PCS 2190 - AL: Algoritmos

Ricardo Nakamura e Romero Tori

2015

1 O que são algoritmos?

Informalmente, pode-se dizer que um algoritmo é uma sequência de ações bem definidas, que são realizadas na ordem especificada e que ocorrem dentro de um limite de tempo. Uma receita de bolo é um bom exemplo de algoritmo:

1. Misture gemas de ovos e açúcar até obter um creme;
2. Misture farinha aos poucos, formando uma massa uniforme;
3. Acrescente fermento misturando lentamente;
4. Despeje a massa em uma forma;
5. Coloque a forma em forno pré-aquecido;
6. Deixe assar até a massa dourar.

Outros exemplos de algoritmos encontrados no dia-a-dia são instruções de montagem de móveis e roteiros de ruas para chegar a um destino ("siga pela rua X, vire à direita na terceira esquina...") – você consegue pensar em outros exemplos?

Programas de computador também são algoritmos: listas de ações que devem ser executadas sequencialmente, na ordem em que aparecem. No entanto, devido a limitações dos próprios computadores, essas ações precisam ser especificadas de uma maneira muito mais objetiva do que em uma comunicação com pessoas.

Atividade 1

Escreva um algoritmo para descrever a construção de um cubo de papelão, a partir de uma única folha de papelão (usando ferramentas que você escolher: tesoura, estilete, cola etc.)

2 Comandos para o computador

Ao contrário de receitas ou roteiros de ruas, o computador só é capaz de interpretar um conjunto limitado de operações, que veremos em detalhes durante o curso:

- Comandos sequenciais - ações simples ou complexas que devem ser executadas em ordem;
- Comandos condicionais - escolhas a serem feitas, dependendo de condições;
- Comandos de repetição - fazer uma mesma ação várias vezes.

Desta forma, nossos algoritmos podem ser expressos a partir destas operações. No entanto, uma vez que o algoritmo esteja definido desta forma, ele pode ser transformado em um programa escrito em diferentes linguagens de programação. Nesse curso, usaremos o ambiente de programação Processing (que é baseado na linguagem Java) para as atividades práticas.

Além dos comandos, precisamos também de uma forma de representar as informações que estão disponíveis para o computador. Essas informações podem ser classificadas com base na sua mutabilidade em constantes e variáveis. Para cada informação, daremos um nome adequado. Por exemplo, “cor”, “nome”, “comprimento”. Obs: em alguns exemplos, usaremos apenas letras como “A”, “B” ou “C”, quando não existe um significado específico para uma dada informação.

3 Pensando em programas como algoritmos

Pensar em programas de computador como algoritmos corresponde a entendê-los como uma sequência de comandos com um objetivo. Isto significa que projetar um algoritmo (ou escrever um programa de computador) consiste em definir essa sequência, utilizando raciocínio lógico e outros conhecimentos.

Imagine a seguinte situação: enquanto caminha pela rua, alguém num carro lhe pergunta sobre como chegar a uma loja, que deveria estar próxima. Provavelmente, você tentaria se lembrar da localização da loja, das mãos das ruas próximas e então descreveria um roteiro a partir da rua em que está até chegar ao local desejado. O mesmo tipo de raciocínio se aplica na criação de um algoritmo computacional.

3.1 Descrevendo um algoritmo

Quando começamos a esboçar um algoritmo, pode ser útil não ter de pensar em todos os detalhes da linguagem de programação que será utilizada. Por este motivo, é comum se escrever algoritmos utilizando as chamadas "pseudo-linguagens", que nada mais são do que formas textuais para se representar as operações básicas de um computador. A listagem a seguir apresenta um exemplo de um algoritmo para selecionar o maior entre três números, usando uma *pseudo-linguagem*, conhecida como "português estruturado" ou "portugol".

```
A = número digitado pelo usuário
B = número digitado pelo usuário
C = número digitado pelo usuário
se A > B então
    se A > C então
        MAIOR = A
    senão
        MAIOR = C
senão
    se B > C então
        MAIOR = B
    senão
        MAIOR = C
```

Como se pode observar no exemplo, existe uma sequência de ações (ou instruções) que devem ser seguidas mas, em vez de serem organizadas em frases, elas assumem uma estrutura mais parecida com uma lista – evidenciando, assim, a sequencialidade. Além disso, as palavras em negrito possuem significado especial, correspondendo, nesse caso, a ações de seleção (ou condicionais). Como existem inúmeras pseudo-linguagens, vamos propor a seguinte convenção para uso em atividades desta disciplina:

Operação	Pseudo-linguagem
Atribuição	X = ...
Condicional	SE ... ENTÃO ... SENÃO ...
Repetição	ENQUANTO ... FAÇA ...

