrição pelo seu colega.

0s e 1s

Existe mais de uma maneira de converter um número inteiro da base decimal para a base binária e o método que vamos utilizar se baseia em encontrar quais potências de 2 são menores do que o número que se deseja representar.

Para entender essa abordagem, precisamos levar em conta uma propriedade das potências de 2. Essa propriedade diz que a soma das potências de 2 até uma determinada potência é menor do que a próxima potência de 2. Isto é, $2^0 + 2^1 + 2^2 + 2^3 + \cdots + 2^n < 2^{n+1}$.

Não demonstraremos essa propriedade aqui, mas isso pode ser feito com auxílio de progressões geométricas.

O que essa propriedade nos permite concluir é que se uma potência de 2 é menor do que um determinado número, ela necessariamente deve aparecer na representação binária desse número. Vejamos um exemplocom o número 46.

Começamos listando as potências de 2 até que elas sejam maiores do que 46: 1, 2, 4, 8, 16, 32 e 64. Como 64 é maior do que 46, ele não vai entrar na representação binária desse número. Mas como 32 é menor, ele vai entrar. Podemos escrever o seguinte:

$$46 = 32 + 14 = 2^5 + 14$$

Agora, temos que continuar o processo com a parte que ainda não é uma potência de 2, ou seja, o 14. Como a maior potência de 2 menor do que 14 é 8, escrevemos o 14 como uma soma de 8 mais a diferença, no caso, 6:

$$46 = 2^5 + 14$$
$$46 = 2^5 + (8 + 6)$$
$$46 = 2^5 + 2^3 + 6$$

O processo continua até termos apenas potências de 2 na soma.

$$46 = 2^5 + 14$$
$$46 = 2^5 + (8 + 6)$$
$$46 = 2^5 + 2^3 + 6$$
$$46 = 2^5 + 3^3 + (4 + 2)$$
$$46 = 2^5 + 2^3 + 2^2 + 2^1$$

Finalmente, indicando o 1 como multiplicador de cada uma das potências acima e completamos com as potências de 2 que não apareceram multiplicando-as por 0 (para não alterar o valor da soma:

$$46 = 2^5 + 2^3 + 2^2 + 2^1$$
$$46 = 1 \cdot 2^5 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1$$
$$46 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$$

Portanto, a representação binária de 46 é dada por 101110.

0s e 1s

Atividade 2

a) Use o procedimento descrito acima para obter a representação binária do número 145.

b) Escreva em Portugol um algoritmo que converta um número inteiro, dado na base decimal, para a base binária. Vamos assumir, por enquanto, que o número dado é menor do que 256, ou seja, você precisa considerar apena as potências de 2 até 2^7.

c) Use o algoritmo obtido na questão anterior para obter a representação decimal do número 200.

ORGANIZANDO 0S E 1S

A base binária é importante não apenas pelo seu uso em computação, mas também pela sua simplicidade: o fato de usarmos apenas 2 símbolos (0s e 1s) para representar um número qualquer confere a ela algumas propriedades que podem ser usadas em diversos outros contextos.

Em contagem, é possível fazer analogias muito interessantes entre a base binária e o processo de escolher elementos de um conjunto de elementos disponíveis, pois podemos pensar que cada elemento admite dois estados: ser ou não ser escolhido.

Considere o seguinte cenário: queremos escolher 2 letras dentre as letras A, B, C, D e E. Podemos traduzir as escolhas para o seguinte formato:

- Escolher A e B pode ser representado como $1, 1, 0, 0, 0$
- Escolher C e E pode ser representado como $0, 0, 1, 0, 1$
- Escolher A e D pode ser representado como $1, 0, 0, 1, 0$

Nessa representação, 1 significa "escolhido" e "0" significa "não escolhido" e todos os elementos disponíveis devem ser marcados em um dos dois estados.

Note que agora cada escolha pode ser vista como um número de 5 algarismos na base binária. Portanto, para saber de quantas formas podemos fazer essa escolha, basta contar quantos números binários de 5 algarismos são representados exatamente com dois algarismos 1.

Esse é apenas um dos exemplos fora da computação e da escrita em Braile em que podemos aplicar a base binária.

PARA SABER + OUTRAS BASES

Mas não só das bases decimal e binária vive o mundo atual!

Em contextos digitais, cores são comumente representadas na base 16 (hexadecimal). Nessa base, além dos algarismos $0, 1, 2, 3, 4, 5, 6, 7, 8$ e 9, são usadas as letras A, B, C, D, E, e F, totalizando 16 opções.

