

Laços

Roland Teodorowitsch

Fundamentos de Programação - Escola Politécnica - PUCRS

24 de agosto de 2022

Introdução

Objetivos

- Implementar laços com `while`, `for` e `do`
- Acompanhar a execução de um programa manualmente
- Familiarizar-se com os algoritmos comuns com laços
- Entender laços aninhados
- Implementar programas que leem e processam conjuntos de dados

Conteúdos

- O Laço `while`
- Resolução de Problemas: Teste de Mesa
- O Laço `for`
- O Laço `do`
- Sentinelas de Processamento
- Algoritmos Comuns com Laços
- Laços Aninhados
- Resolução de Problemas: *Storyboards*
- Resumo

O Laço while

O Laço while

- Exemplos de aplicações com laços

- Cálculo de juros compostos
- Cálculo de um somatório
- Cálculo de um produtório
- Cálculo de média
- Simulações, programas orientados a eventos, etc.

- Cálculo de juros compostos (Unidade 1)

Inicie com um valor de ano igual a zero e um total de \$10.000

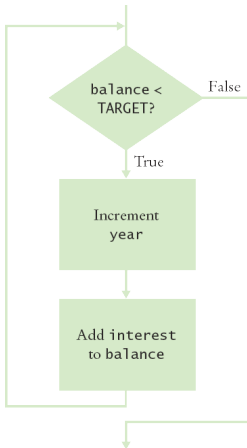
Repita o seguinte enquanto o total seja menor do que \$20.000

 Adicione 1 ao valor do ano

 Multiplique o total por 1,05 (crescimento de 5%)

Informe o último valor atribuído ao ano como resposta

Planejando um Laço `while`



- Um laço executa instruções repetidamente enquanto uma condição for verdadeira

```
while (balance < TARGET) {  
    year++;  
    double interest = balance * RATE/100;  
    balance = balance + interest;  
}
```

Sintaxe do Comando `while`

This variable is declared outside the loop and updated in the loop.

If the condition never becomes false, an infinite loop occurs.

```
double balance = 0;  
.  
.  
.  
while (balance < TARGET)  
{  
    double interest = balance * RATE / 100;  
    balance = balance + interest;  
}
```

This variable is created in each loop iteration.

These statements are executed while the condition is true.

Beware of "off-by-one" errors in the loop condition.

Don't put a semicolon here!

Lining up braces is a good idea.

Braces are not required if the body contains a single statement, but it's good to always use them.

DoubleInvestment.java (HORSTMANN, 2013, p. 143)

```
/**
 * This program computes the time required to double an investment.
 */
public class DoubleInvestment {
    public static void main(String[] args) {
        final double RATE = 5;
        final double INITIAL_BALANCE = 10000;
        final double TARGET = 2 * INITIAL_BALANCE;
        double balance = INITIAL_BALANCE;
        int year = 0;
        // Count the years required for the investment to double
        while (balance < TARGET) {
            year++;
            double interest = balance * RATE / 100;
            balance = balance + interest;
        }
        System.out.println("The investment doubled after " + year + " years.");
    }
}
```

Resultado da execução:

The investment doubled after 15 years.

Execução do Laço (1)

1 Check the loop condition

balance = 10000

year = 0

```
while (balance < TARGET)
{
    year++;
    double interest = balance * RATE / 100;
    balance = balance + interest;
}
```

The condition is true

2 Execute the statements in the loop

balance = 10500

year = 1

interest = 500

```
while (balance < TARGET)
{
    year++;
    double interest = balance * RATE / 100;
    balance = balance + interest;
}
```

3 Check the loop condition again

balance = 10500

year = 1

```
while (balance < TARGET)
{
    year++;
    double interest = balance * RATE / 100;
    balance = balance + interest;
}
```

The condition is still true

Execução do Laço (2)

4 After 15 iterations

balance = 20789.28

year = 15

```
while (balance < TARGET)
{
    year++;
    double interest = balance * RATE / 100;
    balance = balance + interest;
}
```

The condition is no longer true

5 Execute the statement following the loop

balance = 20789.28

year = 15

```
while (balance < TARGET)
{
    year++;
    double interest = balance * RATE / 100;
    balance = balance + interest;
}
System.out.println(year);
```

Exemplos de Laços com while (1)

Laço	Saída	Explicação
<pre>i = 0; sum = 0; while (sum < 10) { i++; sum = sum + i; System.out.println(i+" "+sum); }</pre>	<pre>1 1 2 3 3 6 4 10</pre>	Quando <code>sum</code> for igual a 10, a condição do laço será falsa, e o laço se encerra.
<pre>i = 0; sum = 0; while (sum < 10) { i++; sum = sum - i; System.out.println(i+" "+sum); }</pre>	<pre>1 -1 2 -3 3 -6 4 -10 ...</pre>	Como <code>sum</code> nunca atinge 10, isto consiste em um “laço infinito”.
<pre>i = 0; sum = 0; while (sum < 0) { i++; sum = sum - i; System.out.println(i+" "+sum); }</pre>	(Nenhuma saída)	A condição <code>sum < 0</code> é falsa quando é testada pela primeira vez, e o laço nunca é executado.

