capítulo 1

introdução e conceitos básicos

Este capítulo faz uma apresentação da matemática discreta, de sua importância, de seus conceitos básicos e de seus usos. Também apresenta uma revisão dos conceitos básicos de teoria dos conjuntos, pré-requisitos para todo o texto subsequente. Esses conceitos básicos são instanciados em alguns fundamentos de computação e informática como alfabetos, palavras e linguagens e ilustram o seu uso em linguagens de programação como Pascal.

1.1 introdução à matemática discreta

Praticamente qualquer estudo em computação e informática, teórico ou aplicado, exige como pré-requisito conhecimentos de diversos tópicos de matemática. Tal fato é normalmente explicitado na maioria dos livros de computação e informática, sendo que alguns possuem um capítulo específico em que tais tópicos são resumidamente introduzidos.

A importância da matemática é explicitada nas Diretrizes Curriculares do MEC para cursos de computação e informática (Brasil, [20--?]), como segue (a matéria matemática é integrante da área de formação básica):

> *A formação básica tem por objetivo introduzir as matérias necessárias ao desenvolvimento tecnológico da computação. O principal ingrediente desta área é a ciência da computação, que caracteriza o egresso como pertencente à área de computação. A maioria das matérias tecnológicas são aplicações da ciência da computação. São matérias de formação básica dos cursos da área de computação: a ciência da computação, a* ***matemática****, a física e eletricidade e a pedagogia.*

Especificamente em relação à matéria matemática, o texto destaca que:

> *A matemática, para a área de computação, deve ser vista como uma ferramenta a ser usada na definição formal de conceitos computacionais (linguagens, autômatos, métodos, etc). Os modelos formais permitem definir suas propriedades e dimensionar suas instâncias, dadas suas condições de contorno.*

Assim, este livro contém uma seleção de tópicos de matemática, os quais são essenciais para o estudo de computação e informática, tanto na área de formação básica como na de formação tecnológica. Esta seleção de tópicos é comumente denominada de *matemática discreta*:

> *Considerando que a maioria dos conceitos computacionais pertencem ao domínio do discreto, a* ***matemática discreta*** *(ou também chamada álgebra abstrata) é fortemente empregada.*

Deve-se observar que não é objetivo desta publicação cobrir todos os tópicos de matemática discreta. Assim, alguns temas como *análise combinatória* e *probabilidade discreta* não são desenvolvidos. Adicionalmente, apenas alguns tópicos de *teoria dos grafos* são introduzidos.

Uma questão importante para o entendimento do que segue é a origem do termo matemática discreta. Intuitivamente falando, qualquer sistema computador possui limitações finitas em todos os seus principais aspectos como, por exemplo, tamanho da memória, número de instruções que pode executar, número de símbolos diferentes que pode tratar, etc. Assim, o estudo dos *conjuntos finitos* é fundamental.

O fato de um sistema computador possuir limitações finitas não implica necessariamente uma limitação ou pré-fixação de tamanhos máximos. Por exemplo, no que se refere à capacidade de armazenamento, um computador pode possuir unidades auxiliares como discos removíveis, fitas, etc. Portanto, para um correto entendimento de diversos aspectos computacionais, frequentemente não é possível pré-fixar limites, o que implica tratar tais questões em um contexto infinito.

Entretanto, qualquer conjunto de recursos computacionais, finito ou infinito, é *contável* ou *discreto* (em oposição ao termo *contínuo*), no sentido de que seus elementos (recursos) podem ser *enumerados* ou sequenciados (segundo algum critério) de tal forma que não existe um elemento entre quaisquer dois elementos consecutivos da enumeração. Por exemplo, o conjunto dos números naturais é obviamente contável. Um importante contraexemplo é o conjunto dos números reais, que é *não contável* ou *não discreto*. Isso significa que existem conjuntos infinitos contáveis e conjuntos infinitos não contáveis.

