* **A sintaxe do while e for;**
* **O operador +=;**
* **O operador ++;**
* **Laços aninhados;**
* **A funcionalidade do break.**

**C# PARTE 1**

## Qual é o grande benefício de uma Maquina Virtual .NET *(CLR - common language runtime)*?

* Alternativa correta
* **Executar o código independente do sistema operacional.**
* No mundo .NET você terá sempre o mesmo MSIL que será executado pela Maquina Virtual CLR, independente do sistema operacional. Assim não é preciso rescrever ou adaptar o código para rodar em um outro sistema operacional ou integrar em um projeto escrito com outra linguagem .NET.

O que aprendemos?

**Nessa aula introdutória já aprendemos vários assuntos fundamentais sobre C#** e, em especial, o mundo .NET.

* Toda aplicação .NET é escrita em uma linguagem compilada para o código intermediário MSIL (Microsoft intermediate language);
* O .NET Framework é uma biblioteca utilizada pelas aplicações .NET;
* Uma aplicação .NET é compilada para o MSIL;
* O código MSIL é executado pela máquina virtual do .NET, a CLR (common language runtime);
* O just-in-time compiler é uma parte da CLR que transforma MSIL em código de máquina, apenas em tempo de execução.

Nós fomos cautelosos ao definir o tipo de número inteiro (int) ou flutuante (double).

* Double - admite valores com ponto e também admite números inteiros;
* Int - só admite números inteiros.
* o tipo long não é tão utilizado justamente por ocupar mais espaço na memória. Por isso, o tipo int ainda é o mais utilizado. 32 bits
* short é uma variavel de 16 bits suporta ate 16000;
* Qual a diferença do tipo float para o tipo double? O tipo float tem uma precisão curta de números suportados nas casas decimais - ao contrário do double, que suporta uma precisão maior. Se prestarmos atenção, o valor 150000 está sinalizado em vermelho, porque ele não foi aceito pelo compilador dentro da variável short. Se altermos o valor para 15000, o editor vai aceitá-lo.
* Também teremos problema com o valor da altura, utilizando o float. Isto ocorre, porque quando trabalhamos com número decimal é estranho usarmos o float, devido a falta de suporte para casas decimais.

Como vimos nos vídeos, quando tentamos colocar um valor inteiro em uma variável do tipo double, o C# não mostra erro. Porém, quando tentamos colocar um double numa variável no tipo inteiro, temos um erro de compilação.

Esta propriedade se dá por que o C# faz conversão implícita de um tipo menor para os tipos "maiores". De inteiro para double, por exemplo. O contrário não é verdade, pois existe perda de dados quando é feita a conversão. Isso acarreta um "type mismatch", mostrando que esta instrução é de tipos incompatíveis.

Para fazer uma conversão onde pode haver perda de informações, é necessário fazer um type casting. Veja a instrução abaixo.

double idade = (int) 30.0;

No caso acima, está explícito que será feito o cast de double para inteiro. Veja como funciona o cast implícito e explícito na tabela abaixo.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PARA / DE** | **byte** | **short** | **char** | **int** | **long** | **float** | **double** |
| byte | ---- | *Impl.* | (char) | *Impl.* | *Impl.* | *Impl.* | *Impl.* |
| short | (byte) | ---- | (char) | *Impl.* | *Impl.* | *Impl.* | *Impl.* |
| char | (byte) | (short) | ---- | *Impl.* | *Impl.* | *Impl.* | *Impl.* |
| int | (byte) | (short) | (char) | ---- | *Impl.* | *Impl.* | *Impl.* |
| long | (byte) | (short) | (char) | (int) | ---- | *Impl.* | *Impl.* |
| float | (byte) | (short) | (char) | (int) | (long) | ---- | *Impl.* |
| double | (byte) | (short) | (char) | (int) | (long) | (float) | ---- |

Para comparar cada tipo primitivo de forma mais clara, a tabela abaixo mostra qual o tamanho de cada um.

|  |  |
| --- | --- |
| **TIPO** | **TAMANHO** |
| boolean | 1 byte |
| byte | 1 byte |
| short | 2 bytes |
| char | 2 bytes |
| int | 4 bytes |
| float | 4 bytes |
| long | 8 bytes |
| double | 8 bytes |

Passamos pelos tipos numéricos do C#, em seguida, falaremos sobre **caracteres e textos**. Para isto, criaremos um novo projeto na aba de solução, e depois, o nomearemos como "5-CaracteresETextos".