Por exemplo, a cor conhecida como "verde militar" é dada pela seguinte trinca de números em hexadecimal: $78866b$. Essa trinca representa a quantidade de vermelho (78), verde (86) e azul $(6b)$ que devem ser usadas para compor a cor desejada. Na verdade, o código que representa cores é usualmente apresentado com um "#" na frente e sem nenhum espaço entre as três quantidades, ou seja, verde militar será dado por "$\#78866b$". Neste vídeo, você pode aprender um pouco mais sobre a base hexadecimal.

A base 3 é pouco utilizada em contexto digitais, mas aparece de forma natural em um problema concreto: como encontrar um saquinho mais leve em um conjunto de saquinhos que deveriam ter exatamente o mesmo peso com o auxílio de uma balança de braços? O problema, a resolução e a sua relação com a base 3 são apresentados no texto Balança e a base 3.

EXPLORANDO O MÉTODO DA BISSECÇÃO

Usando o computador para calcular $\log_2 5$

No módulo sobre Logaritmos, você viu o método da bissecção sendo aplicado para calcular uma aproximação para $\log_2 3$. O método foi descrito da seguinte maneira:

> **Método da bissecção**
>
> Para aplicarmos o método da bissecção, tomamos uma aproximação superior e outra inferior para o número buscado, encontrando assim um intervalo no qual temos certeza que a solução se encontra. Então tomamos o ponto médio desse intervalo como a próxima aproximação e buscamos decidir se ele é superior ou inferior ao número buscado, daí basta substituirmos o respectivo extremo do intervalo pelo ponto médio. Repetindo esse processo obtemos aproximações cada vez melhores.

Logo em seguida, uma aplicação do método no cálculo de uma proximação para $\log_2 3$ é apresentada em detalhes. O que faremos nesta seção é implementar um algoritmo que execute este método pensando, inicialmente, em $\log_2 5$.

Porém, para isso, você precisa aprender antes a usar uma nova funcionalidade do Portugol chamada **biblioteca**. Uma biblioteca é um conjunto de comandos que não estão disponíveis entre as funcionalidades básicas de uma linguagem de programação. Esse é o caso do comando que calcula o valor de um número real elevado a outro número real, ou seja, . Para realizar esse cálculo, precisamos de uma função que faz parte da biblioteca chamada Matematica no Portugol. Veja no código abaixo como essa biblioteca é carregada e usada.

```
1  programa
2  {
3      inclua biblioteca Matematica --> mat
4      funcao inicio()
5      {
6          real a, b, resultado
7          leia(a)
8          leia(b)
9          resultado = mat.potencia(a, b)
10         escreva(resultado)
11     }
12 }
```

O comando usado na linha 3 carrega os comandos que fazem parte dessa biblioteca na variável `mat`. Na linha 9, o comando `potencia` é usado, mas como ela faz parte da biblioteca, ele precisa ser acionado não apenas pelo seu nome, mas acrescentando `mat.` logo antes. Isso diz ao computador para usar o comando potencia que está carregado na variável `mat`.

PARA REFLETIR

A possibilidade de usarmos esse comando poupa as várias comparações feitas no módulo sobre logaritmos com as potências de expoente fracionário. Isso é conveniente no sentido de economizar cálculos, mas é importante estarmos atentos quanto às saídas apresentadas, a fim de verificarmos que elas correspondem à programação prentendida.

Implementando o método da bissecção

Atividade 3

a) Leia a descrição do método da bissecção dada acima e, se necessário, o exemplo dado no módulo sobre logaritmos e descreva textualmente, por meio de um fluxograma, o método da bissecção para obter uma aproximação para o valor de $\log_2 5$.

b) Escreva um algoritmo em Portugol implementando a resposta da questão anterior. Seu algoritmo deve pedir ao usuário quantas repetições ele deseja fazer e quais são os valores inferior e superior iniciais. Ao final, seu algoritmo deve mostrar o último valor médio calculado.

c) Se usarmos 5 repetições e 1 e 10 como valores inferior e superior iniciais, qual é o valor que o seu algoritmo obtem para $\log_2 5$? E se usarmos 10 repetições?

d) Use o seu algoritmo para gerar aproximações para $\log_2 7$ e $\log_{10} 2$. Qual valor você obteve?

PARA REFLETIR

Quais alterações devem ser feitas no seu algoritmo para que ele obtenha aproximações para $\sqrt{2}$?