Assim, a *matemática discreta* possui como ênfase os estudos matemáticos baseados em conjuntos contáveis, finitos ou infinitos. Em oposição, a *matemática do continuum* possui como ênfase os estudos matemáticos baseados em conjuntos não contáveis. Um importante exemplo de *matemática do continuum* é o *cálculo diferencial e integral*.

Com o objetivo de manter o leitor motivado com o desenvolvimento do conteúdo, sempre que possível, para cada conceito de matemática discreta introduzido, são discutidas aplicações típicas de diversas áreas da computação e informática.

1.2 → conceitos básicos de teoria dos conjuntos

O texto que segue introduz alguns conceitos básicos relativos à *teoria dos conjuntos* que, possivelmente, são do conhecimento do leitor. Neste caso, sugere-se uma rápida passagem para verificar as nomenclaturas e convenções adotadas ao longo do livro, bem como a leitura da exemplificação de seu uso em computação e informática.

1.2.1 conjuntos

O conceito de conjunto é fundamental, pois praticamente todos os conceitos desenvolvidos em computação e informática, bem como os correspondentes resultados, são baseados em conjuntos ou construções sobre conjuntos.

Conjunto é uma estrutura que agrupa objetos e constitui uma base para construir estruturas mais complexas. Assim, informalmente, um conjunto é uma coleção, sem repetições e sem qualquer ordenação, de objetos denominados *elementos*. O termo "elemento" é usado de forma ampla e pode designar um objeto concreto ou abstrato. Neste contexto, um elemento é uma entidade básica que não é definida formalmente.

definição 1.1 – Conjunto

Um *conjunto* é uma coleção de zero ou mais objetos distintos, chamados *elementos* do conjunto, os quais não possuem qualquer ordem associada. ❑

exemplo 1.1 – Conjuntos

a As vogais **a**, **e**, **i**, **o**, **u**;
b O par de sapatos preferido;
c Os dígitos **0**, **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8** e **9**;
d Todos os brasileiros;
e Os números pares **0**, **2**, **4**, **6**,...
f O personagem **Snoopy**, a letra **a**, a **baía da Guanabara** e o **Pelé**. ❑

Observe que um conjunto pode ser definido listando-se todos os seus elementos (como "as vogais **a**, **e**, **i**, **o**, **u**") ou por propriedades declaradas (como "todos os brasileiros"). Adicionalmente, deve ficar claro que um conjunto não necessariamente é constituído por objetos que compartilham as mesmas características ou propriedades (como em "o personagem **Snoopy**, a letra **a**, a **baía da Guanabara** e o **Pelé**").

A definição de um conjunto listando todos os seus elementos é denominada *denotação por extensão* e é dada pela lista de todos os seus elementos, em qualquer ordem (separados por vírgulas) e entre chaves, como por exemplo:

$$\textbf{Vogais} = \{ \textbf{a, e, i, o, u} \}$$

Neste caso, **Vogais** denota o conjunto **{ a, e, i, o, u }**.

A definição de um conjunto por propriedades é denominada *denotação por compreensão* como, por exemplo:

$$\textbf{Pares} = \{ \textbf{n} \mid \textbf{n} \text{ é número par} \}$$

a qual é interpretada como:

o conjunto de todos os elementos ***n*** tal que ***n*** é número par

Assim, a forma geral de definição de um conjunto por propriedades é como segue:

$$\{ \textbf{x} \mid \textbf{p(x)} \}$$

e é tal que um determinado elemento **a** é elemento deste conjunto se a propriedade **p** é verdadeira para **a**, ou seja, se **p(a)** é verdadeira. Por exemplo, para o conjunto:

$$\textbf{B} = \{ \textbf{x} \mid \textbf{x} \text{ é brasileiro} \}$$

tem-se que **Pelé** é elemento de **B** e **Bill Gates** *não* é elemento de **B**.

