Capítulo 7

Controle de Fluxo de Execução

Até o momento os algoritmos estudados utilizam apenas instruções primitivas de atribuição, e de entrada e saída de dados. Qualquer conjunto de dados fornecido a um algoritmo destes será submetido ao mesmo conjunto de instruções, executadas sempre na mesma seqüência.

No entanto, na prática muitas vezes é necessário executar ações diversas em função dos dados fornecidos ao algoritmo. Em outras palavras, dependendo do conjunto de dados de entrada do algoritmo, deve-se executar um conjunto diferente de instruções. Além disso, pode ser necessário executar um mesmo conjunto de instruções um número repetido de vezes. Em resumo, é necessário controlar o fluxo de execução das instruções (a seqüência em que as instruções são executadas num algoritmo) em função dos dados fornecidos como entrada ao mesmo.

Neste capítulo serão estudadas as estruturas básicas de controle do fluxo de instruções de um algoritmo. De acordo com o modo como este controle é feito, estas estruturas são classificadas em:

- estruturas seqüenciais;
- estruturas de decisão;
- estruturas de repetição.

7.1 Comandos Compostos

Um **comando composto** é um conjunto de zero ou mais comandos (ou instruções) simples, como atribuições e instruções primitivas de entrada ou saída de dados, ou alguma das construções apresentadas neste capítulo.

Este conceito é bastante simples e será útil e conveniente nos itens seguintes, na definição das estruturas básicas de controle de execução.

7.2 Estrutura Seqüencial

Na **estrutura seqüencial** os comandos de um algoritmo são executados numa seqüência pré-estabelecida. Cada comando é executado somente após o término do comando anterior.

Em termos de fluxogramas, a estrutura seqüencial é caracterizada por um único fluxo de execução (um único caminho orientado) no diagrama. Em pseudocódigos, a estrutura seqüencial caracteriza-se por um conjunto de comandos dispostos ordenadamente. Como exemplos de aplicação desta estrutura de controle tem-se os algoritmos do capítulo anterior, onde não há estruturas de decisão ou de repetição.

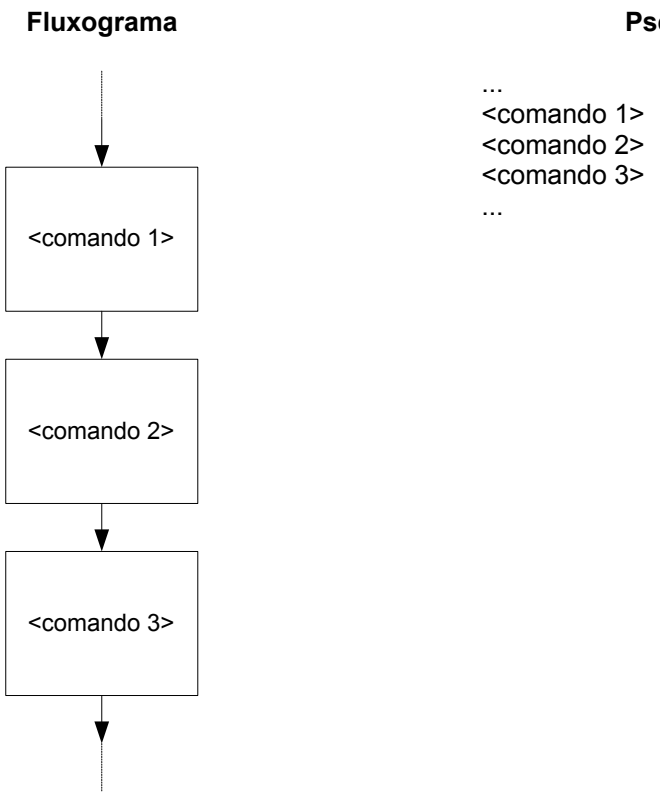


Figura 7.1 Trecho seqüencial de um algoritmo.