Lembre-se de selecionar a opção "Aplicativo do Console[.NET Framework]".

Adicionaremos no código padrão, a linha com Console.WriteLine("Executando o projeto 5 - Caracteres e textos"). Observe que usaremos um tipo de variável especial: char, uma simplificação de "character".

namespace \_5\_CaracteresETextos

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Executando o projeto 5 - Caracteres e textos");

char primeiraLetra;

Console.ReadLine();

}

}

}

Quando trabalhamos com caracteres dentro da variável, no caso, colocaremos 'a'. Neste caso, inserimos a letra entre aspas simples, isto porque existe **diferenças entre um caractere e um texto**.

A linguagem C# **não aceita as aspas duplas**, se colocarmos aspas duplas, o conteúdo será interpretado como um texto, ou seja, uma string. Teremos que fazer algo com caractere, no caso, ele será um parâmetro de primeiraLetra.

namespace \_5\_CaracteresETextos

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Executando o projeto 5 - Caracteres e textos");

char primeiraLetra = 'a';

Console.WriteLine(primeiraLetra);

Console.ReadLine();

}

}

}

Ao executarmos o projeto 5, será exibida a letra "a".

Executando o projeto 5 - Caracteres e textos

a

Mas vale ressaltar que internamente, a variável char é representada como número, caberia usar um número de 16 bits. Por isso, podemos pegar a primeiraLetra e colocar dentro dela um número.

Apesar do char ser, internamente, um número na memória do computador, se tentarmos atribuir o valor 65 para primeiraLetra, o C# não vai aceitar. Isto, porque quando escrevemos o número, não temos um caractere. No caso, temos um inteiro de 32 bits, porém, **a variável char é bem menor**. Como falamos anteriormente, nela, cabe apenas 16 bits.

Se estamos fazendo isso, precisamos fazer o casting, assim converteremos de um tipo para outro, da mesma maneira como fizemos com double e o Integer, nós converteremos o valor 65 para caractere, adicionando (char).

namespace \_5\_CaracteresETextos

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Executando o projeto 5 - Caracteres e textos");

char primeiraLetra = 'a';

Console.WriteLine(primeiraLetra);

primeiraLetra = (char)65;

Console.WriteLine(primeiraLetra);

Console.ReadLine();

}

}

}

Desta vez, veremos uma pequena diferença no retorno. O valor 65 vai ser convertido para A.

Executando o projeto 5 - Caracteres e textos

a

A

Note que a letra "A" foi exibida maiúscula. Será que com isso é possível fazer manipulaçãoes numéricas com esses caracteres? Vamos testar isso, atribuindo outro valor para primeiraLetra, escrevendo um expressão que use a variável. No caso, vamos adicionar +1.

namespace \_5\_CaracteresETextos

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Executando o projeto 5 - Caracteres e textos");

char primeiraLetra = 'a';

Console.WriteLine(primeiraLetra);

primeiraLetra = (char)65;

Console.WriteLine(primeiraLetra);

primeiraLetra = primeiraLetra + 1;

Console.ReadLine();

}

}

}

No entanto, o C# não vai ficar feliz, porque estamos trabalhando com um número inteiro, seguido de char e o resultado do cálculo vai ser um tipo com mais alcance, o int. E inteiro não cabe em um char.

Se temos todos esses elementos, colocaremos a expressão (primeiraLetra + 1) entre parênteses, porque queremos a precedência do resultado para, em seguida, fazer um *casting* criando char.

namespace \_5\_CaracteresETextos

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Executando o projeto 5 - Caracteres e textos");

char primeiraLetra = 'a';

Console.WriteLine(primeiraLetra);

primeiraLetra = (char)65;

Console.WriteLine(primeiraLetra);

primeiraLetra = (char)(primeiraLetra + 1);

Console.WriteLine(primeiraLetra);

Console.ReadLine();

}

}

}

Desta vez, o C# vai realizar a soma, e depois, fará o casting com o resultado - que será um número inteiro. Talvez, você esteja questionando o uso do parênteses na expressão. No entanto, se ele não for adicionado, a variável primeiraLetra será convertida, apesar de já ser char. E o valor +1 continuará sem alteração.