Muitas vezes é conveniente especificar um conjunto de outra forma que não por compreensão, como, por exemplo:

Dígitos = { 0, 1, 2, 3,..., 9 }
Pares = { 0, 2, 4, 6,... }

nos quais os elementos omitidos podem ser facilmente deduzidos do contexto.

exemplo 1.2 – Conjuntos

a **Semana = { seg, ter, qua, qui, sex, sab, dom }**

b **Duas Vogais = { aa, ae, ai, ao, au, ea, ee, ei, eo, eu,...,ua, ue, ui, uo, uu }**

c **{ x | x = y^2** sendo que **y** é número inteiro }
o que corresponde ao conjunto **{ 0, 1, 4, 9, 16,... }** ❑

1.2.2 pertinência

Se um determinado elemento **a** é elemento de um conjunto **A**, tal fato é denotado por:

$$a \in A$$

que é interpretado como segue:

a pertence *ao conjunto A*

Caso contrário, afirma-se que **a** *não* pertence ao conjunto **A**. Tal fato é denotado por:

$$a \notin A$$

exemplo 1.3 – Pertence, não pertence

a Relativamente ao conjunto **Vogais = { a, e, i, o, u }**, tem-se que:
a ∈ Vogais
h ∉ Vogais

b Relativamente ao conjunto **B = { x | x** é brasileiro **}**, tem-se que:
Pelé ∈ B
Bill Gates ∉ B ❑

1.2.3 alguns conjuntos importantes

Um conjunto especialmente importante é o *conjunto vazio*, ou seja, o conjunto sem elementos **{ }**, usualmente representado pelo seguinte símbolo:

$$\varnothing$$

exemplo 1.4 – Conjunto vazio

a o conjunto de todos os brasileiros com mais de **300** anos;

b o conjunto de todos os números que são simultaneamente pares e ímpares. ❑

Um tipo de conjunto quase tão importante como o vazio é o *conjunto unitário*, ou seja, um conjunto constituído por um único elemento. Existem infinitos conjuntos unitários. Entretan-

to, para muitas aplicações, pode-se usar qualquer conjunto unitário, ou seja, o fato importante é que o conjunto considerado possui um único elemento, sendo irrelevante qual é o elemento que o constitui. Nesse caso, um conjunto unitário fixado é usualmente denotado por **1**.

exemplo 1.5 – Conjunto unitário

a o conjunto constituído pelo jogador de futebol Pelé;

b o conjunto de todos os números que são simultaneamente pares e primos;

c $\mathbf{1 = \{ * \}}$ ❑

Os seguintes conjuntos (os quais devem ser do conhecimento do leitor) são importantes na matemática em geral e na computação e informática em particular e possuem uma denotação universalmente aceita:

N	*conjunto dos números naturais*
Z	*conjunto dos números inteiros*
Q	*conjunto dos números racionais*
I	*conjunto dos números irracionais*
R	*conjunto dos números reais*

1.2.4 conjuntos finitos e infinitos

Um conjunto pode possuir um número finito ou infinito de elementos. A definição formal de conjunto finito e infinito será apresentada adiante. Informalmente, um conjunto é dito:

a *Conjunto finito* se pode ser denotado por extensão, ou seja, listando exaustivamente todos os seus elementos;

b *Conjunto infinito*, caso contrário.

exemplo 1.6 – Conjunto finito, conjunto infinito

a Os seguintes conjuntos são *finitos*:

$\varnothing$

$\mathbf{\{ \varepsilon \}}$

Vogais = { a, e, i, o, u }

Dígitos = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

{ Snoopy, a, baía da Guanabara, Pelé }

$\mathbf{A = \{ x \in N \mid x > 0}$ e $\mathbf{x < 4 \}}$

B = { x | **x** é brasileiro **}**

b Os seguintes conjuntos são *infinitos*:

Z

R

$\mathbf{\{ x \in Z \mid x \geq 0 \}}$

$\mathbf{Pares = \{ y \mid y = 2x}$ e $\mathbf{x \in N \}}$

Observe que o conjunto **A = { x ∈ N | x > 0 x < 4 }** é finito, pois poderia ser representado por extensão, ou seja, **A = { 1, 2, 3 }**. Analogamente para o conjunto **B = { x | x** é brasileiro **}** (a representação por extensão é claramente possível, embora trabalhosa). Também, repare que o conjunto **{ x ∈ Z | x ≥ 0 }** corresponde ao conjunto **N**. ❑

