Estruturas de Controle
(2ª Parte)

Estruturas de Repetição

Uma estrutura de **repetição** permite que um mesmo trecho de um algoritmo seja executado quantas vezes seja desejado. Por exemplo, suponha um programa que leia as notas de um aluno e calcule sua média. Para fazer o cálculo das médias de 50 alunos, teríamos que executar esse programa 50 vezes. Outra solução seria escrever outro programa, repetindo o mesmo trecho de código 50 vezes. Embora ambas as soluções sejam simples, nenhuma delas é viável. Em situações como essa, em que o mesmo trecho de código terá de ser executado diversas vezes, devemos utilizar uma estrutura de repetição.

Os trechos do algoritmo que são repetidos são chamados de laços de repetição ou *loops*. Ganham esse nome por lembrarem uma execução finita em círculos, que depois segue seu curso normal.

Quando criamos um laço de repetição, podemos não saber de antemão quantas vezes esse laço se repetirá, mas precisamos garantir que as repetições cessarão em algum momento. Em outras palavras, o número de repetições pode ser indeterminado, porém necessariamente finito. Caso contrário, o programa nunca terminará sua execução.

O controle do número de repetições sempre é feito por uma **condição** (geralmente chamada de "condição de parada") ou por uma variável de controle (quando sabemos o número exato de repetições). Veremos três tipos de estruturas de repetição: repetição com teste no início, repetição com teste no fim e repetição com variável de controle. Veremos também dois conceitos importantes geralmente usados em laços de repetição: o contador e o acumulador.

Repetição com Teste de Condição no Início (*Enquanto*)

Consiste em uma estrutura de controle do fluxo de execução que permite repetir diversas vezes um mesmo trecho do algoritmo, porém sempre verificando antes de cada execução se é "permitido" executar tal trecho.

Para realizar a repetição com teste no início utilizamos a estrutura **enquanto**, que permite que uma ação ou um bloco de ações seja repetido enquanto uma determinada **condição** for verdadeira. O modelo genérico desse tipo de repetição é o seguinte:

```
enquanto <condição> faça                ou        enquanto <condição> faça
início                                                    ação;
        ação 1;
        ação 2;
        ...
        ação n;
fim;
```

Enquanto a condição for verdadeira, o bloco de ações (ação 1; ação 2; ... ação n;) será executado. Quando a condição for falsa, o comando de repetição é abandonado e fluxo de execução é retomado a partir do primeiro comando que segue o término do **enquanto**. Se a condição for falsa logo no primeiro teste, as ações não serão executadas nenhuma vez, o que representa a característica principal desse modelo de repetição.

No modelo genérico mostrado acima, o bloco de ações pertencente ao comando **enquanto** está delimitado por **início** e **fim**. Assim como nos comandos de seleção, quando *uma única ação* faz parte do comando **enquanto**, não é necessário o uso de **início** e **fim**, conforme exemplificado.

Contador

Voltemos ao exemplo em que se quer ler as notas de 50 alunos e calcular suas respectivas médias. Nesse caso, queremos repetir a seqüência de ações – ler notas; calcular média; imprimir média – 50 vezes. Tal algoritmo pode ser feito usando-se uma estrutura de repetição com teste no início. Nesse caso, a condição para a repetição seria que "*a quantidade de médias calculadas fosse menor ou igual a 50*". Porém, o que indica quantas vezes a média foi calculada? A estrutura **enquanto** não oferece esse recurso. Portanto, devemos estabelecer uma forma de contagem, o que pode ser feito com a ajuda de um **contador**. Usando um contador teremos o controle de quantas médias já foram calculadas.

Um **contador** é representado por uma variável com um dado valor inicial e que é incrementado a cada repetição. Incrementar é o mesmo que somar um valor constante (geralmente 1). Veja as linhas de exemplo abaixo.

