Lógica de programação

Algoritmos II

Prof. Kleber Petry

Instrução break

- Considere um jogo de dados no qual o jogador ganha quando a soma dos números obtidos em lançamentos consecutivos de um dado ultrapassar um determinado valor. Antes de começar o -jogo, é necessário definir a quantidade máxima de lançamentos e o valor que deve ser ultrapassado para obter a vitória. Eventualmente, se o valor desejado for ultrapassado antes do último lançamento, não é necessário continuar jogando o dado pois a vitória já está garantida. Podemos implementar um programa de computador para simular a execução desse jogo. Nesse programa, podemos utilizar a instrução break para interromper os lançamentos se o valor desejado for ultrapassado.
- A instrução break não é uma instrução de repetição, mas está fortemente relacionada às instruções while e for. Ela é utilizada para forçar a parada de um laço.

Instrução continue

- Nesse caso, podemos utilizar a instrução continue. Essa instrução permite que, durante a execução de um laço, uma determinada iteração seja abortada fazendo com que o fluxo de execução continue para a próxima iteração.

Comando de repetição

para facilitar a programação, existem comandos de repetição que permitem que certos comandos sejam executados mais de uma vez. Veremos, os três comandos de repetição disponíveis;

Enquanto.. Faça (while)

Repetir...até (repeat)

Para..te..faça (for)

Instruções de Repetição

- Considere um programa que gera bilhetes de loteria. O número do primeiro bilhete é 1000, do segundo 1001, do terceiro 1002 e assim por diante até o último bilhete numerado com 9999. Para esse tipo de tarefa, podemos utilizar as instruções de repetição oferecidas pelas linguagens de programação.
- Basicamente, as instruções de decisão permitem que um determinado trecho de código seja executado ou não. Em algumas situações, é necessário repetir a execução de um determinado trecho de código. Nessas situações, devemos utilizar as instruções de repetição.

Comando enquanto..faça WHILE

O comando repetição **enquanto..faça('while...do')** enquanto a condição for verdadeira, o comando ou bloco de comandos dentro do enquanto será executado.

Quando a condição for falsa, será executado o próximo comando depois do bloco de comandos do **enquanto.**

Enquanto “While”

EX:

Considere um algoritmo que lê duas notas e calcula e escreve a média de 30 alunos. Se a média for maior ou igual a 7, escrever a mensagem **Aprovado**. Caso a média for menor do que 7, escrever **Reprovado**. A leitura das notas e o cálculo das médias deverão ser executados 30 vezes.

While Definido

1	Algoritmo calcula_media
2	variaveis nota1,nota2,media:real
3	cont:inteiro
4	inicio
5	cont<-1
6	enquanto cont<=30 faça
7	inicio
8	escreva (’informe as duas notas’)
9	ler(nota1,nota2)
10	media<-(nota1+nota2)/2
11	escreva(’media=’, media)
12	se media>=7 entao
13	escreva(’aprovado’)
14	senao
15	escreva(’reprovado’)
16	cont<-cont+1
17	fim
18	fim

While indefinido

```
1  Algoritmo calcula_media
2  variaveis nota1,nota2,media:real
3        cont:inteiro
4  inicio
5       escreva('deseja calcular média? S-sim N-não
6       ler(opcao)
7       enquanto opcao='S' faça
8  inicio
9          escreva ('informe as duas notas')
10         ler(nota1,nota2)
11         media<-(nota1+nota2)/2
12         escreva('media=', media)
13         se media>=7 entao
14         escreva('aprovado')
15         senao
16         escreva('reprovado')
17         ler(opcao)
18         fim
19 fim
```

Prof. Kleber Petry

Instrução while

- A instrução while indica o início de um laço e recebe como parâmetro uma condição. Essa condição é chamada de condição de parada, pois quando ela for falsa, o laço é interrompido. A estrutura
- ou sintaxe da instrução while é a seguinte:

```
1 while(condição de parada){
2   // comando 1
3   // comando 2
4   // comando 3
5 }
```

- Se traduzirmos para o português a instrução while como enquanto, fica mais fácil entender o seu funcionamento. O código acima poderia ser lido da seguinte forma: “Enquanto a condição de parada for verdadeira, execute comando 1, comando 2 e comando 3.”