1.2.5 alfabetos, palavras e linguagens

A noção de conjunto permite definir *linguagem*, um dos conceitos mais fundamentais em computação e informática. Para a definição de linguagem, é necessário antes introduzir os conceitos de *alfabeto* e de *cadeia de caracteres*. O estudo detalhado destes e de outros conceitos correlatos é realizado em algumas disciplinas como *linguagens formais* e *compiladores*.

definição 1.2 – Alfabeto

Um *alfabeto* é um conjunto finito. Os elementos de um alfabeto são usualmente denominados de *símbolos* ou *caracteres*. ❑

Portanto, o conjunto vazio é um alfabeto, e um conjunto infinito *não* é um alfabeto.

definição 1.3 – Palavra, cadeia de caracteres, sentença

Uma *palavra* ou *cadeia de caracteres* ou *sentença* sobre um alfabeto é uma sequência finita de símbolos (do alfabeto) justapostos. ❑

Portanto, uma cadeia sem símbolos é uma palavra válida, e o símbolo:

ε denota a *cadeia vazia*, *palavra vazia* ou *sentença vazia*.

Se Σ representa um alfabeto, então:

Σ* denota o conjunto de todas as palavras possíveis sobre Σ.

exemplo 1.7 – Alfabeto, palavra

a Os conjuntos ∅ e **{ a, b, c }** são alfabetos;
b O conjunto **N** *não* é um alfabeto;
c ε é uma palavra sobre o alfabeto **{ a, b, c }**;
d ε é uma palavra sobre o alfabeto ∅;
e **a**,**e**,**i**,**o**,**u**,**ai**,**oi**,**ui** e **aeiou** são exemplos de palavras sobre **Vogais**;
f **1** e **001** são exemplos de palavras *distintas* sobre **Dígitos**;
g **{ a, b }* = { ε, a, b, aa, ab, ba, bb, aaa,… }**;
h **∅* = { ε }**.

definição 1.4 – Linguagem formal

Uma *linguagem formal*, ou simplesmente *linguagem*, é um conjunto de palavras sobre um alfabeto. ❑

exemplo 1.8 – Linguagem formal

Suponha o alfabeto $\Sigma = \{ a, b \}$. Então:

a O conjunto vazio e o conjunto formado pela palavra vazia são linguagens sobre Σ. Obviamente, $\varnothing \neq \{ \varepsilon \}$;

b O conjunto de *palíndromos* (palavras que têm a mesma leitura da esquerda para a direita e vice-versa) sobre Σ é um exemplo de linguagem (infinita):

$$\text{Palíndromos} = \{ \varepsilon, a, b, aa, bb, aaa, aba, bab, bbb, aaaa, \ldots \}. \quad ❑$$

exemplo 1.9 – Linguagens de programação

As linguagens de programação como Pascal, C e Java são linguagens sobre o alfabeto constituído por letras, dígitos e alguns símbolos especiais (como espaço, parênteses, pontuação, etc.). Nesse caso, cada programa na linguagem corresponde a uma palavra sobre o alfabeto. Ou seja, uma linguagem de programação é definida por todos os seus programas possíveis. Portanto, Pascal, Java, C, bem como qualquer linguagem de programação de propósitos gerais, são conjuntos infinitos. ❑

observação 1.10 – Compilador × pertinência à linguagem

Um *compilador* de uma linguagem de programação é um *software* que traduz um programa escrito na linguagem de programação (*linguagem fonte*) para um código executável no sistema computador (*linguagem objeto*). Em geral, um compilador é estruturado em duas grandes partes: análise (*análise léxica*, *análise sintática* e *análise semântica*) e síntese (*geração* e *otimização* de código executável). Resumidamente, a análise verifica se um dado programa fonte **p** é, de fato, um programa válido para a linguagem **L** em questão, ou seja, verifica se:

$$p \in L \quad ❑$$

1.2.6 subconjunto e igualdade de conjuntos

Além da noção de pertinência já introduzida, outra noção fundamental da teoria dos conjuntos é a de *continência*, que permite introduzir os conceitos de *subconjunto* e de *igualdade de conjuntos*.