1. i: inteiro; *{declara do contador i}*
2. i ← 0; *{inicializa do contador i}*
3. i ← i + 1; *{incrementa o contador i de 1}*

Na linha 1 vemos a declaração da variável *i*, que será usada como contador. Na linha 2, a variável *i* recebe ser valor inicial, por isso dizemos que o contador foi *inicializado*. O processo de contagem ocorre na linha 3, por meio da expressão aritmética que obtém o valor da variável *i* e adiciona 1 a esse valor, armazenando o resultado na própria variável *i*. Se repetirmos esse comando (linha 3) várias vezes, perceberemos que a variável vai aumentando gradativamente de valor (de 1 em 1), simulando uma contagem de execuções.

Voltando ao exemplo das 50 médias, podemos usar um contador para armazenar o número de médias calculadas e usar esse contador para declarar a condição de repetição. Abaixo é mostrado um algoritmo que lê as notas de 50 alunos, calcula e mostra suas respectivas médias, exemplificando o uso do comando **enquanto** e de um **contador** que chamado de *nmedias*.

```
programa media50;
var
   n1, n2, nmedias: inteiro;  {n1 e n2 são as notas; nmedias é o contador}
   m : real;                  {m é a média}
inicio
   nmedias ← 0;                        {inicialização do contador}
   enquanto (nmedias < 50) faça        {teste da condição de parada da repetição}
   inicio
      leia(n1, n2);
      m ← (n1+n2)/2;
      escreva(m);
      nmedias ← nmedias + 1;           {incremento do contador }
   fim;
fim.
```

Observe que o contador *nmedias* foi inicializado com zero e por isso a condição do comando enquanto é (*nmedias* < 50). Se o contador fosse inicializado com 1, a condição teria que ser (*nmedias* <= 50) para que o laço se repetisse 50 vezes.

Acumulador

Usando ainda o exemplo do algoritmo para o cálculo das médias de 50 alunos, suponha que também quiséssemos calcular a média geral da turma (a média aritmética das médias dos 50 alunos). Isso poderia ser feito por meio da expressão aritmética:

$$(m1 + m2 + m3 + ... + m49 + m50)/50$$

A expressão utilizada é gigantesca e, portanto, inviável (sem contar que teríamos que alocar 50 variáveis diferentes só para armazenar as médias). Nesse caso, podemos utilizar as vantagens da estrutura de repetição, fazendo um laço que a cada execução acumule em uma variável, conhecida conceitualmente como **acumulador**, o somatório das médias de cada aluno. Após o término da repetição, teríamos a soma de todas as médias na variável de acumulação, restando apenas dividi-la pela quantidade de médias somadas, ou seja, 50.

O processo de acumulação é muito similar ao processo de contagem (contador). Assim como o contador, o acumulador precisa ser declarado e inicializado. A única diferença entre o contador e o acumulador é que na acumulação o valor adicionado pode variar, enquanto na contagem o valor adicionado é constante. Para ilustrar o uso do **acumulador**, abaixo é mostrado o mesmo algoritmo apresentado anteriormente para o cálculo das 50 médias, porém agora incluindo o cálculo da média geral dos 50 alunos.

```
1.  programa media50geral;
2.  var
3.     n1, n2, nmedias: inteiro;
4.     m, mgeral, acum: real;        {m é a média individual; mgeral é a média geral}
                                     { acum é o acumulador de médias }
5.  inicio
6.     nmedias ← 0;                  {inicialização do contador}
7.     acum ← 0;                     {inicialização do acumulador}
8.     enquanto (nmedias < 50) faça  {teste da condição de parada da repetição}
9.     inicio
10.       leia(n1, n2);
11.       m ← (n1+n2)/2;
12.       escreva(m);
13.       acum ← acum + m;           {acumulo do valor de m em acum }
14.       nmedias ← nmedias + 1;     {incremento do contador }
15.    fim;
16.    mgeral ← acum/50;             {cálculo da média geral}
17.    escreva(mgeral);
18. fim.
```

Na linha 3 e 4 vemos a declaração das variáveis, incluindo a variável que será utilizada como acumulador (*acum*). Na linha 7 é feita a inicialização do acumulador com o valor zero. O processo de acumulação ocorre na linha 13, por meio da expressão aritmética que obtém o valor da variável *acum* e adiciona a ele o valor de *m* (média), armazenando o resultado na própria variável *acum*. A cada passo da repetição, o valor da média calculada naquele passo é adicionado ao somatório das médias calculadas nas repetições anteriores. Assim, podemos dizer que as médias são *acumuladas* em *acum*. Na linha 16 é calculada a média geral por meio da expressão aritmética que divide o somatório das médias armazenado em *acum* por 50 (número de médias acumuladas).