ORGANIZANDO APROXIMAÇÃO, ERRO E ESTIMATIVA DO ERRO

O uso que fizemos do método da bissecção permite-nos encontrar a representação decimal de $\log_2 5$, com o número de casas de precisão que se desejar, supondo que sabemos como calcular potências de 2. O Portugol também apresenta na biblioteca matematica a função logaritmo, que calcula o valor de $\log_2 5$ diretamente, mas o nosso objetivo ao apresentar o método da bissecção é obter a representação decimal de um número com a precisão que se quiser.

É importante estarmos cientes de que métodos como os da bissecção obtêm aproximações para os valores buscados. Essas aproximações tornam-se cada vez mais precisas à medida que aumentamos o número de repetições. Para ilustrar esse fenômeno, vamos usar o algoritmo abaixo. Leia-o com atenção antes de utilizá-lo para ter certeza de que você entende como ele funciona.

```
programa
{
    inclua biblioteca Matematica --> mat
    funcao inicio()
    {
        real inf, sup, meio
        inteiro n, i
        leia(inf,sup,n)
        i=1
        meio = (sup+inf)/2
        enquanto (i<=n) {
            se ( mat.potencia(2.0, meio) > 5 ) {
                sup=meio
            }
            senao {
                inf=meio
            }
            meio = (sup+inf)/2
            i=i+1
        }
        escreva("valor obtido: ", meio)
        escreva("estimativa do erro: ", sup-inf)
    }
}
```

Note que no final, o algoritmo mostra não apenas o valor obtido, como também a diferença entre os valores inferior e superior do intervalo. Essa diferença representa uma estimativa para o erro da aproximação. Dizemos estimativa porque não sabemos qual é o número que estamos procurando para calcularmos o erro de fato, sabemos apenas que ele certamente se encontra dentro de um determinado intervalo.

Usando esse algoritmo com os valores inferior e superior sempre iguais a 2 e 3, mas variando o número de repetições, obtemos os seguintes resultados:

Número de repetições	Valor obtido	Estimativa do erro
5	2,328125	0,03125
10	2,32177734375	0,0009765625
20	2,3219285011291504	0,0000009536743

Note que com 5 repetições, a diferença entre os valores mínimo e máximo que delimitam o intervalo onde sabemos que está o valor analisado é igual a $0{,}03125$. Isso significa que a primeira casa decimal desses dois valores são iguais e, portanto, temos certeza de que a primeira casa depois da vírgula da representação decimal do número $\log_2 5$ é 3 e que, portanto, poderíamos usar a aproximação $2{,}3$ tendo certeza de que esses algarismos estão corretos.

Quando consideramos os resultados depois de 10 repetições, já podemos fazer essa afirmação para as 3 primeiras casas depois da vírgula, ou seja, já temos certeza de que a representação decimal do número $\log_2 5$ começa com $2{,}321$. De fato, note que a quarta casa decimal obtida com 10 repetições é diferente do valor obtido com 20 repetições.

Com 20 repetições, já temos certeza sobre as 6 primeiras casas depois da vírgula!

Se o nosso foco é determinar uma certa quantidade de casas decimais do número em questão, poderíamos alterar o algoritmo de modo que ele realizasse repetições até que tenhamos certeza sobre essa quantidade de casas decimais. Para isso, bastaria usar a diferença entre os valores mínimo e máximo do intervalo como critério da repetição.

Com 50 repetições, algo que é feito quase que instantaneamente com esse algoritmo em um computador convencional, estaríamos chegando no limite que o próprio Portugol consegue registrar, afinal, o computador também tem limitações na representação de números decimais. Porém, com linguagens de programação especificamente criadas para processamento numérico de alta precisão, esse processo poderia ser repetido um número muito maior de vezes.

EXPLORANDO QUANTAS CARAS?

Usando o computador para jogar moedas

Conta-se que um matemático foi feito prisioneiro durante a segunda guerra mundial e, para manter a mente ocupada no cárcere, lançou uma moeda 10 mil vezes obtendo uma das faces 5067 vezes e a outra 4933.[1]

A fascinação dos matemáticos que estudam probabilidade por moedas deve-se, provavelmente, ao fato de que este objeto simples (que admite apenas 2 resultados, ambos com a mesma probabilidade de ocorrência) permite a simulação de situações mais complexas e generalizantes.

Nesta seção, vamos ver como utilizar a linguagem Portugol para simular o lançamento de moedas. Nosso objetivo é criar um simulador para a seguinte questão:

Quantas caras? Duas moedas são lançadas e conta-se quantas caras foram obtidas a cada lançamento. Sendo assim, os resultados possíveis são: 0, 1 ou 2. Antes de as moedas serem lançadas, você escolhe em qual possibilidade quer apostar. Se acertar, vence. Se errar, perde.