Se todos os elementos de um conjunto **A** também são elementos de um conjunto **B**, então se afirma que **A** está *contido* em **B** e denota-se por:

$$A \subseteq B$$

ou, alternativamente, que **B** *contém* **A**, e denota-se por:

$$B \supseteq A$$

Nesse caso ($A \subseteq B$ ou $B \supseteq A$), afirma-se que **A** é *subconjunto* de **B**.

Adicionalmente, se $A \subseteq B$, mas existe $b \in B$ tal que $b \notin A$, então se afirma que **A** está *contido propriamente* em **B**, ou que **A** é *subconjunto próprio* de **B**, e denota-se por:

$$A \subset B$$

Ou, alternativamente, que **B** *contém propriamente* **A** e denota-se por:

$$\mathbf{B} \supset \mathbf{A}$$

Quando não é fato que $\mathbf{A} \subseteq \mathbf{B}$ (respectivamente, $\mathbf{A} \subset \mathbf{B}$), é usual denotar como segue:

$\mathbf{A} \not\subseteq \mathbf{B}$ (respectivamente, $\mathbf{A} \not\subset \mathbf{B}$)

Outra terminologia usual, mas que não será adotada neste livro, é a seguinte:

- *contido amplamente* significando contido;
- *contido estritamente* significando contido propriamente.

exemplo 1.11 – Continência, subconjunto

a $\{ a, b \} \subseteq \{ b, a \}$
b $\{ a, b \} \subseteq \{ a, b, c \}$, $\{ a, b \} \subset \{ a, b, c \}$
c $\{ 1, 2, 3 \} \subseteq \mathbf{N}$, $\{ 1, 2, 3 \} \subset \mathbf{N}$
d $\mathbf{N} \subseteq \mathbf{Z}$, $\mathbf{N} \subset \mathbf{Z}$
e $\varnothing \subseteq \{ a, b, c \}$, $\varnothing \subset \{ a, b, c \}$
f $\varnothing \subseteq \mathbf{N}$, $\varnothing \subset \mathbf{N}$ ❑

Um conjunto especial e importante é o *conjunto universo*, normalmente denotado por **U**, que contém todos os conjuntos que estão sendo considerados. Ou seja, o conjunto universo define o "contexto de discussão" (e portanto, **U** *não* é um conjunto fixo). Uma vez definido o conjunto universo **U**, para qualquer conjunto **A**, tem-se que:

$$\mathbf{A} \subseteq \mathbf{U}$$

Os conjuntos **A** e **B** são ditos *conjuntos iguais*, o que é denotado por:

$$\mathbf{A} = \mathbf{B}$$

se e somente se possuem exatamente os mesmos elementos. Formalmente, afirma-se que:

$$\mathbf{A} = \mathbf{B} \quad \text{se e somente se} \quad \mathbf{A} \subseteq \mathbf{B} \text{ e } \mathbf{B} \subseteq \mathbf{A}$$

exemplo 1.12 – Igualdade de conjuntos

a $\{ 1, 2, 3 \} = \{ x \in \mathbf{N} \mid x > 0 \text{ e } x < 4 \}$
b $\mathbf{N} = \{ x \in \mathbf{Z} \mid x \geq 0 \}$
c $\{ 1, 2, 3 \} = \{ 3, 3, 3, 2, 2, 1 \}$

Este último item ilustra claramente a definição de igualdade. De fato, é fácil verificar que:

$$\{ 1, 2, 3 \} \subseteq \{ 3, 3, 3, 2, 2, 1 \} \quad \text{e} \quad \{ 3, 3, 3, 2, 2, 1 \} \subseteq \{ 1, 2, 3 \}.$$