Se executarmos o código, veremos a letra "B" exibida na tela.

Executando o projeto 5 - Caracteres e textos

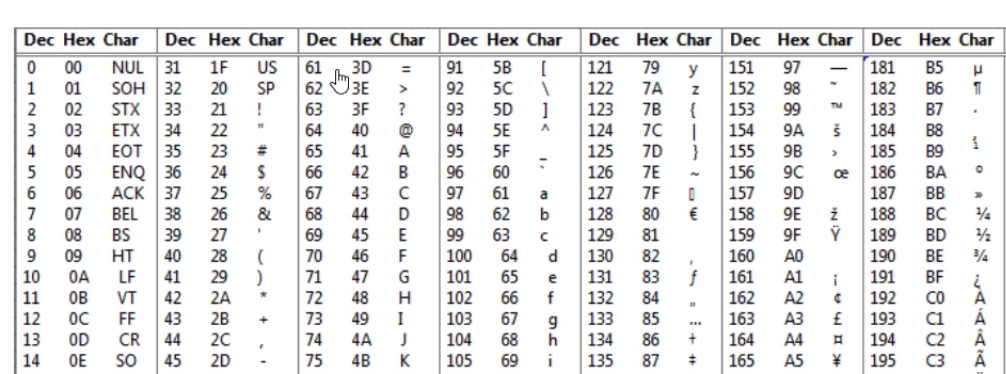
a

A

B

Nós vimos que 65 pode ser a, A e B, no entanto, esses retornos são definições existentes que a linguagem C# segue. Isto respeita a ***Ascii table***(Tabela de *American Standard Code*). Trata-se de uma tabela com vários caracteres - entre eles encontraremos não apenas letras, mas também números, cedilha, e caracteres especiais.

Temos também a representação númerica, em número decimal. Na tabela, veremos que 61 é o sinal de igual (=).



Em seguida, vamos **parar a execução da aplicação**. Ao retornarmos ao código, alteraremos o valor 65 para 61.

namespace \_5\_CaracteresETextos

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Executando o projeto 5 - Caracteres e textos");

char primeiraLetra = 'a';

Console.WriteLine(primeiraLetra);

primeiraLetra = (char)61;

Console.WriteLine(primeiraLetra);

primeiraLetra = (char)(primeiraLetra + 1);

Console.WriteLine(primeiraLetra);

Console.ReadLine();

}

}

}

Quando iniciarmos a aplicação, veremos que o operador = é impresso na tela.

Executando o projeto 5 - Caracteres e textos

a

=

>

Por último, o 61 foi somado com +1, o que resultou em > (equivalente ao 62).

Mas você pode ainda não entender o porquê de trabalharmos com esses caracteres. Então vale lembrar que **a linguagem C# é utilizada em diferentes nichos**, desde aplicações do mercado financeiro, como do governo, até internet das coisas. Quando temos uma placa pequena, sem muitos recursos, trabalhamos mais próximo do hardware.

A variável char será importante nesses casos, por exemplo, quando formos mandar um caractere para um display bem pequeno. Para entender as diferentes possibilidades de uso, nós encontraremos esses recursos. Mas dependendo do mundo que você trabalha, não faz sentido.

Apresentamos como trabalhar com caracteres, veremos como usaremos textos no C#. No código, adicionamos o texto entre aspas duplas incluindo juntamente com WriteLine(). Vamos atribuir uma variável do tipo string.

Neste caso, utilizaremos as aspas duplas.

Junto à variável, vamos incluir o texto Alura Cursos de tecnologia. O próximo passo será executar Console.WriteLine(), com titulo.

namespace \_5\_CaracteresETextos

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Executando o projeto 5 - Caracteres e textos");

char primeiraLetra = 'a';

Console.WriteLine(primeiraLetra);

primeiraLetra = (char)61;

Console.WriteLine(primeiraLetra);

primeiraLetra = (char)(primeiraLetra + 1);

Console.WriteLine(primeiraLetra);

string titulo = "Alura Cursos de tecnologia";

Console.WriteLine(titulo);

Console.ReadLine();

}

}

}

Se executarmos a aplicação, veremos que o texto foi impresso na tela.