7.3 Estruturas de Decisão

Neste tipo de estrutura o fluxo de instruções a ser seguido é escolhido em função do resultado da avaliação de uma ou mais condições. Uma **condição** é uma expressão lógica.

A classificação das estruturas de decisão é feita de acordo com o número de condições que devem ser testadas para que se decida qual o caminho a ser seguido. Segundo esta classificação, têm-se dois tipos de estruturas de decisão:

- Se
- Escolha

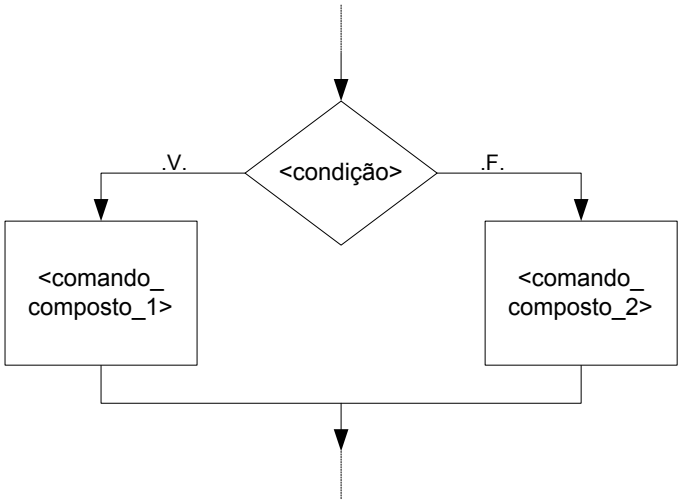
7.3.1 Estruturas de Decisão do Tipo Se

Nesta estrutura uma única condição (expressão lógica) é avaliada. Se o resultado desta avaliação for verdadeiro (.V.), então um determinado conjunto de instruções (comando composto) é executado. Caso contrário, ou seja, quando o resultado da avaliação for falso (.F.), um comando diferente é executado.

Em termos de fluxogramas, uma construção do tipo Se pode ser encarada como uma bifurcação onde há dois caminhos que podem ser seguidos (Figura 7.2). A execução do algoritmo prosseguirá necessariamente por um deles. Esta escolha é feita em função do resultado da expressão: um dos caminhos é rotulado com (.V.) e será seguido quando a condição for falsa.

A sintaxe da estrutura de decisão do tipo Se é mostrada na Figura 7.2.

Fluxograma



Pseudocódigo

Se <condição>
    Então
        <comando_composto_1>
    Senão
        <comando_composto_2>
Fim_se

Figura 7.2 Sintaxe da estrutura de decisão **Se-Então-Senão-Fim_se**.

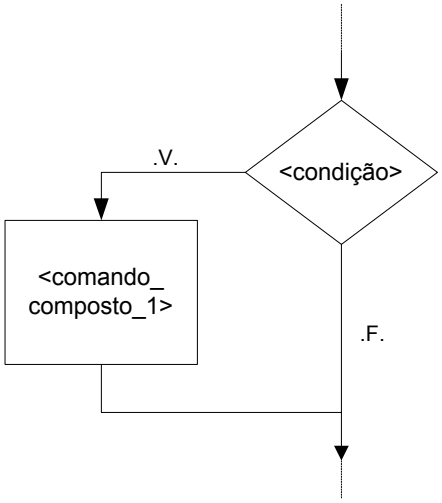
Note-se o aparecimento de novas palavras-reservadas **Se**, **Então**, **Senão** e **Fim_se**.

A semântica desta construção é a seguinte: a condição é avaliada. Se o resultado for verdadeiro, então o **comando_composto_1** é executado. Ao término de sua execução o fluxo do algoritmo prossegue pela instrução seguinte à construção, ou seja, o primeiro comando após o **Fim_se**.

Nos casos em que a condição é avaliada como falsa, o **comando_composto_2** é executado e, ao término do mesmo, o fluxo de execução prossegue pela primeira instrução seguinte ao **Fim_se**.