Exemplos de Laços com while (2)

Laço	Saída	Explicação
<pre>i = 0; sum = 0; while (sum >= 10) { i++; sum = sum + i; System.out.println(i+" "+sum); }</pre>	(Nenhuma saída)	O programador provavelmente pensou: "Pare quando a soma for pelo menos 10". Entretanto a condição do laço controla quando o laço é executado, e não quando ele se encerra.
<pre>i = 0; sum = 0; while (sum < 10) ; { i++; sum = sum + i; System.out.println(i+" "+sum); }</pre>	(Nenhuma saída e o programa nunca termina)	Observe que o ponto-e-vírgula após o teste do laço faz com que o corpo do laço corresponda a um comando vazio. O programa executará para sempre pois <code>sum < 0</code> e este valor não será atualizado no corpo do laço (comando vazio).

Erros Comuns (1)

Não pense: “Já chegamos lá?”

- O corpo do laço somente será executado se a condição de teste for verdadeira
- A lógica para “Já chegamos lá?” executaria o corpo do laço se o teste fosse falso
- Se `bal` deve crescer até que alcance `TARGET`, qual versão executará o corpo do laço?

```
while (bal < TARGET) {  
    year++;  
    interest = bal * RATE;  
    bal = bal + interest;  
}
```

```
while (bal >= TARGET) {  
    year++;  
    interest = bal * RATE;  
    bal = bal + interest;  
}
```

Erros Comuns (2)

Laços Infinitos

- O corpo do laço será executado até que a condição de teste se torne falsa
- O que acontece se você se esquecer de atualizar a variável de teste?
 - `bal` é a variável de teste (`TARGET` não é alterado)
 - Seu programa ficará no laço para sempre! (ou até que você pare o programa)

```
while (bal < TARGET) {  
    year++;  
    interest = bal * RATE;  
    // bal = bal + interest;  
}
```

Erros Comuns (3)

Erros de Limite

- Uma variável do tipo contadora é frequentemente usada na condição de teste
- A variável contadora pode iniciar em 0 ou 1 (programadores frequentemente iniciam contadores com 0)
- Se você quer contar 5 dedos, qual código deve ser usado?

```
// Inicia em 0, usa-se <
int finger = 0;
final int FINGERS = 5;
while (finger < FINGERS) {
    System.out.println(finger);
    finger++;
}
// 0, 1, 2, 3, 4
```

```
// Inicia em 1, usa-se <=
int finger = 1;
final int FINGERS = 5;
while (finger <= FINGERS) {
    System.out.println(finger);
    finger++;
}
// 1, 2, 3, 4, 5
```


Resumo do Laço `while`

- Laços com `while` são usados com grande frequência
- Inicialize as variáveis antes do teste
- A condição é testada ANTES do corpo do laço
 - Isto é chamado pré-teste
 - A condição frequentemente usa uma variável contadora
- Algo dentro do corpo do laço deve alterar uma das variáveis usadas no teste
- Cuidado com laços infinitos!

Exercícios

Faça laços em Java usando `while` para mostrar:

- 1 Os números inteiros de 100 a 200
- 2 Os números inteiros de -10 a -50
- 3 Os números pares entre a e b (com $a \geq b$)
- 4 As 10 primeiras potências de 2
- 5 Os elementos de uma progressão aritmética de n elementos que inicia em a e tem razão r
- 6 A soma dos elementos do item anterior
- 7 Os elementos de uma progressão geométrica de n elementos que inicia em a e tem razão r
- 8 A soma dos elementos do item anterior

Resolução de Problemas: Teste de Mesa

Teste de Mesa

Exemplo: Calcule a soma dos dígitos de um número (por exemplo, para 1729 o valor seria $1+7+2+9$)

- Faça colunas para as principais variáveis (n , sum , $digit$)
- Acompanhe a execução e mantenha os valores das variáveis atualizados na tabela

```
int n = 1729;
int sum = 0;
while (n > 0) {
    int digit = n % 10;
    sum = sum + digit;
    n = n / 10;
}
System.out.println(sum);
```

Exercícios

Exercícios (1)

Escreva laços `while` em Java, declarando todas as variáveis utilizadas, para:

- 1 Mostrar os valores de 1 até 10.
- 2 Mostrar os valores de 10 até 1, em ordem regressiva.
- 3 Calcular a soma dos valores de 1 até 20.
- 4 Calcular o fatorial de um número inteiro lido do terminal.
- 5 Ler 20 pares de valores (`a` e `b`) escrevendo qual é o maior valor.
- 6 Ler um número inteiro e escrever se ele é primo ou não.