Como alternativa ao sinal "=", você també pode utilizar uma seta $\leftarrow$ se preferir.

O exemplo a seguir mostra o uso da operação de repetição para calcular a divisão A/B. Observe que as operações que devem ser executadas condicionalmente ou dentro de um laço de repetição devem ser escritas com uma indentação em relação ao restante do texto.

```
A = número digitado pelo usuário
B = número digitado pelo usuário
DIVISAO = 0
enquanto A ≥ B faça
    A = A - B
    DIVISAO = DIVISAO + 1
mostrar o valor de DIVISAO
```

A última linha do exemplo ("mostrar o valor de DIVISAO") é uma forma de anotarmos que o resultado deve ser mostrado para o usuário de alguma forma. Como estamos descrevendo o algoritmo de forma abstrata, não importa saber exatamente como isso será feito.

3.2 Dividindo o problema em partes

Como foi discutido anteriormente, projetar um algoritmo corresponde a pensar em uma sequência de operações que solucionam um problema.

Em muitas situações, esse processo pode ser simplificado se o problema for dividido em partes menores. Desta forma, podemos trabalhar na solução de cada um destes sub-problemas separadamente e, a partir dessas, construir a solução do problema original. O desafio está em se decompor o problema sem deixar de lado nenhuma de suas características originais.

Este tipo de abordagem também permite abstrair (considerar níveis superiores de abstração, nos quais os detalhes de níveis inferiores são omitidos) partes do problema que já estão resolvidas ou que não nos interessam no momento. Por exemplo, o algoritmo para desenhar um retângulo pode ser descrito como:

> Desenhar um segmento de reta horizontal
> Desenhar um segmento de reta horizontal a uma distância H abaixo do primeiro
> Desenhar segmentos de reta verticais conectando as extremidades mais próximas de cada segmento horizontal

Neste algoritmo, a operação de *desenhar segmentos de reta* foi abstraída. Se necessário, podemos definir um outro algoritmo para descrever esta operação. Retornando a nossas analogias culinárias, podemos dividir a receita de uma macarronada em duas partes: a preparação do molho e o cozimento do macarrão. Se formos utilizar um molho pronto, não é preciso detalhar esta parte da receita.

3.3 Diferentes níveis de detalhamento

Além da divisão de um problema em partes menores, podemos começar a planejar um algoritmo em níveis abstratos, progressivamente aumentando o nível de detalhamento até que o algoritmo esteja descrito somente com instruções básicas de lógica de programação: definição e atribuição de variáveis, condicionais e laços de repetição (estas instruções básicas também são chamadas de “comandos primitivos”).

Na prática, as técnicas de dividir um problema em partes e de trabalhar com níveis progressivos de detalhamento são usadas juntas com frequência. Como exemplo, vamos planejar um algoritmo para criar uma lista com os números primos entre 2 e 100. A forma mais simples deste algoritmo pode ser algo como:

```
X ← 2
enquanto X < 100 faça
    se X é primo então
        acrescentar X na lista
    X ← X + 1
imprimir a lista
```

Como podemos observar, as operações *se X é primo*, *acrescentar X na lista* e *imprimir a lista* foram colocadas de forma abstrata nesta versão. No entanto, precisamos aumentar o detalhamento delas. Sabendo que um número primo só é divisível por 1 e por si próprio, podemos expandir o algoritmo, reduzindo o seu nível de abstração:

```
X ← 2
enquanto X < 100 faça
    Y ← 2
    enquanto Y < X faça
        se X é divisível por Y então
            X não é primo
        Y ← Y + 1
    se X é primo então
        acrescentar X na lista
    X ← X + 1
imprimir a lista
```

Da mesma forma, poderíamos aumentar o detalhamento das outras operações progressivamente, até chegarmos a um algoritmo completo.