Há casos particulares e muito comuns desta construção, onde o comando composto 2 é um conjunto vazio de instruções. Neste caso, a porção relativa ao **Senão** pode ser omitida, resumindo a sintaxe da construção à forma mostrada na Figura 7.3.

Fluxograma



Pseudocódigo

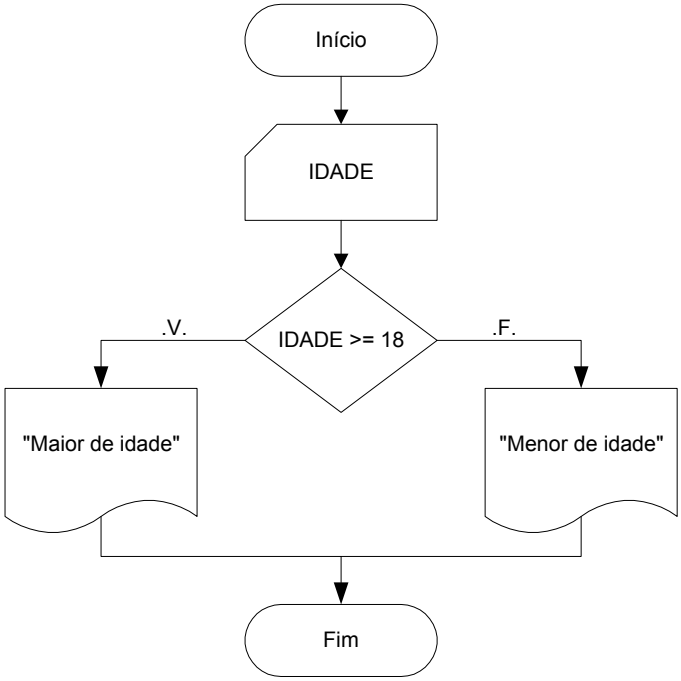
Se <condição>
Então
<comando_composto_1>
Fim_se

Figura 7.3 Sintaxe da estrutura de decisão **Se-Então-Fim_se**.

A semântica desta construção é a seguinte: no caso da condição ser verdadeira, o **Comando_composto_1** é executado e, após seu término, o fluxo de execução prossegue pela próxima instrução após o **Fim_se**. Quando a condição é falsa, o fluxo de execução prossegue normalmente pela primeira instrução após o **Fim_se**.

A Figura 7.4 exemplifica o uso da construção **Se-Então-Senão-Fim_se** num algoritmo para determinar se uma pessoa é maior ou menor de idade.