Repetição com Quantidade Pré-determinada e Indeterminada

O algoritmo acima utiliza o pré-conhecimento da quantidade de alunos da turma da qual se desejava a média geral, o que permitiu construir um laço de repetição com quantidade *pré-determinada* de execuções (50 vezes). Entretanto, se não soubéssemos quantos eram os alunos, como faríamos para controlar o laço de repetição? Precisaríamos de um laço que fosse executado por uma quantidade *indeterminada* de vezes. Assim, teríamos que encontrar outro critério de parada, que possibilitasse que o laço fosse finalizado após as últimas notas terem sido informadas. Isso pode ser feito utilizando um valor predefinido como **finalizador**, a ser informado após a última média.

Para aplicar tal conceito é mostrado abaixo um algoritmo que lê um número indeterminado de médias de alunos e calcula a média geral da turma, usando como finalizador o valor -1. Dessa forma, quando o -1 for lido, o laço será encerrado sem ter seu valor computado no acumulador para o cálculo da média geral.

```
1.  programa mediageral;
2.  var
3.     media, mgeral, acum: real;    {media é a variável de leitura das médias individuais}
                                     { mgeral é a média geral; acum é o acumulador de médias }
4.     nmedias: inteiro;             {nmedias é o contador de médias lidas}
5.  inicio
6.     nmedias ← 0;                  {inicialização do contador}
7.     acum ← 0;                     {inicialização do acumulador}
8.     leia(media);                  {1ª leitura para a inicialização da variável}
9.     enquanto (media <> -1) faça   {teste da condição de parada da repetição}
10.    inicio
11.       acum ← acum + media;       {acumulo do valor de média em acum }
12.       nmedias ← nmedias + 1;     {incremento do contador }
13.       leia(media);               {nova leitura da média}
14.    fim;
15.    Se (nmedias > 0) então        {verifica se pelo menos uma média válida foi lida}
16.    inicio
17.       mgeral ← acum/nmedias;     {cálculo da média geral}
18.       escreva(mgeral);
19.    fim;
20. fim.
```

Perceba que antes de iniciar o laço de repetição, uma média precisa ser lida para que a condição do comando enquanto possa ser avaliada corretamente. Essa leitura para a inicialização da variável média é feita na linha 8. Uma vez dentro do laço de repetição, a média é acumulada em *acum*, o contador é incrementado de 1 e a leitura de uma nova média é feita, na linha 13. Dessa forma, a cada repetição uma nova média lida. No momento em que for digitado o valor -1 para a variável média, o laço é interrompido e o processamento prossegue a partir da linha 15. Se pelo menos uma média válida foi lida (*nmedias* > 0), então a média geral é calculada e impressa.

Repetição com Teste de Condição no Final (*Repita*)

Para realizar a repetição com teste no final utilizamos a estrutura **repita**, que permite que uma ação ou um bloco de ações seja repetido **até** que uma determinada condição seja verdadeira. O modelo genérico desse tipo de repetição é mostrado abaixo:

```
repita                                  ou      repita
      ação 1;                                       ação
      ação 2;                           até <condição>;
      ...
      ação n;
até <condição>;
```

A diferença da estrutura **repita** para a estrutura **enquanto** é que o **repita** *sempre* executará o bloco de ações *pelo menos uma vez* e no **enquanto** o bloco de ações pode ser executado *nenhuma vez*. Isso ocorre porque a verificação da *condição* é feita após a execução do bloco de ações, o que representa a característica principal desse modelo de repetição.

