

Fundamentos de C#

Conceitos Iniciais de C#



Tópicos Abordados



- Estrutura de uma aplicação C#
- Variáveis
 - Declaração
 - Inicialização
 - Mudança de valor
- Constantes
- Tipos de Dados
 - Conversão de tipos de dados
 - Casting implícito
 - Casting explícito

Tópicos Abordados



- Operadores
 - Aritméticos
 - Comparação
 - Lógicos
- Estruturas de controle
 - **if-else**
 - **switch-case**
- Estruturas de repetição
 - **while**
 - **do-while**
 - **for**
 - **foreach**
 - Controlando iterações com **break** e **continue**

Estrutura de uma Aplicação C#

Softblue

- Exemplo de uma aplicação simples desenvolvida em C#

The diagram shows a code snippet for `MyFirstApp.cs` with the following structure and annotations:

```

namespace Softblue
{
    class MyCSharpApp
    {
        /*
         * Este é o ponto de entrada da aplicação.
         * Quando ela inicia, Main() é chamado.
         */
        static void Main()
        {
            // Exibe uma mensagem na tela.
            System.Console.WriteLine("Hello, C#");
        }
    }
}

```

Annotations in the diagram:

- Namespace:** Points to `namespace Softblue`.
- Classe:** Points to `class MyCSharpApp`.
- Arquivo com extensão .cs:** Points to the filename `MyFirstApp.cs`.
- Case Sensitive:** Points to the `namespace` keyword.
- Método Main():** Points to `static void Main()`.
- Comentário:** Points to the multi-line comment block.
- Exibe mensagem:** Points to `System.Console.WriteLine("Hello, C#");`.
- Termina com ;:** Points to the semicolon at the end of the `Main` method.

Declaração de Variáveis

Softblue

- Variáveis devem possuir um tipo e um identificador (nome)

The diagram shows a code snippet with variable declarations:

```

int    anoNasc;
double peso;
char   sexo;
bool   canhoto;

```

Annotations in the diagram:

- Tipo de dado:** Points to the data types (`int`, `double`, `char`, `bool`).
- Identificador:** Points to the variable names (`anoNasc`, `peso`, `sexo`, `canhoto`).

- Declaração de múltiplas variáveis

The diagram shows a code snippet for multiple variable declaration:

```

int n1, n2, n3;

```

Annotation in the diagram:

- 3 variáveis do tipo int:** Points to the entire declaration `int n1, n2, n3;`.

Regras na Declaração de Variáveis

Softblue


- Os identificadores de variáveis devem seguir algumas regras
 - O primeiro caractere deve ser uma letra ou `'_'`
 - A partir do segundo caractere, podem ser usados
 - Letras
 - Números
 - `'_'`
 - Se o identificador for igual a uma palavra-chave do C#, ele deve iniciar com `'@'`

Regras na Declaração de Variáveis



| | | | |
|--------|-------|-------|-------|
| abc; | OK | 1abc; | ERRO! |
| _abc; | OK | \$abc | ERRO! |
| ab1c1; | OK | abc# | ERRO! |
| _a_b_c | OK | a-b | ERRO! |
| class | ERRO! | int | ERRO! |
| @class | OK | @int | OK |

Inicialização de Variáveis



- Atribuição (operador '=')

```
anoNasc = 1980;
peso = 65.7;
sexo = 'M';
```

- Declaração e inicialização simultânea


```
double altura = 1.8;
```

- Declaração e inicialização múltipla

```
int n1 = 10, n2 = 20, n3 = 30;
```

- Todas as variáveis devem ser inicializadas antes de serem utilizadas
 - Erro de compilação


Alteração do Valor de Variáveis



- Outros exemplos de uso de variáveis

| | |
|--|-------------------|
| int contador = 20; int novoContador = contador + 1; | novoContador = 21 |
| int x = 15; x = x + 1; | x = 16 |
| int y = x + x - 10; | y = 22 |

Constantes




- Uma constante é uma variável que não pode ter seu valor alterado após a inicialização
 - Auxiliam o entendimento do código
 - Evitam erros na programação
- O modificador **const** é utilizado

```
const int Taxa = 10;
```

```
Taxa = 20;
```