Executando o projeto 5 - Caracteres e textos

a

=

Alura Cursos de tecnologia

Fecharemos a aplicação, e como já fizemos anteriormente, vamos manipular a string. Nós podemos, por exemplo, usar o operador de soma +, seguido de um número que será convertido e concatenado com o texto.

//...

primeiraLetra = (char)(primeiraLetra + 1);

Console.WriteLine(primeiraLetra);

string titulo = "Alura Cursos de tecnologia " + 2020;

Console.WriteLine(titulo);

Console.ReadLine();

}

}

}

Se executarmos a aplicação, teremos o resultado esperado.

Executando o projeto 5 - Caracteres e textos

a

=

Alura Cursos de tecnologia 2020

Repare que eu deixei um espaço após Alura Cursos de tecnologia, assim o texto foi impresso corretamente. Inclusive, vale lembrar que **espaço também é um caractere**, com o qual também podemos trabalhar no código, adicionado **entre as aspas simples**.

namespace \_5\_CaracteresETextos

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Executando o projeto 5 - Caracteres e textos");

char primeiraLetra = ' ';

Console.WriteLine(primeiraLetra);

//...

Como vimos, caracteres especiais podem ser incluídos também. Mas será que podemos ter um caractere vazio? Não, se trabalhamos com caracteres, teremos que necessariamente adicionar pelo menos um **entre as aspas simples**.

No entanto, podemos criar uma variável vazia do tipo string, basta não adicionarmos um valor entre as aspas. Desta vez, o C# vai rodar sem problemas.

//...

string titulo = "Alura Cursos de tecnologia " + 2020;

Console.WriteLine(titulo);

string vazia = "";

Console.ReadLine();

}

}

}

Vamos agora apagar string vazia, criada apenas para ilustrar o caso, e voltaremos a escrever primeiraLetra, adicionando o caractere espaço.

//...

Console.WriteLine("Executando o projeto 5 - Caracteres e textos");

//character

char primeiraLetra = ' ';

Console.WriteLine(primeiraLetra);

primeiraLetra = (char)61;

Console.WriteLine(primeiraLetra);

primeiraLetra = (char)(primeiraLetra + 1);

Console.WriteLine(primeiraLetra);

string titulo = "Alura Cursos de tecnologia " + 2020;

Console.WriteLine(titulo);

Console.ReadLine();

}

}

}

Existe ainda a situação na qual queremos listar um texto, por exemplo, se quero fazer uma listagem dos cursos de programação da Alura. Criaremos a string cursosProgramacao e, depois, incluiremos os itens da lista, cada um adicionado entre aspas duplas.

//...

primeiraLetra = (char)61;

Console.WriteLine(primeiraLetra);

primeiraLetra = (char)(primeiraLetra + 1);

Console.WriteLine(primeiraLetra);

string titulo = "Alura Cursos de tecnologia " + 2020;

string cursosProgramacao = " - .NET" +

" - Java" +

" - Javascript";

Console.WriteLine(titulo);

Console.WriteLine(cursosProgramacao);

Console.ReadLine();

}

}

}

No entanto, quando separamos os itens em linhas diferentes, o VS Code não só quebrará o código, ele adicionará o operador de soma (+) e já vai abrir a próxima dupla de aspas.

Isso é o suficiente para montar o texto com várias linhas, para representar essa lista de cursos. Veremos isso graças à linha Console.WriteLine(cursosProgramacao).

Ao clicarmos em "iniciar", teremos o seguinte resultado:

Executando o projeto 5 - Caracteres e textos

=

>

Alura Cursos de tecnologia 2020

- .NET - Java - Javascript

Em vez de quebrar as linhas, os cursos foram exibidos lado a lado, porque os operadores de soma foram ignorados. Mas queríamos quebrar a linha.

A questão é que não podemos quebrar a linha de uma string. Também não podemos criar a listagem sem adicionar as aspas entre os cursos - o compilador rejeitaria nosso código. A menos que seja adicionado o caractere @ e removido o sinal +, neste caso, o código passará a ser válido.