PARA REFLETIR

Em um primeiro olhar, algum dos três resultados parece ter maior probabilidade de ocorrer? Porquê?

Vamos começar simulando o problema sem computador e depois resolveremos algumas atividades auxiliares que irão nos ajudar a finalmente construir o simulador para o jogo "Quantas caras?".

Jogando moedas e contando caras

Atividade 4

a) Usando duas moedas, faça 20 lançamentos e anote os resultados em uma tabela como a mostrada abaixo.

Jogada	Faces obtidas	Quantas caras?
1	Cara-Cara	2
2		
3		
⋮		

b) Considerando as suas jogadas, qual foi o resultado mais frequente?

Embora você tenha feito 20 jogadas, essa quantidade ainda é muito pequena para tirarmos conclusões sobre qual das possibilidades tem maior probabilidade de ocorrer. Podemos reunir os dados da turma toda, o que nos renderia um conjunto de dados bem

[1] Aqui seria legal uma imagem (com função meramente ilustrativa) de uma pessoa fazendo aquela contagem de risquinhos na parede típica de prisioneiros, só que dessa vez em duas colunas indicando cara e coroa e algumas moedas jogadas no chão

maior, mas poderíamos obter ainda mais repetições desse experimento aleatório com o auxílio de um computador que simule as jogadas e anote os resultados para nós.

Então, vamos começar a pensar em como o processo de fazer várias jogadas poderia ser transformado em um algoritmo.

c) Descreva textualmente um algoritmo que realize n repetições do jogo "Quantas caras?" e registre quantas vezes saíram 2 caras, 1 cara ou 0. Não se preocupe com comandos específicos de Portugol ainda, apenas com a ideia geral do que deve ser realizado em cada etapa.

ORGANIZANDO USANDO O COMPUTADOR PARA JOGAR MOEDAS

Agora vamos analisar o algoritmo abaixo, escrito em Portugol. Este algoritmo não simula o jogo "Quantas caras?", mas sim um experimento mais simples: jogar uma moeda e anotar a face obtida. É importante compreendê-lo bem, pois todos os elementos que precisaremos para simular o jogo "Quantas caras?" (e tantos outros jogos aleatórios envolvendo moedas e dados) estão presentes nele.

```
1  programa
2  {
3      funcao inicio()
4      {
5          inteiro moeda, n, i, ncaras, ncoroas
6          leia(n)
7          i = 1
8          ncaras = 0
9          ncoroas = 0
10         enquanto (i<=n) {
11             moeda = sorteia(0,1)
12             se (moeda==1) {
13                 ncaras = ncaras+1
14             }
15             senao {
16                 ncoroas = ncoroas+1
17             }
18             i = i+1
19         }
20         escreva("Caras: ", ncaras, " Coroas: ", ncoroas)
21     }
22 }
```

Há um comando novo no algoritmo acima.

O comando "`sorteia(x,y)`" seleciona aleatoriamente um número inteiro no intervalo de x a y (incluindo os próprios x e y), de modo que cada valor tem a mesma chance de ser sorteado. No nosso caso, o comando da linha 11 seleciona o número 0 ou o número 1, com 50%de chance de cada um deles ser selecionado.

Um último aspecto que vale a pena salientar é a interpretação do valor numérico sorteado como cara ou coroa. Isso é feito pelos comandos se e senao. Nesse caso, o algoritmo está considerando que o valor 1 é cara e o valor 0 é coroa, mas isso pode ser definido da forma mais conveniente para os seus objetivos.

EXPLORANDO SIMULANDO O JOGO QUANTAS CARAS?

Agora que já vimos como sortear números naturais aleatoriamente usando o Portugol, vamos criar um algoritmo que faça a simulação de quantas jogadas quisermos para o jogo "Quantas caras?".

Tudo o que você precisa para criar esse algoritmo foi apresentado anteriormente. Dois detalhes precisam ser levados em conta:

1. Duas moedas precisam ser lançadas;
2. Existem três resultados possíveis para o experimento aleatório em questão: obter 2 caras, 1 cara ou 0.

Simulando o jogo "Quantas caras?"

Atividade 5

a) Escreva um algoritmo que permita repetir o jogo "Quantas caras?" quantas vezes o usuário desejar.

b) Use esse algoritmo para realizar 10 jogadas. Qual dos resultados foi o mais frequente? Qual foi a frequência relativa deste resultado?

c) Use esse algoritmo para realizar 100 jogadas. Qual dos resultados foi o mais frequente? Qual foi a frequência relativa deste resultado?

d) Use esse algoritmo para realizar 10 mil jogadas. Qual dos resultados foi o mais frequente? Qual foi a frequência relativa deste resultado?

e) Repita a simulação com 10 mil jogadas algumas vezes. Considerando os resultados observados nessas simulações, qual parece ser a probabilidade de cada um dos possíveis resultados? Você concorda com essa probabilidade? Justifique.