Erro de compilação

Tipos de Dados



- O C# possui uma série de tipos de dados
 - Todos eles tem um mapeamento no CTS da plataforma .NET
- Os tipos de dados estão divididos em
 - Value Types
 - Reference Types
- Esta divisão indica como os dados são armazenados na memória

Value Types



- Números inteiros

Aderem ao CLS

sbyte

byte

short

ushort


int

uint

long

Ulong

| C# | CTS | Descrição |
|--------------|---------------------|---------------------------|
| sbyte | System.SByte | 8 bits, com sinal |
| byte | System.Byte | 8 bits, sem sinal |
| short | System.Int16 | 16 bits, com sinal |
| ushort | System.UInt16 | 16 bits, sem sinal |
| int | System.Int32 | 32 bits, com sinal |
| uint | System.UInt32 | 32 bits, sem sinal |
| long | System.Int64 | 64 bits, com sinal |
| Ulong | System.UInt64 | 64 bits, sem sinal |

Value Types



- Números decimais

| | C# | CTS | Descrição |
|---------------|---------|----------------|--------------------------|
| Aderem ao CLS | float | System.Single | 32 bits, ponto flutuante |
| | double | System.Double | 64 bits, ponto flutuante |
| | decimal | System.Decimal | 128 bits, alta precisão |

Uso do tipo **decimal** pode ocasionar queda de performance


- Booleano

| | C# | CTS | Descrição |
|--------------|------|----------------|---------------------|
| Adere ao CLS | bool | System.Boolean | verdadeiro ou falso |

Value Types


- Caractere

| | C# | CTS | Descrição |
|--------------|------|-------------|----------------------|
| Adere ao CLS | char | System.Char | 16 bits, 1 caractere |

Exemplos de Uso dos Value Types


```
int x = 90;
```

90 é um número inteiro, portanto pode ser atribuído a uma variável do tipo **int**

```
System.Int32 x = 90;
```

Idêntico ao exemplo anterior, mas utilizando o tipo CTS **System.Int32**

```
bool x = true;
```

true é um valor booleano, portanto pode ser atribuído a uma variável do tipo **bool**

```
double x = 2.5;
```

2.5 é um valor do tipo **double**, portanto pode ser atribuído a uma variável **double**

```
char x = 'A';
```

'A' é um caractere, portanto pode ser atribuído a uma variável do tipo **char**

```
short x = 90;
```

90 é um número inteiro, portanto pode ser atribuído a uma variável do tipo **short**

Ambiguidade de Tipos

É possível utilizar os qualificadores com letras maiúsculas ou minúsculas

Conversão de Tipos de Dados

- Algumas vezes precisamos atribuir um valor de um tipo a uma variável de outro tipo

- A conversão pode ser feita
 - De um tipo “menor” para um tipo “maior”
 - Ex: **int** para **double**
 - De um tipo “maior” para um tipo “menor”
 - Ex: **long** para **int**
- Este processo é chamado de **casting**

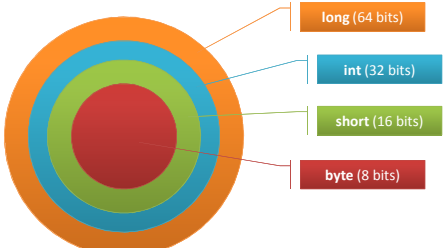
Casting Implícito

- O compilador cuida de todo o processo de conversão
- Ocorre quando é preciso converter de um tipo “menor” para um tipo “maior”
- Não ocorre perda de informação

Casting Implícito

Softblue

- Exemplos de casting implícito



- Existem outros exemplos na linguagem C#

Casting Explícito

Softblue

- O programador é responsável pela conversão dos dados
- Necessário quando é preciso converter de um tipo “maior” para um tipo “menor”
- Pode ocorrer perda de informação

```
double d = 100.0;
int i = d;
```

➔

```
double d = 100.0;
int i = (int)d;
```

Casting Explícito

Softblue

- Exemplos

```
int i = 10;
byte b = (byte)i;
```

```
double d = 140.32;
float f = (float)d;
```

O valor de **i** (10) é convertido para **byte** e armazenado em **b**

O valor de **d** (140,32) é convertido para **float** e armazenado em **f**

- Cuidado com o casting explícito

```
int i = 55000;
short s = (short)i;
```

```
int n = (int)3.5;
```

O resultado é **s = -10536**.
O número 55000 é muito grande para ser armazenado dentro de uma variável **short**.