Fluxograma



Pseudocódigo

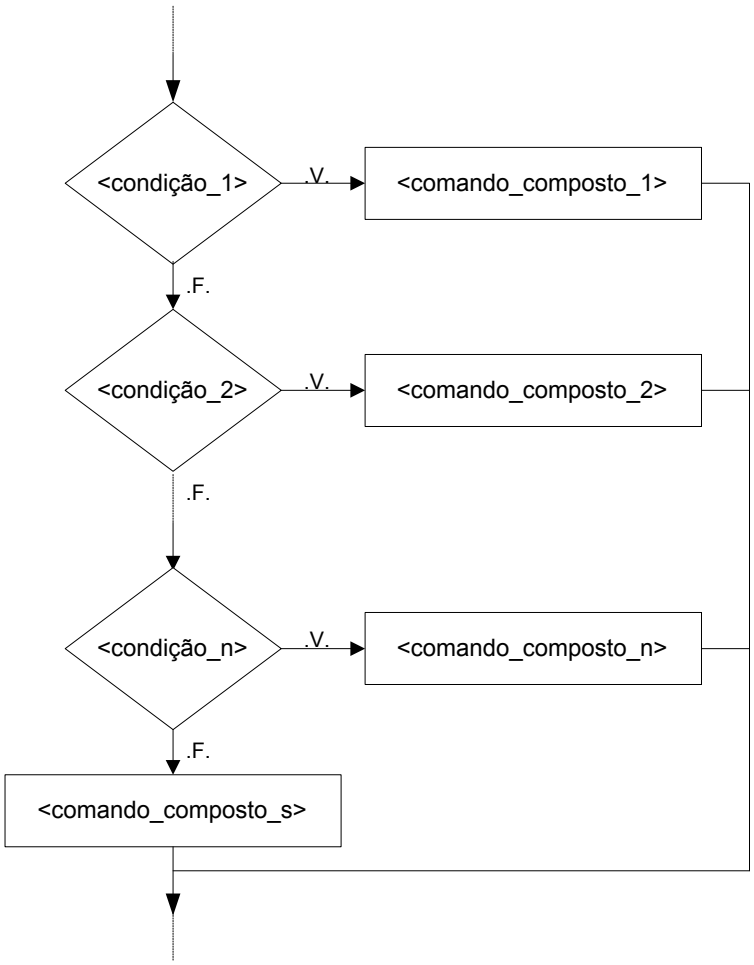
```
Algoritmo Exemplo_7.4
Var  IDADE : inteiro
Início
      Leia IDADE
      Se IDADE >= 18
            Então
               Escreva "Maior de idade"
            Senão
               Escreva "Menor de idade"
      Fim_se
Fim.
```

Figura 7.4 Exemplo de aplicação da estrutura de decisão **Se-Então-Senão-Fim_se**.

7.3.2 Estruturas de Decisão do Tipo Escolha

Este tipo de estrutura é uma generalização da estrutura **Se**, onde somente uma condição era avaliada e dois caminhos podiam ser seguidos. Na estrutura de decisão do tipo **Escolha** pode haver uma ou mais condições a serem testadas e um comando composto diferente associado a cada uma destas.

Fluxograma



Pseudocódigo

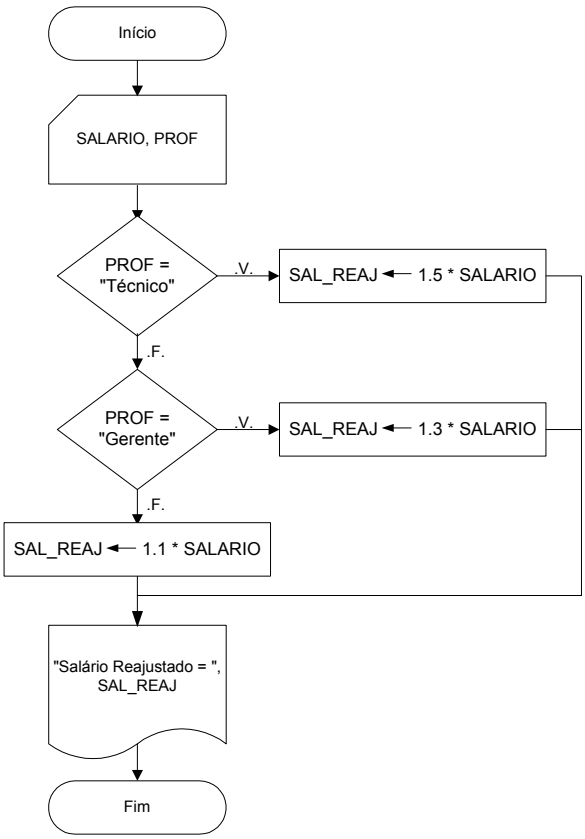
```
Escolha
    Caso <condição_1>
      <comando_composto_1>
    Caso <condição_2>
      <comando_composto_2>
    Caso <condição_n>
      <comando_composto_n>
    Senão
      <comando_composto_s>
Fim_escolha
```

Figura 7.5 Sintaxe usada em fluxograma e pseudocódigo para construção **Escolha**.