Repetir até “Repeat Until”

O comando **repetir(‘repeat’)** não necessita de **inicio** e **fim** para delimitar o bloco de comandos. O início é delimitado pela palavra **repetir**, e o fim pela palavra **até**. Diferente do comando **enquanto(‘while’)** que testa a condição no inicio, o comando **repetir** tem uma entrada vazia, pois o teste é feito somente no final da execução. Isso significa que o bloco de comando **repetir(‘repeat’)** será executado no mínimo uma vez.

Repeat definido

```
1  Algoritmo calcula_media
2  variaveis nota1,nota2,media:real
3        cont:inteiro
4  inicio
5       escreva('deseja calcular média? S-sim N-não
6       ler(opcao)
7       enquanto opcao='S' faça
8  inicio
9          escreva ('informe as duas notas')
10         ler(nota1,nota2)
11         media<-(nota1+nota2)/2
12         escreva('media=', media)
13         se media>=7 entao
14         escreva('aprovado')
15         senao
16         escreva('reprovado')
17         ler(opcao)
18         fim
19 fim
```

Prof. Kleber Petry

Repeat indefinido

```
1  Algoritmo calcula_media
2  variaveis nota1,nota2, media media_geral:real
3          cont:inteiro
4           opcao:caracter
5  inicio
6         cont<-1
7         media_geral<-0
8       repetir
9          escreva (’informe as duas notas’)
10         ler(nota1,nota2)
11         media<-(nota1+nota2)/2
12         escreva(’media=’, media)
13         se media>=7 entao
14         escreva(’aprovado’)
15         senao escreva(’reprovado’)
16         media_geral<-media_geral+media
17        cont<-cont+1
18        escreva(’deseja calcular media? S-sim N-nao’)
19        ler(opcao)
20      ate opcao = ‘N’
21       escreva(’media geral da turma:’, media_geral/cont)
22 fim
```

Prof. Kleber Petry

Instrução for

- A instrução for é uma outra instrução de repetição e tem a mesma finalidade da instrução while.
- Na maioria dos casos, podemos resolver questões que envolvem repetições com while ou for. A diferença é que, geralmente, utilizamos a instrução for nos casos em que precisamos de um contador em nossa condição de parada. Para ficar mais claro, veja a estrutura ou sintaxe da instrução for:

Comando para..ate..faça

O comando de repetição **para..ate..faça** é um comando de repetição contada, ou seja, em que o numero de repetições é conhecido. A inicialização da variável de controle é feito no próprio comando(depois da palavra **para**), e o incremento é realizado automaticamente pelo comando **para** depois que o comando ou bloco de comandos do **para** é executado.

Comando FOR

```
1 for(inicialização; condição de parada; atualização){
2   // comandos
3 }
```

- No lugar da inicialização, devemos inserir os comandos que serão executados antes do início do laço. No lugar da atualização, devemos inserir os comandos que serão executadas ao final de cada iteração (repetição)

Instruções de Repetição Encadeadas

- Considere o programa de computador que gera os ingressos das apresentações de um determinado teatro. Esse teatro foi dividido em 4 setores com 200 cadeiras cada. Os ingressos devem conter O número do setor e o número da cadeira. Podemos utilizar laços encadeados para implementar esse programa.

```java
int i = 1;
while(i <= 4) {
  int j = 1;
  while(j <= 200) {
    System.out.println("SETOR: " + i + " CADEIRA: " + j);
    j++;
  }
  i++;
}
```

```java
int i = 1;
while(i <= 4) {
  int j = 1;
  for(int j = 1; j <= 200; j++) {
    System.out.println("SETOR: " + i + " CADEIRA: " + j);
  }
  i++;
}
```

```java
for(int i = 1; i <= 4; i++) {
  int j = 1;
  while(j <= 200) {
    System.out.println("SETOR: " + i + " CADEIRA: " + j);
    j++;
  }
}
```