Abaixo é mostrado um algoritmo que lê a média anual de 50 alunos e calcula a média geral da turma, utilizando a estrutura de repetição com teste de condição no final.

```
1.  programa mediageral;
2.  var
3.     media, mgeral, acum: real;    {media é a variável de leitura das médias individuais}
                                     {mgeral é a média geral; acum é o acumulador de médias }
4.     nmedias: inteiro;             {nmedias é o contador de médias lidas}
```

```
5. inicio
6.     nmedias ← 0;                    {inicialização do contador}
7.     acum ← 0;                       {inicialização do acumulador}
8.     repita
9.         leia(media);                { leitura da média individual}
10.        acum ← acum + media;        {acumulo do valor de média  em acum }
11.        nmedias ← nmedias + 1;      {incremento do contador }
12.    até (nmedias >= 50);            {teste da condição de parada}
13.    mgeral ← acum/nmedias;          {cálculo da média geral}
14.    escreva(mgeral);
15. fim.
```

Note que na utilização da estrutura **repita**, a condição de parada do laço de repetição é a *negação* da condição utilizada na estrutura **enquanto**. O mesmo se aplicaria se, em vez de uma condição com quantidade pré-determinada, fosse utilizada uma condição com quantidade indeterminada (uso de um valor finalizador).

Repetição com Variável de Controle (*Para*)

Nas estruturas de repetição vistas até aqui, ocorrem casos em que se torna difícil determinar o número de vezes em que o laço de repetição será executado. Sabemos que ele será executado *enquanto* uma condição for satisfeita (**enquanto**) ou *até* que uma condição seja satisfeita (**repita**). A estrutura **para** é diferente, já que sempre executa o laço de repetição um número pré-determinado de vezes, pois ela não prevê uma condição e sim limites fixos de variação.

O modelo genérico para a estrutura de repetição **para** é o seguinte:

para v ← v_i **até** v_f **faça**
início
    ação 1;
    ação 2;
    ...
    ação n;
fim;

ou

para v ← v_i **até** v_f **faça**
    ação;

onde *v* é a variável de controle; v_i é o valor inicial da variável *v*; e v_f é o valor final da variável *v*, ou seja, o valor limite até o qual ela será incrementada. Dessa forma, a cada repetição da estrutura **para**, a variável *v* é incrementada de 1, a partir de seu valor inicial v_i, até atingir o valor final v_f.

A estrutura **para** possibilita o uso de um laço com contador de forma compacta, em que sempre temos uma inicialização (v_i) da variável de controle (*v*), um teste para verificar se a variável atingiu o limite (v_f) e um incremento de 1 na variável de controle após cada execução do bloco de repetição. Uma exemplificação simples é do uso da estrutura **para** é mostrado abaixo.

para i ← 1 **até** 10 **faça**
    escreva(i);

A execução do exemplo acima causaria a repetição do comando escreva(i) por 10 vezes, de modo que a saída na tela seria o valor da variável i em cada repetição, ou seja, *1 2 3 4 5 6 7 8 9 10*.

Abaixo é mostrado um algoritmo que lê a média anual de 50 alunos e calcula a média geral da turma, utilizando a estrutura de repetição com variável de controle.

```
1. programa mediageral;
2. var
3.     media, mgeral, acum: real;    {media é a variável de leitura das médias individuais}
                                     {mgeral é a média geral; acum é o acumulador de médias }
4.     nmedias: inteiro;             {nmedias é a variável de controle, i.e., o contador de médias lidas}
```

```
5. inicio
6.     acum ← 0;                        {inicialização do acumulador}
7.     para nmedias ← 1 até 50 faça
8.     inicio
9.         leia(media);                 { leitura da média individual}
10.        acum ← acum + media;         {acumulo do valor de média  em acum }
11.    fim;
12.    mgeral ← acum/nmedias;           {cálculo da média geral}
13.    escreva(mgeral);
14. fim.
```

Uma variação da estrutura para pode ser usada para *decrementar* a variável de controle. O decremento também é de 1 e, assim como no **para** convencional (com incremento), é feito a cada repetição do laço. Para sinalizar que a variável de controle deve ser decrementada em vez de incrementada, vamos trocar o **até** por **descer_até**, conforme exemplificado abaixo.

para i ← 10 **descer_até** 1 **faça**
    escreva(i);

A execução do exemplo acima também causaria a repetição do comando escreva(i) por 10 vezes, entretanto, a saída na tela seria *10 9 8 7 6 5 4 3 2 1*.