Seu funcionamento é o seguinte: ao entrar-se numa construção do tipo **Escolha**, a **condição_1** é testada: se for verdadeira, o **comando_composto_1** é executado e, após seu término, o fluxo de execução prossegue pela primeira instrução após o final da construção (**Fim_escolha**); se a **condição_1** for falsa, a **condição_2** é testada: se esta for verdadeira, o **comando_composto_2** é executado e, ao seu término, a execução prossegue normalmente pela instrução seguinte ao **Fim_escolha**. O mesmo raciocínio é estendido a todas as condições da construção. No caso em que todas as condições são avaliadas como falsas, o **comando_composto_s** (correspondente ao **Senão** da construção) é executado.

Um exemplo de aplicação desta construção é mostrado na Figura 7.6, baseado num algoritmo de reajuste salarial variável em função da profissão.

Fluxograma



Pseudocódigo

```
Algoritmo Exemplo_7.6
Var  SALARIO, SAL_REAJ : real
     PROF : literal[20]
Início
     Leia SALARIO, PROF
     Escolha
          Caso PROF = "Técnico"
            SAL_REAJ ← 1.5 * SALARIO
          Caso PROF = "Gerente"
            SAL_REAJ ← 1.3 * SALARIO
          Senão
            SAL_REAJ ← 1.1 * SALARIO
     Fim_escolha
     Escreva "Salário Reajustado = ", SAL_REAJ
Fim.
```

Figura 7.6 Exemplo de aplicação da construção **Escolha**.

Um caso particular desta construção é aquele em que o **Comando_composto_s** não contém nenhuma instrução. Isto ocorre nas situações em que não se deseja efetuar nenhuma ação quando todas as condições testadas são falsas. Assim, pode-se dispensar o uso do **Senão** na construção, como acontece também na construção **Se**.

7.4 Estruturas de Repetição

São muito comuns as situações em que se deseja repetir um determinado trecho de um programa certo número de vezes. Por exemplo, pode-se citar o caso em que se deseja realizar um mesmo processamento para conjuntos de dados diferentes. Exemplo: processamento de folha de pagamentos de uma empresa em que o mesmo cálculo é efetuado para cada um dos funcionários.

As estruturas de repetição são muitas vezes chamadas de **Laços** ou, também, de **Loops**.

A classificação das estruturas de repetição é feita de acordo com o conhecimento prévio do número de vezes que o conjunto de comandos será executado. Assim, os laços dividem-se em:

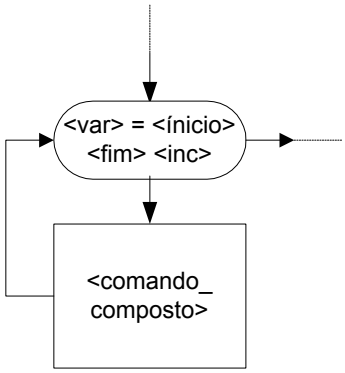
- **laços contados**, quando se conhece previamente quantas vezes o comando composto no interior da construção será executado;

- **laços condicionais**, quando não se conhece de antemão o número de vezes que o conjunto de comandos no interior do laço será repetido, pelo fato de o mesmo estar amarrado a uma condição sujeita à modificação pelas instruções do interior do laço.

7.4.1 Laços Contados

Os **laços contados** são úteis quando se conhece previamente o número de vezes que se deseja executar um determinado conjunto de comandos. Então, este tipo de laço nada mais é que uma estrutura dotada de mecanismos para contar o número de vezes que o corpo do laço (ou seja, o comando composto em seu interior) é executado. A sintaxe usada em pseudocódigos para os laços contados é mostrada na Figura 7.7.

Fluxograma



Pseudocódigo

Para <var> **de** <início> **até** <final> **incr de** <inc> **faça**
    <comando_composto>
Fim_para

Figura 7.7 Sintaxe usada para os **laços contados**.