Exercícios (2)

Faça o teste de mesa para o trecho de programa em Java a seguir, mostrando todas as alterações de valores nas variáveis declaradas e todas as saídas de terminal, e considerando que os valores lidos do teclado serão respectivamente 2, 4, 3, 2, 0, 2.

```
Scanner in = new Scanner(System.in);
int x, a, n, z;

x = in.nextInt();
n = in.nextInt();
while (x > 0) {
    a = 1;
    while (a <= n) {
        z=a*n;
        System.out.println(z);
        a = a + 1;
    }
    x = in.nextInt();
    n = in.nextInt();
}
```

O Laço for

O Laço for

- Em Java, tudo que é feito com `while` pode ser feito também com `for`
- Mas use laços `for` quando
 - Houver uma variável de indução com início, atualização e fim claramente identificáveis
 - For interessante deixar o código mais conciso (controle do laço em uma única linha)
- Por exemplo, para fazer o somatório dos números de 1 até 10 poderíamos fazer:

```
int soma = 0;
// versao com while
int i = 1; // inicializacao
while (i <= 10) { // teste
    soma = soma + i;
    i++; // atualizacao
}
```

```
int soma = 0;
// versao com for
for (int i = 1; i <= 10; i++) {
    soma = soma + i;
}
```

Sintaxe do Comando `for`

```
for ( inicializacao; condicao; atualizacao)  
    corpo;
```

- O comando `for` tem 4 partes:
 - 1 Inicialização: é executada uma vez antes do laço iniciar
 - 2 Condição de permanência: é verificada antes de cada iteração (deve ser verdadeira)
 - 3 Corpo do laço (bloco de comandos ou comando único): executado enquanto a condição `for` verdadeira
 - 4 Atualização: executada sempre depois do bloco de comandos e antes de se fazer um novo teste da condição
- Depois da inicialização, o laço `for` repetirá um ciclo formado por
 - Teste da condição — Execução do corpo do laço — Atualização

Execução de um laço for

1 Initialize counter

counter = 1

```
for (counter = 1; counter <= 10; counter++)  
{  
    System.out.println(counter);  
}
```

2 Check condition

counter = 1

```
for (counter = 1; counter <= 10; counter++)  
{  
    System.out.println(counter);  
}
```

3 Execute loop body

counter = 1

```
for (counter = 1; counter <= 10; counter++)  
{  
    System.out.println(counter);  
}
```

4 Update counter

counter = 2

```
for (counter = 1; counter <= 10; counter++)  
{  
    System.out.println(counter);  
}
```

5 Check condition again

counter = 2

```
for (counter = 1; counter <= 10; counter++)  
{  
    System.out.println(counter);  
}
```

Exemplos de Laços for (1)

- Contar de 1 (inclusive) até 5 (inclusive) [5 iterações: 1 2 3 4 5]

```
for (int i = 1; i<=5; i++) System.out.println(i);  
for (int i = 1; i<6; i++) System.out.println(i);
```

- Contar de 1 (inclusive) até 5 (exclusive) [4 iterações: 1 2 3 4]

```
for (int i = 1; i<=4; i++) System.out.println(i);  
for (int i = 1; i<5; i++) System.out.println(i);
```

- Contar de 0 (inclusive) até 5 (inclusive) [6 iterações: 0 1 2 3 4 5]

```
for (int i = 0; i<=5; i++) System.out.println(i);  
for (int i = 0; i<6; i++) System.out.println(i);
```

- Contar de 0 (inclusive) até 5 (exclusive) [5 iterações: 0 1 2 3 4]

```
for (int i = 0; i<=4; i++) System.out.println(i);  
for (int i = 0; i<5; i++) System.out.println(i);
```

Exemplos de Laços for (2)

- Contagem regressiva [6 iterações: 5 4 3 2 1 0]

```
for (int i = 5; i >= 0; i--) System.out.println(i);
```

- Incremento igual a 2 [5 iterações: 0 2 4 6 8]

```
for (int i = 0; i < 9; i = i + 2) System.out.println(i);
```

- Razão geométrica igual a 2 [5 iterações: 1 2 4 8 16]

```
for (int i = 1; i <= 20; i = i * 2) System.out.println(i);
```

- Percorrer todas as letras de uma cadeia de caracteres [4 iterações: J A V A]

```
String s = "JAVA";  
for (int i = 0; i < s.length(); i++) System.out.println(s.charAt(i));
```

Planejando um Laço for



- Considerando um valor inicial investido de 10000, e uma taxa de juros anual de 5%, escreva um programa que: leia o número total de anos de investimento (`numAnos`) e, para cada ano transcorrido, imprima o número do ano e o saldo total ao final desse ano.
- Por exemplo, para `numAnos` igual a 5, o programa deve imprimir:
 - ano 1: 10500.00
 - ano 2: 11025.00
 - ano 3: 11576.25
 - ano 4: 12155.06
 - ano 5: 12762.82

```
for (int ano = 1; ano <= numAnos; ano++) {  
    // Atualiza saldo  
    // Imprime ano e saldo  
}
```

Investimento.java

```
import java.util.Scanner;

/**
 * Este programa imprime uma tabela mostrando o crescimento anual de um investimento.
 */
public class Investimento {
    public static void main(String[] args) {
        final double TAXA = 5.0;
        final double SALDO_INICIAL = 10000;
        double saldo = SALDO_INICIAL;

        System.out.print("Quantos anos? ");
        Scanner in = new Scanner(System.in);
        int numAnos = in.nextInt();

        for (int ano = 1; ano <= numAnos; ano++) {
            double juros = saldo * TAXA / 100.0;
            saldo = saldo + juros;
            System.out.printf("ano %d: %.2f\n", ano, saldo);
        }
    }
}
```