Observe que este exemplo também ilustra por que as repetições de elementos podem ser desconsideradas (se o leitor tiver alguma dúvida, revise a definição de continência). ❑

É importante distinguir claramente entre pertinência e continência. Sugere-se ler atentamente o exemplo que segue.

exemplo 1.13 – Pertinência × continência

Lembre-se de que os elementos de um conjunto podem ser quaisquer objetos. Em particular, podem ser conjuntos. Considere o conjunto $\mathbf{A} = \{ 1, 2, 3, \varnothing, \{a\}, \{b, c\} \}$. Então (justifique):

a $\{1\} \notin A, \{1\} \subseteq A$
b $\varnothing \in A, \varnothing \subseteq A$
c $\{a\} \in A, \{b, c\} \in A$
d $\{1, 2, 3\} \notin A, \{1, 2, 3\} \subseteq A$ ❑

observação 1.14 – Linguagem formal × conjunto de todas as palavras

Considere a Definição 1.4 – Linguagem Formal. Uma *linguagem formal* L sobre um alfabeto Σ pode alternativamente ser definida como um subconjunto de Σ^*, ou seja:

$$L \subseteq \Sigma^*$$ ❑

1.2.7 conjuntos nas linguagens de programação

Como já introduzido, ao longo de todo o livro são exemplificadas aplicações de matemática discreta em computação e informática. Uma aplicação constantemente explorada é a relação entre os conceitos matemáticos desenvolvidos e suas implementações em linguagens de programação. Entretanto, como conhecimentos de linguagem de programação não são pré-requisitos deste livro, a exemplificação será ilustrativa e, portanto, não é detalhada, nem formal.

Com o objetivo de manter um entendimento crescente e coerente da aplicação da matemática discreta em linguagens de programação, o texto é centrado na linguagem Pascal, por diversas razões, com destaque para as seguintes:

- é uma linguagem desenvolvida objetivando o ensino de programação;
- é formalmente bem definida, o que facilita o seu estudo matemático;
- inspirou diversas linguagens de programação comerciais;
- é disponível em diversos tipos de sistemas computadores;
- é frequentemente adotada como primeira linguagem de programação em cursos de computação e informática.

Para exemplificar conjuntos em linguagens de programação, é necessário antes introduzir o conceito de tipo de dado. Informalmente e resumidamente, um *tipo de dado* em computação e informática é um conjunto de objetos (dados) e certas operações sobre esses objetos. Considerando as limitações usuais dos sistemas computadores e objetivando manter a portabilidade dos *softwares*, algumas linguagens especificam os limites dos valores do tipo de dados, como os valores devem ser armazenados e como as operações devem ser processadas.

A grande maioria das linguagens de programação possui alguns tipos de dados predefinidos como, por exemplo:

- *Real* ou *Ponto Flutuante*
- *Inteiro*
- *Caractere*
- *Booleano* ou *Lógico*

Considerando as limitações físicas de representação de conjuntos infinitos ou de precisão de valores em um sistema computador (esse assunto normalmente é detalhado em disciplinas como *arquitetura de computadores* e *matemática computacional*), os tipos Real e Inteiro implementam um subconjunto próprio de **R** e de **Z**, respectivamente (bem como algumas operações tradicionais como adição, multiplicação, etc.).

O tipo Caractere implementa os caracteres usuais como letras e dígitos, bem como alguns símbolos especiais (parênteses, pontuação, aspas, etc.). O tipo Lógico implementa os valores lógicos `verdadeiro` e `falso`. Para ambos os tipos, são implementadas operações especiais, algumas das quais são discutidas ao longo deste livro.

