



IFRN

Tecnologia em Análise e Desenvolvimento de
Sistemas

Métodos, Decisão e Repetição em C#

Prof. Gilbert Azevedo



Conteúdo

- Métodos
- Escopo
- Variáveis Booleanas
- Operadores Relacionais e Lógicos
- Instruções de Decisão: if, if-else, switch, ?
- Instruções de Repetição: for, while, do-while
- Exercícios



Métodos

- São seqüências nomeadas de instruções
- Possuem um nome e um corpo
 - O nome do método deve ser um identificador significativo para sua funcionalidade
 - O corpo do método contém a seqüência de instruções a ser executada
- Os métodos podem receber dados (parâmetros), retornar dados (retorno) e ter um controle de acesso
 - [accessModifier] returnType methodName (parameterList)
 - {
 - // Instruções
 - }

Exemplo de Método

Assinatura do método

Acesso

Retorno

Identificador

Lista de parâmetros

- public double AreaTriangulo(double b, double h)
- {
- double area; } Variável local
- area = b * h / 2;
- return area; } Resultado retornado
- }



Dicas sobre Métodos

- Métodos podem possuir o mesmo nome se tiverem lista de parâmetros diferentes (sobrecarga)
 - `string ToString();`
 - `string ToString(string format);`
- Métodos que não possuem retorno utilizam a palavra-chave *void*
 - `void ShowError(string errorMsg);`
- Os identificadores públicos devem seguir a notação *PascalCase* (iniciais em maiúsculo) e os privados, a notação *camelCase* (primeira inicial em minúsculo)
- O acesso padrão é *private*



Chamando os métodos

- Para chamar um método, utiliza-se a sintaxe *MethodName(argumentList)*
 - MethodName é o identificador do método
 - argumentList é a lista de dados enviada para o método
 - A quantidade e o tipo dos argumentos deve ser compatível com a lista de parâmetros
 - O resultado, se existir, pode ser utilizado em uma expressão
- `public double AreaTriangulo(double b, double h) { ... }`
- `double area = AreaTriangulo(10, 20);`



Escopo

- O escopo define a região do programa na qual um identificador é válido
 - `class Example {`
 - `public void Metodo1() {`
 - `int minhaVariavel; // Escopo local`
 - `}`
 - `public void Metodo2() {`
 - `minhaVariavel = 10; // Erro – fora do escopo`
 - `}`
 - `}`
- Variáveis dentro de métodos possuem o escopo local



Escopo da Classe

- Variáveis declaradas dentro de uma classe (chamada campos) possuem o escopo da classe
 - class Example {
 - int meuCampo; // Escopo da classe
 - public void Metodo1() {
 - meuCampo = 10; // Ok
 - }
 - public void Metodo2() {
 - meuCampo = 10; // Ok
 - }
 - }



Instruções de Decisão

- Instruções de decisão são freqüentemente usadas em um programa para testar se um comando deve ou não ser executado
 - Variáveis Booleanas (bool)
 - Operadores Relacionais
 - Operadores Lógicos
 - Instruções if, if-else
 - Operador condicional ternário
 - Instruções switch
 - If-else aninhados



Variáveis Booleanas e Operadores Relacionais

- Variáveis booleanas são definidas com a palavra chave *bool*
 - Uma variável bool pode armazenar os valores true (verdadeiro) ou false (falso)
- Operadores relacionais são utilizados para realizar comparações entre dois valores (de tipos compatíveis)
 - O resultado de um operador relacional é sempre um valor booleano. O operador `==` testa se dois valores são iguais.
 - `bool b;`
 - `int i = 10;`
 - `b = i == 5; // b recebe falso`



Operadores Relacionais do C#

Operação	C#	Operandos
Igual	==	I,R,C,L,S
Diferente	!=	I,R,C,L,S
Maior ou igual	>=	I,R,C
Maior	>	I,R,C
Menor ou igual	<=	I,R,C
Menor	<	I,R,C



Operadores Lógicos do C#

Operação	C#	Operandos
Conjunção (e)	&&	L
Disjunção (ou)		L
Negação (não)	!	L

X	Y	X e Y
F	F	F
F	V	F
V	F	F
V	V	V

X	Y	X ou Y
F	F	F
F	V	V
V	F	V
V	V	V

X	não X
F	V
V	F



Precedência de Operadores

- Operadores unários
 - `!` → Negação
 - `+`, `-` → Adição e subtração
 - `++`, `--` → Prefixo de incremento e decremento
- Aritméticos multiplicativos
 - `*`, `/`, `%` → Multiplicação, divisão e resto
- Aritméticos aditivos
 - `+`, `-` → Adição e subtração
- Relacional
 - `<`, `<=`, `>`, `>=` → Menor (igual), maior (igual)
 - `==`, `!=` → Igual e diferente
- Lógico
 - `&&`, `||` → E lógico, Ou lógico