Cuidados a serem tomados

- Limite final do laço: inclusive ou exclusive?
- Variável de indução: crescente ou decrescente?
 - Com valores crescentes usa-se \leq ou $<$
 - Com valores decrescentes usa-se \geq ou $>$
- Condições erradas podem gerar **laços infinitos**:

```
// 0 2 4 6 8 10 ...  
for (int i=0; i!=9; i=i+2) System.out.println(i);  
  
// 0 -1 -2 -3 -4 -5 ...  
for (int i=0; i<10; --i) System.out.println(i);
```

- Evite atualizar o contador no corpo do laço for:

```
for (int i=1; i<=100; i++) {  
    if (i % 10 == 0)           // Pular valores divisíveis por 10  
        i++;                  // Estilo ruim para for...  
    System.out.println(i);  
}
```


Escopo de Variáveis do Laço `for`

- Escopo é o “tempo de vida” de uma variável.
- Quando a variável `x` é declarada no comando `for`, ela existe apenas dentro do bloco do laço

```
for ( int x = 1; x < 10; x = x + 1) {  
    // comandos a serem executados dentro do laço  
    // 'x' pode ser usado em qualquer lugar dentro deste bloco  
}  
if (x > 100)    // Erro! 'x' esta fora de escopo!
```

- Solução: declarar `'x'` fora do laço

```
int x;  
for ( x = 1; x < 10; x = x + 1) {}
```

Resumo do Laço `for`

- Laços com `for` são muito usados
- Eles têm uma notação bastante concisa
 - Inicialização ; Condição ; Atualização ; Corpo do laço
 - A inicialização acontece uma única vez no início do laço
 - A condição é testada todas as vezes ANTES (pré-teste) de executar o corpo do laço
 - O incremento é realizado APÓS o corpo do laço
- Use laços `for` seguindo o modelo padrão
- Adequado para contagens inteiras, processamento de *strings*, vetores ou matrizes, etc.

Exercícios sobre Laço `for`

Exercícios 1 e 2

- 1 Escreva laços `for` em Java, declarando todas as variáveis utilizadas, para:
 - a. Mostrar os valores de 1 até 10.
 - b. Mostrar os valores de 10 até 1, em ordem regressiva.
 - c. Calcular a soma dos valores de 1 até 20.
 - d. Calcular o fatorial de um número inteiro lido do terminal.
 - e. Ler 20 pares de valores (*a* e *b*) escrevendo qual é o maior valor.
 - f. Ler um número inteiro e escrever se ele é primo ou não.
- 2 Escreva um programa em Java para ler o número de alunos de uma turma e a seguir ler as notas destes alunos na prova da disciplina, determinando e imprimindo: a média da turma, a nota mais baixa e a nota mais alta.

Exercício 3

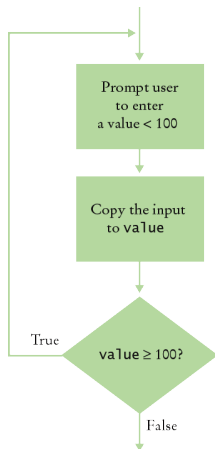
- 3 Faça o teste de mesa para o trecho de programa em Java a seguir, mostrando todas as alterações de valores nas variáveis declaradas e todas as saídas de terminal, e considerando que os valores lidos do teclado serão respectivamente 4, 2, 2, 3, 2, 0.

```
Scanner in = new Scanner(System.in);
int x, a, n, z;

n = in.nextInt();
for ( x = in.nextInt() ; x > 0 ; x = in.nextInt() ) {
    for ( a = 1 ; a <= n ; a = a + 1 ) {
        z=a*n;
        System.out.println(z);
    }
    n = in.nextInt();
}
```

O Laço do

O Laço do



- Usa-se o laço `do` quando se deseja executar o corpo do laço pelo menos uma vez, testando a condição APÓS a primeira repetição do laço

```
int i = 1; // inicializacao
final int FINGERS = 5;
do {
    // comandos...
    i++; // atualizacao
} while (i <= FINGERS); // teste
```

Exemplo de Laço do

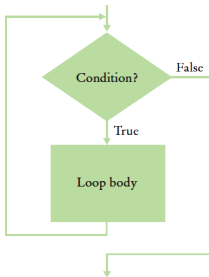
- Validação da entrada de usuários
 - Verificar se um valor lido está dentro do limite
 - O usuário tem que fornecer alguma entrada antes para ser validada

```
int valor;  
do {  
    System.out.println("Forneca um valor inteiro < 100: ");  
    valor = in.nextInt();  
} while (valor >= 100); // Teste
```

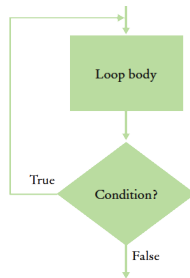

Dica de Programação

- Fluxogramas para laços: evite código "spaghetti"(nunca faça uma seta apontar para dentro do corpo de um laço)