ORGANIZANDO SIMULANDO O JOGO QUANTAS CARAS?

Você deve ter notado que os resultados obtidos na questão 5 para as frequências relativas da atividade anterior foram sempre muito próximos.

Essa estabilidade nos resultados quando consideramos a razão entre o número de caras obtidos e o número total de lançamentos é chamada de "lei dos grandes números". Essa lei, que foi apresentada no módulo sobre Probabilidade, é uma das mais importantes da estatística e diz que quando realizamos um número muito grande de repetições, a frequência observada de cada resultado se aproxima da probabilidade teórica.

O uso de computadores para realizar muitas repetições de um experimento aleatório permite-nos observar essa lei em funcionamento e pode ajudar-nos a ter uma verificação empírica dos cálculos de probabilidade que já realizamos ou indicar tendências para probabilidades que ainda não tenhamos compreendido do ponto de vista teórico.

PARA SABER + OUTROS EXPERIMENTOS COM PORTUGOL

O comando sorteia pode ser utilizado para realizar outros tipos de experimentos aleatórios cujo resultados sejam discretos, como lançar um dado comum. Para isso, bastaria usar o comando sorteia da seguinte maneira: `sorteia(1,6)`.

Um problema que pode ser interessante investigar empiricamente com essa ferramenta é o Jogo do Máximo: a cada jogada, dois dados comuns são lançados. Se a maior face obtida for 5 ou 6, o jogador A vence. Se a maior face for 1, 2, 3 ou 4, o jogador B vence.

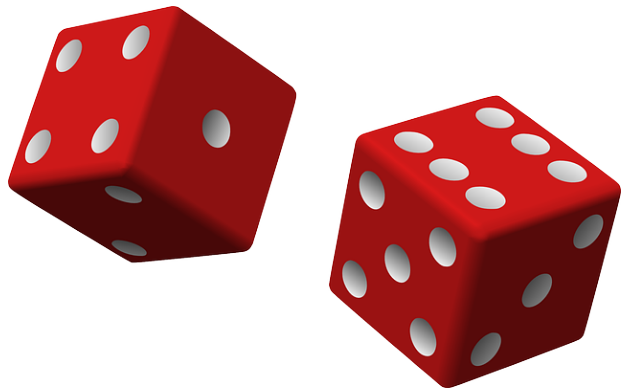


Figura 2.1: Imagem de Clker-Free-Vector-Images por Pixabay

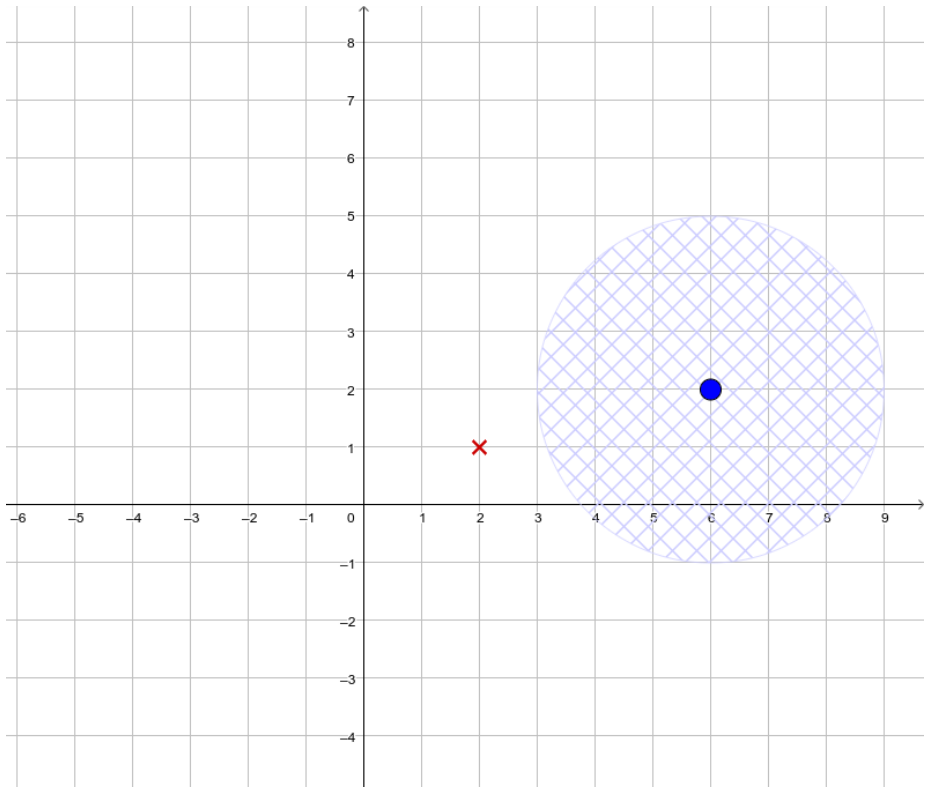
A pergunta a ser investigada é: qual dos dois jogadores tem maior probabilidade de vencer?