//...

primeiraLetra = (char)(primeiraLetra + 1);

Console.WriteLine(primeiraLetra);

string titulo = "Alura Cursos de tecnologia " + 2020;

string cursosProgramacao = @" - .NET

- Java

- Javascript";

Console.WriteLine(titulo);

Console.WriteLine(cursosProgramacao);

Console.ReadLine();

}

}

}

Agora é possível criar uma string única com várias linhas, sem concatenação. Mas a impressão não ficará perfeita.

Executando o projeto 5 - Caracteres e textos

=

>

Alura Cursos de tecnologia 2020

- .NET

- Java

- Javascript

No momento em que usamos @, o compilador vai considerar tudo entre as aspas e o texto, inclusive o espaço. Para solucionar o tema, posso remover os espaços entre os cursos.

//...

primeiraLetra = (char)(primeiraLetra + 1);

Console.WriteLine(primeiraLetra);

string titulo = "Alura Cursos de tecnologia " + 2020;

string cursosProgramacao =

@" - .NET

- Java

- Javascript";

Console.WriteLine(titulo);

Console.WriteLine(cursosProgramacao);

Console.ReadLine();

}

}

}

E teremos a impressão desejada:

Executando o projeto 5 - Caracteres e textos

=

>

Alura Cursos de tecnologia 2020

- .NET

- Java

- Javascript

Os cursos foram listados no mesmo bloco.

**IF**

**Nós vamos criar uma variável do tipo inteiro com o nome idadeJoao, que é igual 16. Agora usaremos uma condição, para verificar se o João tem mais ou menos que 18 anos. Desta forma, verificaremos se a idade do João é maior ou não. Logo, executaremos um código caso isso seja verdadeiro.**

**Para fazer a representação no C#, adicionaremos if() e entre parênteses colocaremos a expressão que queremos avaliar se a idade do João atende a condição de ser maior que 18 anos. Se isso for verdade, executaremos um novo bloco de código.**

**namespace \_7\_Condicionais**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 7 - Condicionais");**

**int idadeJoao = 16;**

**if (idadeJoao > 18)**

**{**

**Console.WriteLine("João possui mais de 18 anos de idade. Pode entrar.");**

**}**

**Console.ReadLine();**

**}**

**}**

**}**

**Se a idade do João for maior do que 18, escreveremos na nossa console se o João tem mais de 18 anos de idade. Caso a resposta seja sim, ele pode entrar. Depois, clicaremos no botão "Iniciar" e será exibido na tela:**

**Executando projeto 7 - condicionais**

**Nenhuma mensagem foi impressa, porque idade do João é 16 e não atende a condição de ser maior que 18. Vamos parar a aplicação e alterar a idade do João, porém, se alterarmos para 18, ainda não veremos a mensagem exibida. Na verdade, devemos repensar essa condição.**

**A idade deve ser maior ou igual a 18, para isto, teremos que modificar o operador de comparação, nós vamos usar >=.**

**namespace \_7\_Condicionais**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 7 - Condicionais");**

**int idadeJoao = 18;**

**if (idadeJoao >= 18)**

**{**

**Console.WriteLine("João possui mais de 18 anos de idade. Pode entrar.");**

**}**

**Console.ReadLine();**

**}**

**}**

**}**

**Ao executarmos o código, veremos a mensagem exibida:**

**Executando projeto 7 - Condicionais**

**João possui mais de 18 anos de idade. Pode entrar.**

**Nós podemos fazer outras coisas também, por exemplo, se alterarmos a idade do João para um valor menor ou igual a 18, queremos que seja exibida uma imagem. Nós podemos criar um bloco de código que seja o contrário do caso no qual a idade seja maior, ou seja o "senão". Abriremos o bloco do else e adicionaremos a mensagem: João não possui mais de 18, não pode entrar.**