Pode aparentar uma contradição, mas muitas linguagens de programação não possuem facilidades adequadas para definir e operar conjuntos. Em particular, Pascal oferece algum tratamento de conjuntos. De fato, em Pascal, pode-se definir tipos baseados em conjuntos *finitos*. Por exemplo, na última linha, o texto `'a'..'z'` denota o intervalo **[a,...,z]**):

```
cores set of (amarelo, vermelho, azul, branco, preto)
dias_semana set of (seg, ter, qua, qui, sex, sab, dom)
alfabeto set of 'a'..'z'
```

Adicionalmente, pode-se definir constantes e variáveis de um tipo conjunto. A definição de constantes de um tipo conjunto é realizada sempre por extensão, como, por exemplo:

```
[vermelho, amarelo, azul]
[ ]
[seg..dom]
[seg..sex]
['a', 'e', 'i', 'o', 'u']
```

os quais correspondem aos seguintes conjuntos, respectivamente:

{ vermelho, amarelo, azul }
∅
{ seg, ter, qua, qui, sex, sab, dom }
{ seg, ter, qua, qui, sex }
{ a, e, i, o, u }

A definição de variáveis de um tipo conjunto é realizada listando quais nomes (das variáveis) correspondem a quais tipos. Por exemplo, no que segue, são declaradas (definidas) cinco variáveis de três tipos:

```
cores_primarias: cores
feriado, semana, trabalho: dias_semana
vogais: alfabeto
```

Assim, os seguintes trechos de programas em Pascal:

```
cores_primarias := [vermelho, amarelo, azul]
feriado := [ ]
semana := [seg..dom]
trabalho := [seg..sex]
vogais := ['a', 'e', 'i', 'o', 'u']
```

correspondem, na teoria dos conjuntos, aos seguintes conjuntos e suas correspondentes denotações:

cores_primarias = { vermelho, amarelo, azul }
feriado = ∅
semana = { seg, ter, qua, qui, sex, sab, dom }
trabalho = { seg, ter, qua, qui, sex }
vogais = { a, e, i, o, u }

Observe que, no trecho de programa acima, foi usado o símbolo "`:=`" e não "`=`" para associar a variável ao seu valor. De fato, em Pascal, bem como na maioria das linguagens de programação, o símbolo "`=`" é usado exclusivamente para verificar uma igualdade, e não para definir ou atribuir valores. Essa distinção objetiva facilitar a construção do correspondente *compilador*. No desenvolvimento de compiladores, o tratamento de questões sintáticas é bem mais simples do que o tratamento de questões semânticas. Por essa razão, o teste da igualdade e a atribuição são *sintaticamente* distinguidos.

As noções de igualdade, subconjunto (contido) e pertinência também podem ser especificadas em Pascal, usando-se a seguinte simbologia:

`<=` (continência)
`=` (igualdade)

Por exemplo, suponha os trechos de programas acima e considere os seguintes trechos (observe que, no que segue, o símbolo "`=`" é usado com o sentido de igualdade):

```
cores_primarias = [vermelho, amarelo, azul]
feriado = trabalho
trabalho <= semana
[sab, dom] <= trabalho
'a' in vogais
dom in trabalho
```

os quais correspondem às seguintes proposições sobre a teoria dos conjuntos:

cores_primarias = { vermelho, amarelo, azul } (verdadeiro)
feriado = trabalho (falso)
trabalho ⊆ semana (verdadeiro)
{ sab, dom } ⊆ trabalho (falso)
a ∈ vogais (verdadeiro)
dom ∈ trabalho (falso)

1.3 → exercícios

exercício 1.1 Para cada conjunto abaixo:

- descreva de forma alternativa (usando outra forma de notação);
- diga se é finito ou infinito.

a Todos os números inteiros maiores que **10**
b **{ 1, 3, 5, 7, 9, 11,... }**
c Todos os países do mundo
d A linguagem de programação Pascal

exercício 1.2 Para **A = { 1 }**, **B = { 1, 2 }** e **C = { { 1 }, 1 }**, marque as afirmações corretas:

a $A \subset B$ [ ]
b $A \subseteq B$ [ ]
c $A \in B$ [ ]
d $A = B$ [ ]
e $A \subset C$ [ ]
f $A \subseteq C$ [ ]
g $A \in C$ [ ]
h $A = C$ [ ]
i $1 \in A$ [ ]
j $1 \in C$ [ ]
k $\{1\} \in A$ [ ]
l $\{1\} \in C$ [ ]
m $\varnothing \notin C$ [ ]
n $\varnothing \subseteq C$ [ ]

exercício 1.3 Sejam $a = \{ x \mid 2x = 6 \}$ e $b = 3$. Justifique ou refute a seguinte afirmação:

$$a = b$$

exercício 1.4 Quais são todos os subconjuntos dos seguintes conjuntos?