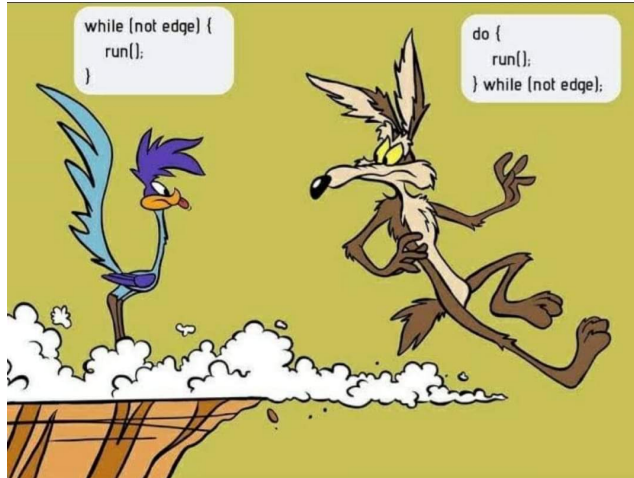
`while` e `for` testam antes



`do` testa depois



Humor



Sentinelas de Processamento

Sentinelas de Processamento

- Valores de sentinela indicam o final de um conjunto de dados, mas não fazem parte dos dados
- Podem ser usados em muitos casos
 - Quando não se sabe quantos itens há em uma lista, usa-se um caracter ou valor especial para sinalizar que não há mais itens
 - Para entradas de números positivos, é comum usar o valor -1:

```
salary = in.nextDouble();  
while (salary != -1) {  
    sum = sum + salary;  
    count++;  
    salary = in.nextDouble();  
}
```

Calculando a Média de um Conjunto Indeterminado de Valores

- Declare e inicialize uma variável `sum` com 0
- Declare e inicialize uma variável `count` com 0
- Declare e inicialize uma variável `input` com 0
- Mostre uma mensagem solicitando que o usuário forneça os dados
- Fique no laço até que o valor de sentinela seja fornecido
 - Leia um valor e salve em `input`
 - Se `input` não for igual a -1
 - Adicione `input` em `sum`
 - Adicione 1 na variável `count`
- Tenha certeza de que pelo menos um valor for fornecido antes de fazer a divisão
 - Divida `sum` por `count` e mostre o resultado
- Fim

SentinelDemo.java (1) (HORSTMANN, 2013, p. 158-159)

```
import java.util.Scanner;

/**
 * This program prints the average of salary values that are terminated with a sentinel.
 */

public class SentinelDemo {
    public static void main(String[] args) {
        double sum = 0;
        int count = 0;
        double salary = 0;
        System.out.print("Enter salaries, -1 to finish: ");
        Scanner in = new Scanner(System.in);

        // Process data until the sentinel is entered
        while (salary != -1) {
            salary = in.nextDouble();
            if (salary != -1) {
                sum = sum + salary;
                count++;
            }
        }
    }
}
```

SentinelDemo.java (2) (HORSTMANN, 2013, p. 158-159)

```
// Compute and print the average
if (count > 0) {
    double average = sum / count;
    System.out.println("Average salary: " + average);
}
else {
    System.out.println("No data");
}
}
```

Execução do programa:

Enter salaries, -1 to finish: 10 10 40 -1

Average salary: 20

Variáveis Booleanas e Sentinelas

- Uma variável booleana (frequentemente chamada de *flag* pode ser usada para controlar um laço)

```
System.out.print("Enter salaries, -1 to finish: ");  
boolean done = false;  
while (!done) {  
    value = in.nextDouble();  
    if (value == -1) {  
        done = true;  
    }  
    else {  
        // Process value  
    }  
}
```


Para permitir qualquer valor numérico...

- Se os valores válidos puderem ser negativos ou positivos, não se pode usar -1 (ou qualquer outro número) como sentinela
- A solução então é usar qualquer outra sentinela não numérica
- Como `in.nextDouble` falha para valores não numéricos, deve-se usar `in.hasNextDouble` antes
 - Retorna um booleano: `true`, se a entrada estiver correta (for um número), ou `false`, se a entrada não for um número
 - Em caso de `true`, pode-se usar `in.nextDouble`

```
System.out.print("Enter values, Q to quit: ");  
while (in.hasNextDouble()) {  
    value = in.nextDouble();  
    // Process value  
}
```

Algoritmos Comuns com Laços

Algoritmos Comuns com Laços

- Somatório
- Produtório
- Valor médio
- Contagem de ocorrências
- Encontrar a primeira ocorrência
- Perguntar até que uma ocorrência seja encontrada
- Máximo e mínimo
- Comparar valores adjacentes

Somatório

- Inicialize `total` com 0
- Pode-se usar o laço com sentinela
- Acrescente o valor em `total`

```
double total = 0;
while (in.hasNextDouble()) {
    double input = in.nextDouble();
    total = total + input;
}
```

Produtório

- Inicialize `produto` com 1
- Multiplique o valor por `produto`

```
double produto = 1;
while (in.hasNextDouble()) {
    double input = in.nextDouble();
    produto = produto * input;
}
```