**namespace \_7\_Condicionais**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 7 - Condicionais");**

**int idadeJoao = 16;**

**if (idadeJoao >= 18)**

**{**

**Console.WriteLine("João possui mais de 18 anos de idade. Pode entrar.");**

**}**

**else**

**{**

**Console.WriteLine("João não possui mais de 18 anos, não pode entrar");**

**}**

**Console.ReadLine();**

**}**

**}**

**}**

**Agora, se executarmos o código, veremos que João não possui mais do que 18anos.**

**Executando projeto 7 - Condicionais**

**João não possui mais de 18 anos, não pode entrar**

**Esse código dentro do bloco do if() não foi executado pelo C#, porque a sua avaliação foi falsa e apenas o bloco de else foi executado pela aplicação. Nós podemos tornar isso mais interessante, nós podemos adicionar mais uma condição. Podemos proibir a entrada de menores de 18, mas permitiremos a entrada se eles estiverem acompanhados. Para isto, criaremos outra variável int, que receberá o nome quantidadePessoas definido com o valor 2.**

**//...**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 7 - Condicionais");**

**int idadeJoao = 16;**

**int quantidadePessoas = 2;**

**if (idadeJoao >= 18)**

**{**

**Console.WriteLine("João possui mais de 18 anos de idade. Pode entrar.");**

**}**

**else**

**{**

**Console.WriteLine("João não possui mais de 18 anos, não pode entrar");**

**}**

**Console.ReadLine();**

**}**

**}**

**}**

**Com o valor 2, indicamos que o João está acompanhado. Se João tem uma idade maior que 18, sabemos que ele pode entrar, caso contrário, verificaremos se ele está acompanhado com um novo if().**

**//...**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 7 - Condicionais");**

**int idadeJoao = 16;**

**int quantidadePessoas = 2;**

**if (idadeJoao >= 18)**

**{**

**Console.WriteLine("João possui mais de 18 anos de idade. Pode entrar.");**

**}**

**else**

**{**

**if(quantidadePessoas >= 2)**

**{**

**Console.WriteLine("João não possui mais de 18 anos, mas está acompanhado. Pode entrar");**

**}**

**else**

**{**

**Console.WriteLine("João não possui mais de 18 anos, não pode entrar");**

**}**

**}**

**Console.ReadLine();**

**}**

**}**

**}**

**Fique atento que quando abrimos um bloco de código no VS Code, o editor automaticamente vai incluir a chave de fechamento. Se clicarmos no botão "Iniciar" teremos o resultado esperado no console.**

**Executando projeto 7 - Condicionais**

**João não possui mais de 18 anos, mas está acompanhado. Pode entrar**

**Se analisarmos o código, entenderemos que a idade do João é 16, que não é maior ou igual a 18 e por isso, o bloco do if() não será executado. Mas a quantidade de pessoa é 2 (ou seja, ele está acompanhado) e o bloco executado é o if dentro do else. Já o segundo else não será executado, porque a condição foi satisfeita com apenas um bloco de código.**

**No entanto, observe que nosso código ficou desorganizado, e apesar da ajuda oferecida pelo VS ao abrirmos um bloco incluindo o fechamento com a chave e indicando a indentação correta, ele acaba ficando pouco ordenado - ainda que seja válido. Podemos mostrar um exemplo abaixo:**

**//...**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 7 - Condicionais");**

**int idadeJoao = 16;**

**int quantidadePessoas = 2;**

**if (idadeJoao >= 18)**

**{**

**Console.WriteLine("João possui mais de 18 anos de idade. Pode entrar.");**

**}**

**else**

**{**

**if(quantidadePessoas >= 2)**

**{**

**Console.WriteLine("João não possui mais de 18 anos, mas está acompanhado. Pode entrar");**

**}**

**else**

**{**

**Console.WriteLine("João não possui mais de 18 anos, não pode entrar");**

**}**

**}**

**Console.ReadLine();**

**}**

**}**

**}**

**O VS tem um recurso para formatação de texto, se clicarmos em "Editar > Avançado > Formatar documento", veremos que o código será indentado automaticamente pelo editor. Conseguiremos fazer o mesmo com o atalho "Control+E, D" - bastante usado no nosso cotidiano.**