A semântica do laço contado é a seguinte: no início da execução da construção o valor **<início>** é atribuído à variável **<var>**. A seguir, o valor da variável **<var>** é comparado com o valor **<final>**. Se **<var>** for maior que **<final>**, então o comando composto não é executado e a execução do algoritmo prossegue pelo primeiro comando seguinte ao **Fim_para**. Por outro lado, se o valor de **<var>** for menor ou igual a **<final>**, então o comando composto no interior da construção é executado e, ao final do mesmo, o valor é adicionado à variável **<var>**. Feito isso, retorna-se à comparação entre **<var>** e **<final>** e repete-se o processo até que <var> tenha um valor maior que **<final>**, quando o laço é finalizado e a execução do algoritmo prossegue pela instrução imediatamente seguinte ao **Fim_para**.

Algumas observações interessantes devem ser feitas:

- **<var>** é necessariamente uma variável, uma vez que seu valor é alterado a cada iteração (volta do laço);
- **<início>**, **<fim>** e **<inc>** podem ser constantes ou variáveis. No segundo caso (variáveis), algumas linguagens de programação proíbem que seus valores sejam modificados durante a execução do laço;
- **<inc>** é o valor que é adicionado à variável **<var>** ao final de cada iteração do laço. Há linguagens de programação que permitem que lhe seja atribuído um valor negativo, de modo que o valor da variável **<var>** diminui a cada iteração. Neste caso, deve-se atentar à necessidade de inversão do sinal da comparação (de > para <) que é feito a cada volta do laço, para seu correto funcionamento. Contudo, este texto assumirá que **<inc>** possui um valor sempre positivo, o que não causa perda de generalidade, já que um resultado análogo pode ser obtido com o uso das construções enquanto e repita, apresentadas a seguir;
- na grande maioria dos casos **<inc>** tem o valor 1 (incremento unitário). Portanto, admite-se a omissão do trecho **incr de <inc>** da sintaxe do comando **Para** e, quando isto ocorre, assume-se um incremento de 1.

7.4.2 Laços Condicionais

Laços condicionais são aqueles cujo conjunto de comandos em seu interior é executado até que uma determinada condição seja satisfeita. Ao contrário do que acontece nos laços contados, nos laços condicionais não se sabe de antemão quantas vezes o corpo do laço será executado.

As construções que implementam laços condicionais mais comuns nas linguagens de programação modernas são:

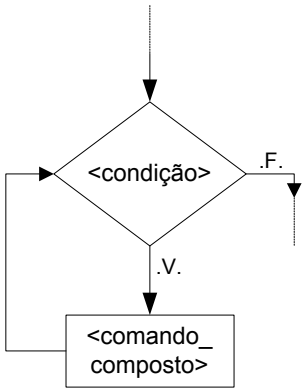
- Enquanto
- Repita

7.4.2.1 Construção Enquanto

Sua semântica é a seguinte: ao início da construção **Enquanto** a condição é testada. Se seu resultado for falso, então o comando composto no seu interior não é executado e a execução prossegue normalmente pela instrução seguinte ao **Fim_enquanto**.

Se a condição for verdadeira o comando composto é executado e ao seu término retorna-se ao teste da condição. Assim, o processo acima será repetido enquanto a condição testada for verdadeira. Quando esta for falsa, o fluxo de execução prosseguirá normalmente pelas instruções posteriores ao **Fim_enquanto**.

Fluxograma



Pseudocódigo

Enquanto <condição> **faça**
    <comando_composto>
Fim_enquanto

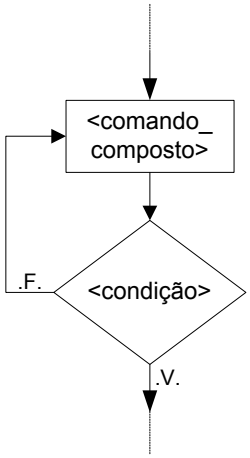
Figura 7.8 Sintaxe da construção **Enquanto** para laços condicionais.