Valor Médio

- Faça o somatório dos valores
- Inicialize `count` com 0, incrementando-a a cada valor lido
- Verifique o valor de `count` antes da divisão!

```
double total = 0;
int count = 0;
while (in.hasNextDouble()) {
    double input = in.nextDouble();
    total = total + input;
    count++;
}
double average = 0;
if (count > 0) {
    average = total / count;
}
```

Contagem de Ocorrências

- Inicialize `count` com 0
- Use um laço `for`
- Incremente o contador a cada ocorrência

```
int upperCaseLetters = 0;
for (int i = 0; i < str.length(); i++) {
    char ch = str.charAt(i);
    if (Character.isUpperCase(ch)) {
        upperCaseLetters++;
    }
}
```

Encontrar a Primeira Ocorrência

- Inicialize uma variável sentinela/booleana com `false`
- Inicialize o contador de posições com 0 (primeiro caracter do *string*)
- Use uma condição composta no laço
- Laços com pré-teste tratam a situação para *string* vazio

```
boolean found = false;
char ch;
int position = 0;
while (!found && position < str.length()) {
    ch = str.charAt(position);
    if (Character.isLowerCase(ch)) {
        found = true;
    }
    else { position++; }
}
```


Perguntar até que uma Ocorrência Seja Encontrada

- Inicialize uma variável sentinela/booleana com `false`
- Teste a variável sentinela no laço `while`
 - Leia a entrada, e compare com o limite
 - Se a entrada está dentro do limite, altere a variável sentinela para `true`
 - O laço parará de executar

```
boolean valid = false;
double input;
while (!valid) {
    System.out.print("Please enter a positive value < 100: ");
    input = in.nextDouble();
    if (0 < input && input < 100) { valid = true; }
    else { System.out.println("Invalid input."); }
}
```

Máximo e Mínimo

- Leia o primeiro valor: este é o maior (ou menor) valor que você obteve até agora!
- Fique no laço enquanto você tiver um valor válido
 - Leia outro valor
 - Compare o novo valor com o maior (ou menor)
 - Atualize o maior (ou menor) valor se for necessário

```
double largest = in.nextDouble();  
while (in.hasNextDouble()) {  
    double input = in.nextDouble();  
    if (input > largest) {  
        largest = input;  
    }  
}
```

```
double smallest = in.nextDouble();  
while (in.hasNextDouble()) {  
    double input = in.nextDouble();  
    if (input < smallest) {  
        smallest = input;  
    }  
}
```

Comparar Valores Adjacentes

- Obtenha o primeiro valor da entrada
- Use o `while` para determinar se há mais valores para serem verificados
 - Copie a entrada para uma variável para armazenar o valor anterior
 - Leia a próxima entrada
 - Compare a entrada lida com o valor anterior, e avise se forem iguais

```
double input = in.nextDouble();
while (in.hasNextDouble()) {
    double previous = input;
    input = in.nextDouble();
    if (input == previous) {
        System.out.println("Duplicate input");
    }
}
```

Passos para Escrever um Laço

Planejamento:

- Decida que tarefa realizar dentro do laço
- Especifique a condição do laço
- Determine o tipo do laço
- Inicialize as variáveis antes da primeira iteração
- Processe os resultados depois que o laço tenha encerrado
- Teste o laço com exemplos típicos

Codificação:

- Implemente o laço em Java

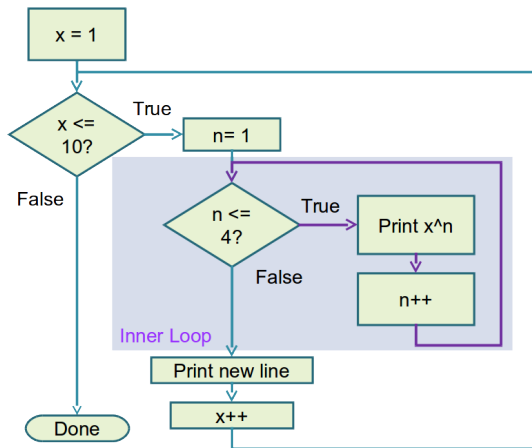
Laços Aninhados

Laços Aninhados

- Como você imprimiria uma tabela com linhas e colunas?
 - Imprima a primeira linha com o cabeçalho
 - Use um laço
 - Imprima o corpo da tabela
 - Quantas linhas?
 - Quantas colunas?
 - Faça um laço para as linhas
 - Faça um laço para as colunas

x^1	x^2	x^3	x^4
1	1	1	1
2	4	8	16
3	9	27	81
...
10	100	1000	10000

Fluxograma de Dois Laços Aninhados



PowerTable.java (HORSTMANN, 2013, p. 173-174)

```
/** This program prints a table of powers of x. */
public class PowerTable {
    public static void main(String[] args) {
        final int NMAX = 4;
        final double XMAX = 10;

        // Print table header
        for (int n = 1; n <= NMAX; n++) {
            System.out.printf("%10d", n);
        }
        System.out.println();
        for (int n = 1; n <= NMAX; n++) {
            System.out.printf("%10s", "x ");
        }
        System.out.println();

        // Print table body
        for (double x = 1; x <= XMAX; x++) {
            // Print table row
            for (int n = 1; n <= NMAX; n++) {
                System.out.printf("%10.0f", Math.pow(x, n));
            }
            System.out.println();
        }
    }
}
```