Instruções If

- Controla a execução de uma instrução (ou bloco de instruções) de acordo com uma expressão booleana
- if (booleanExpression)
- instrução-1; // Executa se verdadeiro
- if (booleanExpression)
- {
- bloco-1; // Executa se verdadeiro
- }

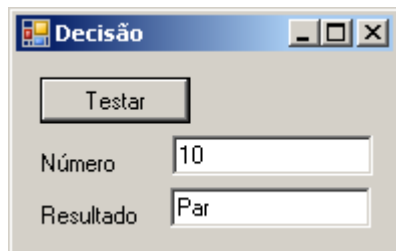


Instruções If-Else

- Controla a execução de duas instruções (ou bloco de instruções) de acordo com uma expressão booleana
- if (booleanExpression)
- instrução-1; // Executa se verdadeiro
- else
- instrução-2; // Executa se falso

Exemplo If-Else

- `private void button1_Click(object sender, EventArgs e)`
- `{`
- `int x = int.Parse(textBox1.Text);`
- `if (x % 2 == 0) textBox2.Text = "Par";`
- `else textBox2.Text = "Ímpar";`
- `}`



A screenshot of a Windows application window titled "Decisão". The window has a standard Windows XP-style title bar with minimize, maximize, and close buttons. Inside the window, there is a button labeled "Testar". Below the button, there are two text boxes. The first is labeled "Número" and contains the value "10". The second is labeled "Resultado" and contains the value "Par".



Operador Condicional Ternário

- O operador condicional ternário ? retorna um valor ou outro de acordo com uma expressão booleana
- `result = booleanExpression ? valor-1 : valor-2;`
- `private void button1_Click(object sender, EventArgs e)`
- `{`
- `int x = int.Parse(textBox1.Text);`
- `textBox2.Text = x % 2 == 0 ? "Par" : "Ímpar";`
- `}`



Instruções Switch

- Controla a execução de várias instruções de acordo com valores de uma expressão de controle que deve ser inteira ou string
- `switch(controllingExpression)`
- `{`
- `case constantExpression : instruções; break;`
- `default : instruções; break;`
- `}`



Regras do Switch no C#

- Os rótulos case devem ser expressões constantes
 - Os rótulos devem ser únicos
 - Rótulos vazios são permitidos, mas rótulos não vazios devem encerrar com um break (return ou throw)
 - O rótulo default é opcional
-
- `switch(Naipe) {`
 - `case "Copas" :`
 - `case "Ouros" : Cor = "Vermelho"; break;`
 - `case "Paus" : Cor = "Preto"; // Erro - sem break;`
 - `case "Espadas" : Cor = "Preto"; break;`
 - `}`



If-Else Aninhados

- Método que compara duas datas, retornando -1 (data1 menor), 1 (data1 maior) ou 0 (datas iguais)
 - `int dateCompare(DateTime data1, DateTime data2) {`
 - `int result;`
 - `if (data1.Year < data2.Year) result = -1;`
 - `else if (data1.Year > data2.Year) result = 1;`
 - `else if (data1.Month < data2.Month) result = -1;`
 - `else if (data1.Month > data2.Month) result = 1;`
 - `else if (data1.Day < data2.Day) result = -1;`
 - `else if (data1.Day > data2.Day) result = 1;`
 - `else result = 0;`
 - `return result;`
 - `}`



Instruções de Repetição

- Instruções de repetição são frequentemente usadas em um programa para repetir comandos
 - Operadores de atribuição compostos
 - Repetições while
 - Repetições for
 - Repetições do-while
 - break, continue



Operadores de atribuição compostos

- São utilizados como atalho para combinar operadores aritméticos com uma atribuição

Operação	Operador composto
variável = variável * número	variável *= número
variável = variável / número	variável /= número
variável = variável % número	variável %= número
variável = variável + número	variável += número
variável = variável - número	variável -= número



Instrução While

- Controla a repetição de instruções enquanto uma expressão booleana for verdadeira
- while (booleanExpression)
- instrução-1; // Repete se verdadeiro
- while (booleanExpression)
- {
- bloco-1; // Repete se verdadeiro
- }