O resultado é **n1 = 3**.
Como o **int** não armazena a parte decimal, ela é perdida.

Operadores Aritméticos



| Operador | Descrição |
|----------|---------------|
| + | Soma |
| - | Subtração |
| * | Multiplicação |
| / | Divisão |
| % | Módulo |

Operadores de Comparação



| Operador | Descrição |
|----------|------------------|
| == | Igual |
| != | Diferente |
| < | Menor que |
| > | Maior que |
| <= | Menor ou igual a |
| >= | Maior ou igual a |

Operadores Lógicos



| Operador | Descrição |
|----------|-----------|
| ! | Negação |
| | OU |
| && | E |

Outros Operadores Importantes



| Operador | Descrição | Exemplo |
|----------|---|---------|
| ++ | Incremento | x++ |
| -- | Decremento | x-- |
| += | Soma com valor e atribui o resultado à própria variável | x += 2 |
| -= | Subtrai do valor e atribui o resultado à própria variável | x -= 5 |
| *= | Multiplca pelo o valor e atribui o resultado à própria variável | x *= 3 |
| /= | Divide pelo valor e atribui o resultado à própria variável | x /= 4 |

Prioridades Entre Operadores




| Tipo de Operador | Operadores |
|-------------------------|------------------------|
| Parênteses | () |
| Unário | ! casting |
| Multiplicação / Divisão | * / % |
| Soma / Subtração | + - |
| Relacional | < > <= >= is as |
| Comparação | == != |
| AND | && |
| OR | |
| Atribuição | = += -= *= /= |

Para expressões complexas, use parênteses para especificar prioridade

Estrutura de Controle 'if-else'



Sintaxe básica

```
if (<condição_booleana>)
{
    <código_se_condição_verdadeira>;
}
```

Opcionalmente, pode existir uma cláusula else

```
if (<condição_booleana>)
{
    <código_se_condição_verdadeira>;
}
else
{
    <código_se_condição_falsa>;
}
```

Estrutura de Controle 'if-else'



- A condição booleana pode ser qualquer expressão lógica
- O resultado deve ser **true** ou **false**

```
int x = 50;

if (x > 30)
{
    Console.WriteLine("Número maior que 30");
}
else
{
    Console.WriteLine("Número menor que 30");
}
```

Estrutura de Controle 'if-else'



- Para blocos compostos por uma linha, não é necessário usar { e }

```
int x = 20, y;

if (x > 10)
    y = x * 2;
else
    y = x * 3;
```

- É preciso ter cuidado com essa sintaxe

```
int x = 20, y, z;

if (x > 10)
    y = x * 2;
    z = 1;
```

A chamada **z = 1** será executada sempre

Estrutura de Controle 'if-else'



- Outra possibilidade é utilizar o **operador ternário** em substituição ao *if-else*

```
int x = 50;
int y;

if (x > 30)
{
    y = 1;
}
else
{
    y = -1;
}
```

```
int x = 50;

int y = x > 30 ? 1 : -1;
```

Resultado, se verdadeiro

Resultado, se falso

Estrutura de Controle 'switch-case'



- A estrutura **switch-case** funciona de forma semelhante ao **if-else**

```
int i = 1;
switch (i)
{
    case 1:
        Console.WriteLine("Valor = 1");
        break;
    case 2:
        Console.WriteLine("Valor = 2");
        break;
    default:
        Console.WriteLine("Valor não reconhecido");
}
```

Não pode ser uma variável

O break separa os blocos

O default é similar ao else

Estrutura de Controle 'switch-case'



- Se um bloco **case** for vazio, o **break** não precisa ser especificado

```
int i = 1;
switch (i)
{
    case 1:
    case 2:
    case 3:
        Console.WriteLine("Valor é 1, 2 ou 3");
        break;
    default:
        Console.WriteLine("Valor não reconhecido");
        break;
}
```

O bloco será executado se i for 1, 2 ou 3

Estrutura de Repetição 'while'



- Repete determinado código enquanto uma condição booleana for verdadeira
- A condição é testada no início do bloco

```
int idade = 15;
while (idade < 18)
{
    Console.WriteLine(idade);
    idade = idade + 1;
}
```