Resultado de PowerTable.java

1	2	3	4
x	x	x	x
1	1	1	1
2	4	8	16
3	9	27	81
4	16	64	256
5	25	125	625
6	36	216	1296
7	49	343	2401
8	64	512	4096
9	81	729	6561
10	100	1000	10000

Exercício

Modifique o programa `PowerTable.java` para que a tabela seja impressa “deitada”, ou seja, na primeira linha os valores para x^1 , na segunda linha os valores para x^2 , e assim por diante.

Exemplos de Laços Aninhados (1)

Laço	Saída	Explicação
<pre>for (i = 1; i <= 3 ; i++) { for (j = 1; j <= 4; j++) { System.out.print("*"); } System.out.println(); }</pre>	<pre>**** **** ****</pre>	<p>Imprime 3 linhas de 4 asteriscos cada.</p>
<pre>for (i = 1; i <= 4 ; i++) { for (j = 1; j <= 3; j++) { System.out.print("*"); } System.out.println(); }</pre>	<pre>*** *** *** ***</pre>	<p>Imprime 4 linhas de 3 asteriscos cada.</p>

Exemplos de Laços Aninhados (2)

Laço	Saída	Explicação
<pre>for (i = 1; i <= 4 ; i++) { for (j = 1; j <= i; j++) { System.out.print("*"); } System.out.println(); }</pre>	<pre>* ** *** ****</pre>	<p>Imprime 4 linhas com tamanhos 1, 2, 3 e 4.</p>
<pre>for (i = 1; i <= 3 ; i++) { for (j = 1; j <= 5; j++) { if (j % 2 == 0) { System.out.print("*"); } else { System.out.print("-"); } } System.out.println(); }</pre>	<pre>-*-*- -*-*- -*-*-</pre>	<p>Imprime asteriscos nas colunas pares e traços nas colunas ímpares.</p>

Exemplos de Laços Aninhados (3)

Laço	Saída	Explicação
<pre>for (i = 1; i <= 3 ; i++) { for (j = 1; j <= 5; j++) { if (i % 2 == j % 2) { System.out.print("*"); } else { System.out.print(" "); } } System.out.println(); }</pre>	<pre>* * * * * * * *</pre>	Imprime o padrão de um tabuleiro de damas.

Exercício

Faça laços aninhados para imprimir:

a)
 * *
 . * . . .
 . . * . .
 . . . * .
 *

b)
 * * * * *
 . * * * *
 . . * * *
 . . . * *
 *

c)

 *
 * * . . .
 * * * . .
 * * * * .

d)
 *
 . . . * .
 . . * . .
 . * . . .
 *

e)
 * * * * *
 * * * * .
 * * * . .
 * * . . .
 *

f)
 *
 . . . * *
 . . * * *
 . * * * *
 * * * * *

g)
 * * * * *
 . * * * .
 . . * . .

h)
 * . . . *
 . * . * .
 . . * . .
 . * . * .
 * . . . *

i)

 . . * . .
 . * * * .
 * * * * *

Resolução de Problemas: Storyboards (Esboços Sequenciais)

Resolução de Problemas: *Storyboards* (Esboços Sequenciais)

- Um storyboard (esboço sequencial) consiste numa sequência de desenhos anotados para cada etapa de uma sequência de ações
- Trata-se de uma técnica útil para solução de problemas que permite modelar a interação com o usuário
- Pode ajudar a responder:
 - Qual informação o usuário deve fornecer e em que ordem?
 - Qual informação o programa deve mostrar e em que formato?
 - O que deve acontecer se houver um erro?
 - Quando o programa deve terminar?

Exemplo de *Storyboard*

- Objetivo: converter uma sequência de medidas
 - Exigirá um laço e algumas variáveis
 - Deverá gerenciar uma conversão de cada vez através de um laço

Converting a Sequence of Values

What unit do you want to convert from? *cm*

What unit do you want to convert to? *in*

Enter values, terminated by zero ————— Allows conversion of multiple values

30

30 cm = 11.81 in

100

100 cm = 39.37 in

0

What unit do you want to convert from?

Format makes clear what got converted

O que pode dar errado?

- Unidades de medidas desconhecidas
 - Como centímetros e polegadas são digitados?
 - Que outras conversões estão disponíveis?
- Solução: mostra uma lista de tipos de unidades aceitáveis

From unit (in, ft, mi, mm, cm, m, km, oz, lb, g, kg, tsp, tbsp, pint, gal): **cm**
To unit: **in** — No need to list the units again

O que mais pode dar errado?

- Como o usuário encerra o programa?