Dicas sobre While

- Se a expressão booleana for falsa no primeiro teste, a instrução não é executada nenhuma vez
- A expressão booleana deve ser falsa em um tempo finito
- Ex: laço de 1 a 10 com while
 - `int i = 1;`
 - `while (i <= 10) {`
 - `// Faça alguma coisa`
 - `i++;`
 - `}`



Instrução For

- De forma análoga ao while, controla a repetição de instruções enquanto uma expressão booleana for verdadeira
 - inicialização;
 - while (booleanExpression) {
 - instruções;
 - atualização da variável de controle;
 - }
 - for (inicialização; booleanExpression; atualização da variável de controle) {
 - instruções
 - }



Exemplos de For

- Ex: laço de 1 a 10 com for
 - `for (int i = 1; i <= 10; i++) {`
 - `// Faça alguma coisa`
 - `}`
- Ex: laço com duas variáveis de controle
 - `for (int i = 0, j = 10; i <= j; i++, j--) {`
 - `// Faça alguma coisa`
 - `}`
- Ex: laço sem inicialização
 - `int i = 1;`
 - `for (; i <= 10; i++) {`
 - `// Faça alguma coisa`
 - `}`



Instrução Do-While

- De forma análoga ao while e ao for, controla a repetição de instruções enquanto uma expressão booleana for verdadeira
- A diferença é que as instruções são realizadas pelo menos uma vez, pois o teste ocorre no final
- do
- instrução-1; // Repete se verdadeiro
- while (booleanExpression);
- do {
- bloco-1; // Repete se verdadeiro
- } while (booleanExpression);



Exemplo de While

- Ex: laço de 1 a 10 com do-while
- `int i = 1;`
- `do`
- `{`
- `// Faça alguma coisa`
- `i ++;`
- `}`
- `while (i <= 10);`



Instruções Break e Continue

- A instrução break é utilizada para sair do corpo de uma instrução de repetição
- A instrução continue faz com que a próxima iteração do laço seja executada (re-avaliando a expressão)
- `int i = 1;`
- `while (true) {`
- `// Faça alguma coisa`
- `i++;`
- `if (i != 11) continue;`
- `else break;`
- `}`



Exercícios

- 1. Ler quatro números inteiros, calcular a soma dos números pares e a soma dos números ímpares.
- 2. Calcular as raízes reais da equação $aX^2 + bX + c = 0$, dados a , b e c .
- 3. Ler três valores e dizer se eles formam um triângulo. Caso afirmativo, dizer seu tipo (equilátero, isósceles ou escaleno).
- 4. Ler três valores e apresentá-los em ordem crescente.
- 5. Ler o número do mês (1 – janeiro; 2 – fevereiro; ...; 12 – dezembro) e identificar em que trimestre o mês está incluído.



Exercícios

- 6. Mostrar os números de 100 até 200 variando de 10 em 10.
- 7. Mostrar os múltiplos positivos de 7 menores que 100.
- 8. Calcular e mostrar a soma dos termos da série: 6, 9, 12, 15, ..., 60.
- 9. Calcular e mostrar a soma dos termos da série: 7, 9, 11, 13, ..., 99.
- 10. Ler um conjunto de valores inteiros positivos e determinar a quantidade de números pares e ímpares digitados.



Exercícios

- 11. Ler dois valores e calcular o MDC e o MMC entre eles.
- 12. Calcular o valor de S, dado abaixo

$$S = \frac{1}{1} + \frac{3}{2} + \frac{5}{3} + \frac{7}{4} + \dots + \frac{99}{50}$$

- 13. Imprimir os N primeiros termos da série de Fibonacci, dada por: 1, 1, 2, 3, 5, 8, 13, ..., ou seja, a série inicia com dois valores iguais a um e cada termo subsequente é dado pela soma dos dois anteriores.
- 14. Calcular o valor de S, dado abaixo:
 - $S = 10 - 15 + 20 - 25 + \dots + 100$
- 15. Ler um número e verificar se ele é ou não um número primo.



Referencias Bibliográficas

- Introduction to C# Programming with Microsoft .Net
 - Microsoft Official Course 2609A
- Microsoft Visual C# 2005 - Passo a passo
 - John Sharp, Bookman, 2007
- Microsoft Asp.Net – Passo a passo
 - George Sheperd, Bookman, 2007
- Programação OO com C#
 - José Antônio da Cunha – Cefet-RN
- The new language for Microsoft .Net
 - Prof. Dr. H. Mössenböck - University of Linz, Institute for System Software, 2004