EXPLORANDO TEM SINAL?

Hoje em dia, a disponibilidade de sinal de celular é uma questão que vai além do acesso a redes sociais ou entretenimento. Com o aumento do uso da internet em diferentes áreas, ter acesso à internet pode ser determinante para que tenhamos acesso a serviços de saúde, educação, bancário, etc.

Nesta seção, vamos investigar, com o auxílio do computador, um problema relacionado à disponibilidade de sinal em uma determinada localidade.

O cenário que vamos adotar aqui é uma pequena simplificação do que acontece na realidade. Vamos considerar posições dadas em um plano cartesiano (com eixos dados em quilômetros) e que algum tipo de sinal é emitido radialmente por torres. A posição dessas torres também é dada por coordenadas no plano e cada torre possui um alcance diferente, como mostrado abaixo.



Por exemplo, na imagem acima, o aparelho celular está na posição marcada com x, que tem coordendas $(2; 1)$ e a torre está na posição $(6; 2)$ e tem um alcance de 3 quilômetros.

Quando temos uma representação visual em mãos, é fácil dizer se um ponto dado está dentro do alcance de uma torre dada. No caso acima, a resposta é não. Mas e quando temos apenas as coordenadas do celular e da torre?

Tem sinal? - parte 1

Atividade 6

a) Considere que uma nova torre com alcance de 3 quilômetros foi instalada no ponto de $(1; -1)$. Represente essa nova torre e o seu alcance na imagem acima. O ponto X está dentro do alcance dessa nova torre?

b) Considere agora que uma terceira torre com alcance de 4 quilômetros foi instalada no ponto de $(-1; 4)$. Represente essa nova torre e o seu alcance na imagem acima. O ponto X está dentro do alcance dessa nova torre?

Por mais que a sua representação tenha sido feita com cuidado, é possível que você tenha ficado com dúvidas sobre a resposta para a questão **b)**, não?

c) Como podemos decidir, a partir das coordenadas da posição do celular e das torres em seu entorno, se o ponto X está dentro do alcance de cada uma das torres?

Embora possa haver limitações em termos de precisão ou de dificuldade em desenhar, nós podemos usar representações visuais para resolver questões, especialmente as que envolvem objetos geométricos. Porém, computadores não contam com esse recurso. De maneira geral, objetos geométricos são tratados por meio de suas coordendas e questões como essa (se um ponto está dentro ou fora de uma circunferência) são respondidas com o auxílio de alguns cálculos.

d) Descreva textualmente como proceder para determinar se um ponto no plano está dentro ou fora de uma circunferência dadas as coordenadas do ponto, as coordenadas do centro da circunferência e o seu raio. Sua descrição deve contemplar todos os passos da resolução e ser aplicável a quaisquer pontos e torres.

e) Aplique a sua descrição para o caso em que o ponto está na posição $(0; -2)$ e a torre foi instalada na posição $(1; -4)$ e tem alcance de 5 quilômetros.