Tipos de dados

TIPO	DESCRIÇÃO	RANGE
CHAR	Texto, utilizado para armazenar cadeia de caracteres.	
CHAR NOCASE	Texto, utilizado para armazenar cadeia de caracteres, porem este não difere uma letra maiúscula de uma minúscula no filtro de uma SELECT.	
SHORT	Um pequeno campo capaz de armazenar números inteiros.	-32768 até 32767
INT	Utilizado para armazenar números inteiros.	-2147483648 até 2147483647
LONG	Quando o número for inteiro, porém ele não pode ser armazenado em um campo INT a única opção é o tipo LONG.	-9223372036854775808 até 9223372036854775807
FLOAT	Quando o programador decide armazenar um valor numérico não inteiro, a opção FLOAT deve ser utilizada se o número for inferior ao range aceito por este tipo.	±1.4E-45 até ±3.4028235E+38
DOUBLE	Se FLOAT não for capaz de armazenar o valor então DOUBLE deve ser adotado, porém este tipo consome muito espaço e este deve ser utilizado com cautela.	±4.9E-324 até ±1.7976931348623157E+308
DATE	Utilizado para armazenar data.	1920-01-01 até 2036-12-31
DATETIME	Utilizado para armazenar data e hora.	1920-01-01 00:00:00 até 2036-12-31 23:59:59

Tipos de dados compostos

Imagine um programa que recebe duas notas de 30 alunos e que seja necessário armazenar essas 60 notas em variáveis distintas. Como seria?

As variáveis poderiam ser declaradas da seguinte forma:

Variaveis: nota1_aluno1, nota2_aluno1, nota1_aluno2...

Seria possível fazer uma estrutura de repetição para ler essas notas?

Dados compostos

Os tipos de dados compostos nos auxiliam a agrupar e a organizar nossas informações dentro de um algoritmo.

Esses tipos de dados compostos são formados a partir de ooutros tipos de dados como:

- Vetor
- Matriz
- Registro

Vetor

O vetor uma estrutura de dado unidimensional, composta e homogênea.

- Unidimensional significa que o vetor é dividido em posições e que os valores são armazenados em cada uma das posições do vetor.
- Composta significa que o vetor é formatado por outros tipos de dados, ou seja, que podemos criar um vetor para armazenar números do tipo inteiro.
- Homogênea determina que todos os elementos do vetor são do mesmo tipo.

Vetor

<variável>:vetor[<índice_inicial>..<índice_final>] de <tipo>

V:vetor[1..5] de inteiro

Notas:vetor[1..15]de real

Vetor

Leitura de um vetor de cinco posições

Escreva(‘informe 5 valores’)

Ler(v) ←ERRADO

Se v é uma variável do tipo vetor, então o comando **ler(v)** está errado, pois devemos dizer a posição para a qual o valor deve ser lido.

Forma correta

Ler(v[1],v[2],v[3],v[4],v[5])

Vetor

Mas se ele tiver cem posições?

A forma correta para ler um vetor é utilizando um laço de repetição para alterar o índice do vetor a cada iteração, assim, inserir um valor em cada um das posições do vetor.

Isso seria escrito da seguinte forma

Para i<-1 ate 5 faça

Ler(v[i])

Vetor

Utilizando um **enquanto..faça**(while)

```
Escrever('informe 5 valores')
i<-1
Enquanto i<=5 faça
Inicio
        ler(v[i])
        i<-i+1
fim
```

Vetor

Para imprimir na tela os dados de um vetor você também deve usar um laço de repetição:

Escreva(‘valores do vetor’)

Para i<-1 ate 5 faça

Escreva(v[i])

Exemplo

Suponha um algoritmo que lê 50 valores e, no final, apresenta valores pares.