Uma vez dentro do corpo do laço, a execução somente abandonará o mesmo quando a condição for falsa. O usuário deste tipo de construção deve estar atento à necessidade de que em algum momento a condição deverá ser avaliada como falsa. Caso contrário, o programa permanecerá indefinidamente no interior do laço, o que é conhecido como **laço infinito**.

7.4.2.2 Construção Repita

Seu funcionamento é bastante parecido ao da construção **Enquanto**. O comando é executado uma vez. A seguir, a condição é testada: se ela for falsa, o comando composto é executado novamente e este processo é repetido até que a condição seja verdadeira, quando então a execução prossegue pelo comando imediatamente seguinte ao final da construção.

Fluxograma



Pseudocódigo

Repita
<comando_composto>
Até que <condição>

Figura 7.9 Sintaxe da construção **Repita** para laços condicionais.

Esta construção difere da construção **Enquanto** pelo fato de o comando composto ser executado **uma** ou mais vezes (pelo menos uma vez), ao passo que na construção **Repita** o comando composto é executado **zero** ou mais vezes (possivelmente nenhuma). Isto acontece porque na construção **Repita** o teste da condição é feito ao final da construção, ao contrário do que acontece na construção **Enquanto**, onde o teste da condição é feito no início da mesma.

7.5 Aninhamentos

Um **aninhamento** ou **embutimento** é o fato de se ter qualquer um dos tipos de construção apresentados anteriormente dentro do conjunto de comandos (comando composto) de uma outra construção.

Em qualquer tipo de embutimento é necessário que a construção interna esteja completamente embutida na construção externa.

A Figura 7.10 ilustra aninhamentos válidos e inválidos.

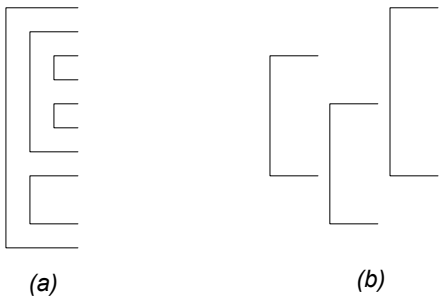


Figura 7.10 Exemplos de aninhamentos *(a)* válidos e *(b)* inválidos.

7.6 Síntese

As estruturas de controle do fluxo de execução são essenciais para que se possa alterar a seqüência de execução dos comandos de um programa em função dos dados do mesmo.

Um **comando composto** é um conjunto de zero ou mais comandos simples, sejam eles instruções primitivas ou construções como as estudadas neste capítulo.

Uma **estrutura seqüencial** é aquela em que os comandos vão sendo executados numa seqüência pré-estabelecida, um após o outro.

As **estruturas de decisão** permitem escolher qual o caminho a ser seguido num algoritmo em função de uma ou mais condições. A construção **Se** utiliza apenas uma condição, ao passo que a construção **Escolha** utiliza uma ou mais condições.

As **estruturas de repetição** são usadas quando se deseja repetir um trecho de um algoritmo (comando composto). Quando o número de vezes que o trecho será repetido é conhecido diz-se que o laço é do tipo **contado** (construção **Para**). Quando este número não é conhecido, mas é função de uma determinada condição, então têm-se os laços **condicionais** (construção **Enquanto** e **Repita**).

As construções **Repita** e **Enquanto** diferem uma da outra pelo fato de a primeira efetuar o teste da condição no final da construção e, portanto, executar o comando composto ao menos uma vez. Por outro lado, a construção **Enquanto** efetua o teste da condição em seu início e executa o comando composto zero ou mais vezes.

Os **aninhamentos** podem usar qualquer uma das construções apresentadas neste capítulo; desde que atendam a uma regra única: a construção mais interna deve estar inteiramente contida na construção imediatamente mais externa.