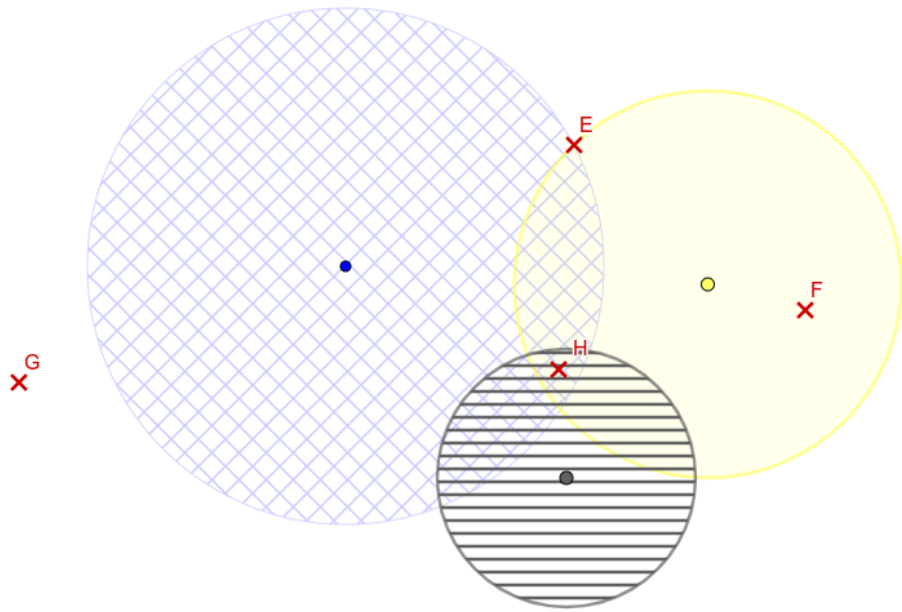
ORGANIZANDO SOLUÇÕES PARA INEQUAÇÕES

Ao responder a pergunta "um ponto dado está dentro do alcance de pelo menos uma das torres dadas?", estamos lidando com um conjunto de inequações e isso é diferente do que fazemos quando lidamos com sistemas de equações lineares, por alguns motivos, como vemos a seguir.

Primeiro, não estamos lidando com equações, mas sim inequações. Neste contexto, as equações correspondem às linhas que delimitam o alcance da torre, enquanto que as inequações correspondem ao círculo que demarca toda a região alcançada pela torre (ou a parte de fora desse círculo se invertermos o sinal de desigualdade).

Segundo, as equações envolvidas não são lineares pois as circunferências são dadas por equações que têm as variáveis e elevadas ao quadrado .

Terceiro, e mais importante, não estamos buscando uma solução para o sistema de inequações. Isso significaria encontrar as coordenadas de um ponto satisfazendo todas elas. O que nos interessa é se o ponto é solução de pelo menos uma das inequações.



Na figura acima, o ponto H é uma solução das três inequações que determinam os três círculos mostrados. O ponto E é solução de duas das equações que determinam as circunferências traçadas. Já o ponto F é solução de apenas uma das inequações que determinam os círculo e o ponto G não é solução de nenhuma dessas inequações.

Considerando esses quatro pontos, em qual deles há sinal?

EXPLORANDO TEM SINAL? - PARTE 2

Agora que você já sabe como resolver esse problema a partir das coordendadas dos pontos envolvidos, podemos utilizar essa abordagem para criar um algoritmo que diga se há sinal em um ponto a partir das informações de várias torres.

Mas antes de começar a implementar a solução em Portugol, vejamos alguns detalhes da linguagem que pode afetar a sua solução:

- Para elevar um número ao quadrado ou extrair a sua raiz quadrada, você vai precisar usar comandos que não fazem parte das funcionalidades básicas do Portugol, mas estão disponíveis como parte de uma biblioteca chamada Matematica. Isso é simples, e de fato está explicado na seção sobre o método da bissecção ou pode ser entendido com auxílio deste vídeo. Mas para evitar esses pontos, podemos simplesmente usar o fato de que $x^2 = x \cdot x$;
- Os símbolos de desigualdades disponíveis em Portugol são: < (menor que), > maior que, <= (menor ou igual a) e >= (maior ou igual a)

Tem sinal? - parte 2

Atividade 7

Nesta atividade, você deve escrever um algoritmo que leia as informações a partir de um formato específico. Primeiro, o usuário irá digitar as coordenadas x e y, nessa ordem, do ponto para o qual vamos analisar a existência de sinal. Esses dois valores devem ser variáveis inteiras. Depois, um número inteiro referente ao número de torres que serão consideradas. Por fim, para cada torre serão dados três valores (todos reais) representando a sua coordenada x, coordenada y e alcance. Segue um exemplo explicado:

```
2     (coordenada x do ponto)
-1    (coordenada y do ponto)
3     (quantidade de torres)
-10   (coordenada x da torre 1)
3     (coordenada y da torre 1)
6     (alcance da torre 1)
7     (coordenada x da torre 2)
6     (coordenada y da torre 2)
4     (alcance da torre 2)
10    (coordenada x da torre 3)
1     (coordenada y da torre 3)
7     (alcance da torre 3)
```

A resposta do seu algoritmo deve ser um único caractere: `S` ou `N`, se houver ou não sinal no ponto dado. No caso do exemplo dado, a saída deve ser: `N`

a) Escreva um algoritmo em Portugol que resolva o problema proposto seguindo o formato de entrada descrito acima.

Você pode usar o exemplo acima para testar a sua solução.

b) A seguir estão disponíveis mais alguns casos para você testar com o seu algoritmo.