a **A = {a, b, c }**
b **B = { a, { b, c }, D }** dado que **D = { 1, 2 }**

exercício 1.5 O conjunto vazio está contido em qualquer conjunto (inclusive nele próprio)? Justifique a sua resposta.

exercício 1.6 Todo conjunto possui um subconjunto próprio? Justifique a sua resposta.

exercício 1.7 Sejam **A = { 0, 1, 2, 3, 4, 5 }**, **B = { 3, 4, 5, 6, 7, 8 }**, **C = { 1, 3, 7, 8 }**, **D = { 3, 4 }**, **E = { 1, 3 }**, **F = { 1 }** e **X** um conjunto desconhecido. Para cada item abaixo, determine quais dos conjuntos **A**, **B**, **C**, **D**, **E** ou **F** podem ser iguais a **X**:

a $X \subseteq A$ e $X \subseteq B$

b $X \not\subset B$ e $X \subseteq C$

c $X \not\subset A$ e $X \not\subset C$

d $X \subseteq B$ e $X \not\subset C$

exercício 1.8 Sejam **A** um subconjunto de **B** e **B** um subconjunto de **C**. Suponha que $a \in A$, $b \in B$, $c \in C$, $d \notin A$, $e \notin B$, $f \notin C$. Quais das seguintes afirmações são verdadeiras?

a $a \in C$

b $b \in A$

c $c \notin A$

d $d \in B$

e $e \notin A$

f $f \notin A$

exercício 1.9 Marque os conjuntos que são alfabetos:

a Conjunto dos números naturais [ ]

b Conjunto dos números primos [ ]

c Conjunto das letras do alfabeto brasileiro [ ]

d Conjunto dos algarismos arábicos [ ]

e Conjunto dos algarismos romanos [ ]

f Conjunto **{ a, b, c, d }** [ ]

g Conjunto das vogais [ ]

h Conjunto das letras gregas [ ]

exercício 1.10 Sejam Σ **= { a, b, c,..., z }** e **Dígitos = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }** alfabetos. Então:

a Para cada um dos alfabetos abaixo, descreva o correspondente conjunto de todas as palavras:

a.1) Σ

a.2) **Dígitos**

b Discuta as seguintes afirmações:

b.1) Português é uma linguagem sobre Σ, ou seja, é um subconjunto de Σ^*
Dica: Quais os símbolos usados para compor um texto em português?

b.2) **N** é uma linguagem sobre **Dígitos**, ou seja, é um subconjunto de **Dígitos***

b.3) **N = Dígitos***
Dica: Como fica o caso da palavra vazia?

exercício 1.11 Em que condições o conjunto de todos os palíndromos sobre um alfabeto constitui uma linguagem *finita*?

exercício 1.12 Para que o leitor se convença plenamente da importância da matemática discreta para a computação e informática, sugere-se, como exercício complementar, duas pesquisas na *internet*, a saber:

a Uma sobre currículos de cursos de computação e informática no mundo, e sua relação com matemática discreta. Observe que algumas vezes matemática discreta é denominada de álgebra;

b Outra sobre a importância da matemática discreta para a computação e informática e o detalhamento do porquê do termo "discreta".

exercício 1.13 Mesmo que o leitor não tenha conhecimentos de linguagens de programação, é possível, com um rápido estudo, desenvolver programas simples em Pascal. Assim, sugere-se como exercício complementar de pesquisa, o desenvolvimento de um programa em Pascal que implemente os trechos de programas exemplificados neste capítulo.