Para resolver esse algoritmo precisamos de um vetor, pois temos que armazenar os 50 valores(um em cada posição do vetor) e, no final, apresentar na tela apenas os valores pares. Se não fosse utilizado um vetor de 50 posições, teríamos que ter 50 variáveis, um para cada valor, o que inviabilizaria a criação do algoritmo

O rascunho dele seria assim:

Ler os 50 valores para o vetor

Percorrer o vetor e testar se o valor desta posição é par

Se o valor dor par, ele deve ser apresentado na tela

Vetor exemplo

1	Algoritmo valores_pares
2	variaveis v:vetor[1..50] de inteiro i:inteiro
3	inicio
4	escreva(’informe 50 valores’)
5	para i<-1 ate 50 faça
6	ler(v[i])
7	escreva(’valores pares do vetor’)
8	para i<-1 ate 50 faça
9	se v[i] mod 2=0
10	entao escreva(v[i])
11	fim

1	Algoritmo valores_pares
2	variaveis v:vetor[1..10] de inteiro i:inteiro
3	inicio
4	escreva(’informe 10 valores’)
5	pos<-0
6	para i<-1 ate 10 faça
7	inicio
8	ler(valor)
9	se valor mod 2<>0
10	entao inicio
11	pos<-pos+1
12	v[pos]<-valor
13	fim
14	fim
15	escreva(’valores ímpares do vetor’)
16	para i<-1 ate pos faça
17	escreva(v[i])
18	fim

Prof. Kleber Petry

Matriz

A matriz é uma estrutura semelhante ao vetor, ou seja, composta e homogênea(formada por outros tipos de dados e tendo todos os valores do mesmo tipo), mas que pode ter mais de uma dimensão.

Leitura da matriz

Escrever('informe uma matriz 3x3')

Ler(m) ← ERRADO

Se **m** é uma matriz, então não podemos utilizar **ler(m)**, pois devemos informar a posição para a qual o valor deve ser lido(linha e coluna)

Matriz

Leitura correta seria:

Ler(m[1,1],m[1,2],m[1,3]

Embora este comando este correto, o mais adequado seria utilizar um laço de repetição.

Assim teríamos:

Para i<-1 ate 3 faça

Para j<-1 ate 3 faça

Ler(m[i,j])

1	Algoritmo maior_valor
2	variaveis m:matriz[1..4, 1..4] de inteiro i,j,maior,coluna,linha:inteiro
3	inicio
4	escreva('informe uma matriz 4x4')
5	para i<-1 ate 4 faça
6	paraj<-1 ate 4 faça
7	ler(m[i,j])
8	maior<-m[1,1]
9	linha<-1
10	coluna<-1
11	para i <- 1 ate 4 faça
12	para j <-1 ate 4 faça
13	se m[i,j]>maior
14	entao inicio
15	maior<-m[i,j]
16	linha<-i
17	coluna<-1
18	fim
19	escreva('Maior valor:',maior)
20	escreva('linha:',linha,' Coluna:',coluna)
21	fim

Prof. Kleber Petry

Registro

Um registro é formado de um ou mais campos, e cada um dos campos possui um nome e um tipo. Os campos do registro podem ser de tipos diferentes.

Utilizando o tipo **registro** podemos agrupar variáveis de maneira a organizar nossos dados, facilitando o entendimento do código e, assim, evitando erros.

Exemplo:

```
Funcionario: registro
        código:inteiro
        nome:caracter
        salario:real
        fim
```

Registro

Leitura de um registro

Leia(funcionário.código, funcionário.nome, funcionario.salario)

Escrever um registro

Escrever(funcionário.código, funcionário.nome, funcionario.salario)

Registro

- Considere um algoritmo que lê o código, nome e salário de 15 funcionários e, no final, apresenta o nome dos funcionários com salário superior a R$ 3.000,00.
- Para resolver esse algoritmo é necessário armazenar esses dados para os 15 funcionários. Sem o conceito de registro, a alternativa seria criar três vetores, um para cada informação.

Registro

```
Variaveis vf: vetor [1..15] de registro
                                        código:inteiro
                                        nome:inteiro
                                        salario:real
                                fim
                        i:inteiro
Inicio
        Escreva("informe código, nome e salario – 15 funcionario")
        para i<-1 ate 15 faça
                leia(vf[i].código, vf[i].nome, vf[i].salario)
        escreva("Funcinarios com salario superior a R$3.000,00")
        para i<-1 ate 15 faça
                se vf[i].salario>3000 então
                escreva (vf[i].nome)
fim
```