Entrada	Entrada	Entrada
0	-1	2
0	1	-1
2	1	4
3	-5	10
3	1	10
4	5	10
-2	Saída	-2
-2	N	-2
3		4
Saída		0
S		6
		7
		1
		-5
		5
		Saída
		N

ORGANIZANDO REGIÕES NO PLANO

Você estudou no módulo sobre sistemas lineares o significado geométrico da resolução de sistemas de equações lineares: equações lineares determinam retas e a solução de sistemas com duas ou mais equações lineares determinam os pontos de intersecção entre essas retas. Resolver uma inequação significa, geometricamente, encontrar os pontos de um intervalo ou região que satisfazem essa inequação. Portanto, resolver um sistema de inequações significa encontrar os pontos que estejam localizado na sobreposição (ou intersecção) de todas as regiões determinadas por cada uma das inequações.

Uma inequação linear de duas variáveis determina um semi-plano e um sistema de inequações lineares determina regiões que resultam da sobreposição desses semiplanos, podendo ser usados para determinar a parte interior de um polígono, por exemplo. Seguindo esse raciocínio, o interior de um pentágono poderia ser dado por um sistema de 5 inequações lineares e é possível checar se um ponto está dentro do pentágono verificando se as coordenadas do ponto satisfazem todas a inequações. Porém, essa é apenas uma das maneiras de fazer essa checagem para o caso de um polígono. Com ferramentas que você poderá estudarno Ensino Superior, mais especificamente no curso de Geometria Analítica, outros métodos estarão disponíveis.

Relembrando que o problema que resolvemos aqui não pretendia verificar se um ponto satisfaz um sistema de inequações, mas sim se ele satisfaz pelo menos uma das inequações de um conjunto de inequações.

PARA SABER + A OLIMPÍADA BRASILEIRA DE INFORMÁTICA

Você talvez não saiba, mas ao longo das seções deste módulo você resolveu várias questões que fizeram parte da Olimpíada Brasileira de Informática. Alguns exemplos são os problemas Cometa, Conteineres, Xadrez e Campeonato.

Nessa olimpíada, os participantes devem resolver 3 ou 4 problemas em um período de tempo delimitado e submeter as suas soluções, que são pontuadas automaticamente. O sistema usado para isso depende de instruções muito rígidas para a entrada e saída de cada algoritmo a ser resolvido, mas agora que você aprendeu sobre isso e já conhece os conceitos básicos de uma linguagem de programação, pode participar dessa competição!

Todos os problemas já propostos nessa olimpíada estão disponíveis em `olimpiada.ic.unicamp.br/pratique`, organizados por ano, nível e fase. Além disso, o sistema de correção de todos eles está disponível para que você possa testar as suas próprias soluções.

Infelizmente, a OBI não aceita algoritmos escritos em Portugol, mas ela oferece alguns materiais de estudo alinhados ao estilo da competição em `olimpiada.ic.unicamp.br/estude`. São referências muito boas para quem deseja aprender mais sobre programação de computadores, mesmo que não planeje participar da olimpíada.

Referências Bibliográficas

Brasil (2018). *Base Nacional Comum Curricular*. Ministério da Educação, Brasília, Brasil. Disponível em: http://basenacionalcomum.mec.gov.br.

Disessa, A. A. (2018). A computation literacy and "the big picture" concerning computers in mathematics education. *Mathematical Thinking and Learning*, 20(1):3–31.

Esteves, A., Noschang, L., Raabe, A. e Filho, A. (2019). Portugol studio: Em direção a uma comunidade aberta para pesquisa sobre o aprendizado de programação. Em *Anais do XXVII Workshop sobre Educação em Computação*, páginas 513–522, Belém, Brasil. Sociedade Brasileira de Computação.

Li, Y., Schoenfeld, A. H., diSessa, A. A., Graesser, A. C., Benson, L. C., English, L. D. e Duschl, R. A. (2020). Computational thinking is more about thinking than computing. *Journal for STEM Education Reserach*, páginas 1–18.

Noschang, L., Pelz, F., de Jesus, E. e Raabe, A. (2014). Portugol studio: Ide para iniciantes em programação. Em *Anais do XXII Workshop sobre Educação em Computação*, páginas 1–10, Braslília, Brasil. Sociedade Brasileira de Computação.

Raabe, A., Zorzo, A. F. e Blikstein, P. (2020). *Computação na Educação Básica: Fundamentos e Experiências*. Penso Editora, Porto Alegre, Brasil.

Shute, V. J., Sun, C. e Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, 22:142–158.