Exiting the Program

From unit (in, ft, mi, mm, cm, m, km, oz, lb, g, kg, tsp, tbsp, pint, gal): **cm**

To unit: **in**

Enter values, terminated by zero

30

30 cm = 11.81 in

0

More conversions (y, n)? **n**

(Program exits)

Sentinel triggers the prompt to exit

- Storyboards ajudam a planejar um programa: conhecer os fluxos ajuda a estruturar o código

Resumo

Resumo (1)

- Há 3 tipos de laços:
 - Laços `while`
 - Laços `for`
 - Laços `do`
- Cada laço possui as seguintes seções:
 - Inicialização (preparação das variáveis para iniciar o laço)
 - Condição (teste para verificar se o corpo do laço deve ser executado)
 - Atualização (alteração de alguma variável testada na condição)

Resumo (2)

- Um laço executa instruções repetidamente enquanto uma condição for verdadeira
- Errar o número de iterações em laço por uma unidade é um erro comum de programação
 - Procure deixar os testes simples para evitar este tipo de erro.
- O laço `for` é usado quando um valor varia de um ponto de partida até um ponto final com um incremento ou decremento constante
- O laço `do` é apropriado quando o corpo do laço deve ser executado pelo menos uma vez

Resumo (3)

- Um valor de sentinela consiste de um valor que determina o final de um conjunto de dados, mas que não faz parte deste conjunto
- Você pode usar uma variável booleana para controlar um laço
 - Defina a variável com `true` antes de entrar no laço, e então defina ela com `false` para sair do laço
- Quando o corpo de um laço contiver outro laço, os laços são aninhados
 - Um uso típico para laços aninhados é a impressão de uma tabela com linhas e colunas
- Em uma simulação, o computador é usado para simular uma atividade
 - Pode-se introduzir aleatoriedade chamando o gerador de números aleatórios

Tópicos Avançados

Tópicos Avançados

- Números Aleatórios e Simulações

Números Aleatórios e Simulações

- Jogos frequentemente usam números aleatórios para tornar as coisas mais interessantes
 - Jogar dados
 - Girar uma roda
 - “Comprar” uma carta
- Uma simulação usualmente envolve executar um laço para uma sequência de eventos
 - Dias
 - Eventos

Exemplo: RandomDemo.java [Publicado em Horstmann (2013, p. 176)]

- O método `Math.random()` pode ser usado para gerar números aleatórios no intervalo `[0;1)`

```
/**  
    Este programa imprime 10 números aleatórios entre 0 e 1.  
*/  
public class RandomDemo {  
    public static void main(String[] args) {  
        for (int i = 1; i <= 10; i++) {  
            double r = Math.random();  
            System.out.println(r);  
        }  
    }  
}
```

Resultado de RandomDemo.java

```
0.6513550469421886  
0.920193662882893  
0.6904776061289993  
0.8862828776788884  
0.7730177555323139  
0.3020238718668635  
0.0028504531690907164  
0.9099983981705169  
0.1151636530517488  
0.1592258808929058
```

Simulando o Lançamento de Dados

- Objetivo: obter valores inteiros aleatórios entre 1 e 6
- Dice.java (HORSTMANN, 2013, p. 177)

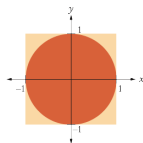
```
/** This program simulates tosses of a pair of dice. */
public class Dice {
    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            // Generate two random numbers between 1 and 6
            int d1 = (int) (Math.random() * 6) + 1;
            int d2 = (int) (Math.random() * 6) + 1;
            System.out.println(d1 + " " + d2);
        }
    }
}
```

- Resultado de Dice.java:

```
5 1
2 1
1 2
5 1
1 2
6 4
4 4
6 1
6 3
5 2
```

O Método Monte Carlo

- Usado para encontrar soluções aproximadas para problemas que não podem ser resolvidos com precisão
- Exemplo: aproximar o valor de π usando áreas relativas de um círculo dentro de um quadrado
 - Usa aritmética simples
 - Hits (Acertos) estão dentro do círculo
 - Tries são o número total de tentativas
 - Razão é $4 \times \text{Hits} / \text{Tries}$



MonteCarlo.java (HORSTMANN, 2013, p. 178-179)

```

/**
 * This program computes an estimate of pi by simulating dart throws onto a square.
 */
public class MonteCarlo {
    public static void main(String[] args) {
        final int TRIES = 10000;
        int hits = 0;
        for (int i = 1; i <= TRIES; i++) {
            // Generate two random numbers between -1 and 1
            double r = Math.random();
            double x = -1 + 2 * r; // Between -1 and 1
            r = Math.random();
            double y = -1 + 2 * r;
            // Check whether the point lies in the unit circle
            if (x * x + y * y <= 1) { hits++; }
        }
        /* The ratio hits / tries is approximately the same as the ratio
           circle area / square area = pi / 4 */
        double piEstimate = 4.0 * hits / TRIES;
        System.out.println("Estimate for pi: " + piEstimate);
    }
}

```

Resultado:

Estimate for pi: 3.1504

Referências

Referências

HORSTMANN, C. **Java for Everyone – Late Objectt**. 2. ed. Hoboken: Wiley, 2013. xxxiv, 589 p.