3.4 Refinamentos sucessivos

Construir um algoritmo ótimo na primeira tentativa é uma situação rara. É preciso ter consciência deste fato, porque é comum que programadores com pouca experiência acreditem no contrário. Uma vez que temos uma proposta inicial para a solução do problema, devemos avaliar criticamente o algoritmo que foi criado. Se o algoritmo já estiver implementado na forma de um programa, esta avaliação pode ser feita também através de programas de teste ou sessões de depuração, para observar o seu funcionamento.

Atividade 2

Utilizando os conceitos e técnicas discutidos, escreva, em pseudo-linguagem, um algoritmo que recebe três números positivos A, B e C e os escreve em sequência do menor para o maior.

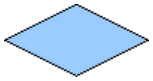
4 Representações gráficas de algoritmos

Desde o surgimento da computação, foram propostas maneiras de se representar algoritmos graficamente. Como ferramentas de visualização de informação, esses diferentes tipos de diagramas realçam as relações de ordem temporal das sequências de operações (ou comandos) e as ramificações ou variações de comportamento do algoritmo.

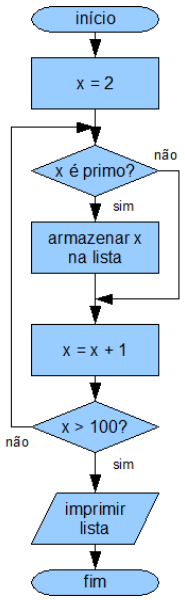
Uma desvantagem das representações gráficas é a complexidade resultante dos diagramas ao se expressar um algoritmo extenso. Em função disso, frequentemente opta-se pelo uso de pseudo-linguagens, como discutido anteriormente, para descrever de forma abstrata um algoritmo.

Apesar de suas limitações, diagramas ainda são utilizados na descrição de algoritmos, especialmente para se criar representações abstratas com pouco detalhamento, realçando suas principais operações. A existência de programas voltados para o desenho destes diagramas também simplifica sua construção. Por fim, eles constituem um caso interessante de design e visualização de informação. Uma das representações gráficas mais conhecidas é o fluxograma.

Fluxogramas representam um algoritmo como uma série de blocos de diferentes formatos, ligados por setas, que indicam a sequência das operações. Os principais símbolos de um fluxograma são mostrados na tabela a seguir. Observe que o fluxograma trata de forma distinta operações de entrada e saída (digitação de uma informação pelo usuário ou impressão de uma informação na tela, por exemplo). Um “processo” no fluxograma corresponde a um ou mais comandos.

Operação	Símbolo
Início ou fim do algoritmo	
Processo	
Condicional	
Entrada/saída	

A figura a seguir apresenta um exemplo de fluxograma correspondente à primeira versão do algoritmo para listar os números primos entre 2 e 100. Como se pode observar, um laço de repetição é representado simplesmente através de uma operação condicional que retorna a um ponto anterior do diagrama.



Atividade 3

Represente, na forma de fluxograma, o algoritmo que foi desenvolvido na atividade anterior.

5 Conclusão

Nesta aula, o conceito de algoritmo foi apresentado, assim como a sua aplicação em programas de computador. Algumas práticas utilizadas no projeto de algoritmos foram discutidas, com exemplos de aplicação. Por fim, os fluxogramas, uma forma gráfica de se descrever um algoritmo, foi apresentada.

Atividade 4

Quando escrevemos um algoritmo para interpretação humana (por exemplo, uma receita), nem sempre é necessário especificar todos os detalhes, pois somos capazes de improvisar, a partir de experiências anteriores e conhecimento em nossa memória. Discuta o que ocorre caso um computador tenha de executar um algoritmo "incompleto".

Atividade 5

Desafio: Utilizando os conceitos e técnicas discutidos, escreva, em pseudo-linguagem, um algoritmo para desenhar um polígono regular de 3 a 10 lados. Considere que já existe uma operação *desenhar linha* que é capaz de desenhar um segmento de reta entre dois pontos ou desenhar um segmento de reta com um certo comprimento, em uma dada direção ou ângulo.