Os valores impressos serão 15, 16 e 17

- Se a condição for falsa já no primeiro teste, o bloco não é executado

```
int idade = 19;
while (idade < 18)
{
    //...
}
```

O bloco nunca será executado

Estrutura de Repetição 'do-while'

- Semelhante ao **while**
 - Repete o código enquanto uma condição booleana for verdadeira
- A condição é testada no final do bloco

```
int contador = 10;
do
{
    Console.WriteLine(contador);
    contador = contador + 1;
}while (contador < 15);
```

Os valores impressos são 10, 11, 12, 13 e 14

Como o teste da condição é feito no final, o bloco sempre será executado pelo menos uma vez

Estrutura de Repetição 'for'

- Repete a execução de determinado código enquanto uma condição for verdadeira
- Permite declarar uma ou mais variáveis

```
for (<inicializador>; <condição_booleana>; <iterador>)
{
    <código_repetido_enquanto_a_condição_for_verdadeira>
}
```

Executa por primeiro

É testada antes de cada iteração

Executa ao final de cada iteração

Estrutura de Repetição 'for'

```
for (int i = 0; i < 10; i++)
{
    Console.WriteLine(i);
}
```

Cria a variável i e inicializa com 0

Testa a condição i < 10

Incrementa o valor de i

Os valores impressos são 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9

```
for (;;)
{
    Console.WriteLine("Loop infinito");
}
```

Nenhuma das informações do for é obrigatória

Estrutura de Repetição 'foreach'

- Permite iterar sobre todos os elementos de uma coleção de dados

```
int[] array = { 5, 2, 3, 1, 7 };  
  
foreach (int i in array)  
{  
    Console.WriteLine(i);  
}
```

Cada elemento de array é armazenado em i

Controlando Iterações

- As palavras-chave **break** e **continue** podem ser usadas para controlar iterações em estruturas de repetição
 - **break** pode ser usado também no **switch-case**
- **break**
 - Força a saída de um loop
- **continue**
 - Força novo teste da condição

Controlando Iterações com 'break'

- Exemplo de uso do **break**

```
for (int i = 0; i < 10; i++)  
{  
    if (i == 5)  
    {  
        break;  
    }  
    Console.WriteLine(i);  
}  
Console.WriteLine("FIM");
```

O break termina a execução do loop

Resultado:
0, 1, 2, 3, 4, FIM

Controlando Iterações com 'continue'

- Exemplo de uso do **continue**

```
for (int i = 0; i < 10; i++)  
{  
    if (i == 5)  
    {  
        continue;  
    }  
    Console.WriteLine(i);  
}  
  
Console.WriteLine("FIM");
```

O **continue** termina a execução da iteração corrente do loop

Resultado:
0, 1, 2, 3, 4, 6, 7, 8, 9, FIM

Qual estrutura escolher?

- Estruturas de controle
 - O **switch-case** tem uma série de limitações ao ser comparado com o **if-else**
 - Na prática, o **switch-case** é muito pouco utilizado
- Estruturas de repetição
 - O **for** é utilizado quando o número de iterações é predeterminado
 - O **foreach** é utilizado para iterar sobre todos os elementos de uma coleção de dados
 - Entre o **while** e o **do-while**, o **do-while** deve ser escolhido quando o bloco precisa ser executado pelo menos uma vez

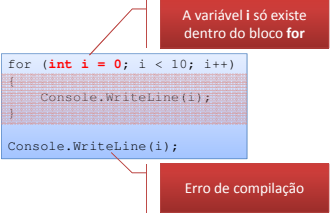
Escopos de Variáveis

- Variáveis declaradas dentro de um bloco existem apenas dentro deste bloco

```
int x = 10;  
if (x == 10)  
{  
    int y = 5;  
    Console.WriteLine(y);  
}  
else  
{  
    int y = 10;  
    Console.WriteLine(y);  
}
```

Uma variável **y** não "enxerga" a outra

Escopos de Variáveis



```
for (int i = 0; i < 10; i++)  
{  
    Console.WriteLine(i);  
}  
Console.WriteLine(i);
```

A variável i só existe dentro do bloco for

Erro de compilação



Softblue