**Desta forma, conseguimos deixar o nosso código organizado.**

**VARIAVEL BOOLEANA**

**Vamos analisar o código com mais atenção:**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 7 - Condicionais");**

**int idadeJoao = 16;**

**int quantidadePessoas = 2;**

**if (idadeJoao >= 18)**

**{**

**Console.WriteLine("João possui mais de 18 anos de idade. Pode entrar.");**

**}**

**else**

**{**

**if(quantidadePessoas >= 2)**

**{**

**Console.WriteLine("João não possui mais de 18 anos, mas está acompanhado. Pode entrar");**

**}**

**else**

**{**

**Console.WriteLine("João não possui mais de 18 anos, não pode entrar");**

**}**

**}**

**Console.ReadLine();**

**}**

**}**

**}**

**Se a idade do João for maior ou igual a 18, ele poderá entrar na festa. Outro caso que atende a condição é se ele estiver acompanhado, isto significa que a quantidade de pessoas deve ser maior ou igual a 2. Temos if - else - if - else, deixando o nosso código mais complexo, porque encadeamos muitos blocos dentro de outros. Existe uma forma de melhorarmos o código? Faremos isso utilizando operadores lógicos do C#.**

**O próximo passo será criar um projeto, clicaremos na nossa solução, depois em "Adicionar projeto". O novo projeto receberá o nome de 8-Condicionais2. Depois, vamos copiar o código do arquivo 7-Condicionais e, em seguida, iremos copiá-lo no outro. Desta forma, manteremos o histórico e poderemos comparar as versões para recordar o que foi aprendido durante o curso.**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 8 - Condicionais 2");**

**int idadeJoao = 16;**

**int quantidadePessoas = 2;**

**if (idadeJoao >= 18)**

**{**

**Console.WriteLine("João possui mais de 18 anos de idade. Pode entrar.");**

**}**

**else**

**{**

**if(quantidadePessoas >= 2)**

**{**

**Console.WriteLine("João não possui mais de 18 anos, mas está acompanhado. Pode entrar");**

**}**

**else**

**{**

**Console.WriteLine("João não possui mais de 18 anos, não pode entrar");**

**}**

**}**

**Console.ReadLine();**

**}**

**}**

**}**

**Nós avaliamos se João tem mais do que 18 anos para permitir que ele entre, ou embaixo, verificaremos se a quantidade de pessoas é maior ou igual a 2. Podemos simplificar esse trecho de código, resolvendo no primeiro if. A nova condição será idadeJoao que deve ser >= a 18 ou se quantidadePessoas for >= a 2.**

**Em algumas linguagens a palavra "ou" com or, a linguagem C# recebeu como herança de C, o uso do || (dois *pipes*). Pode ser pouco intuitivo, mas após adicionarmos as duas barras verticais, podemos incluir outra expressão que terá como retorno verdadeiro ou falso. No caso, adicionaremos quantidadePessoas >= 2.**

**Agora podemos remover o trecho que tem a mesma utilidade, depois da reorganização, o código ficará da seguinte maneira.**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 8 - Condicionais 2");**

**int idadeJoao = 16;**

**int quantidadePessoas = 2;**

**if (idadeJoao >= 18 || quantidadePessoas >= 2)**

**{**

**Console.WriteLine("João possui mais de 18 anos de idade. Pode entrar.");**

**}**

**else**

**{**

**Console.WriteLine("João não possui mais de 18 anos, mas está acompanhado. Pode entrar");**

**}**

**Console.ReadLine();**

**}**

**}**

**Agora, a atual mensagem já não faz mais sentido, porque não sabemos qual das duas condições foram atendidas. Então apenas informaremos se João pode ou não entrar.**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 8 - Condicionais 2");**

**int idadeJoao = 16;**

**int quantidadePessoas = 2;**

**if (idadeJoao >= 18 || quantidadePessoas >= 2)**

**{**

**Console.WriteLine("Pode entrar.");**

**}**

**else**

**{**

**Console.WriteLine("Não pode entrar");**

**}**

**Console.ReadLine();**

**}**

**}**

**Agora se clicarmos em "Iniciar", no console veremos a nova mensagem.**

**Executando projeto 8 - Condicionais 2**

**Pode entrar.**

**A idade do João é 16, então a condição não foi satisfeita, no entanto, a quantidade de pessoas é maior ou igual a dois, como estamos trabalhando com um operador lógico || (que significa "ou"), esta expressão me devolve verdadeiro.**

**Após pararmos a aplicação. Vamos fazer outro teste, imagine a situação em que só se pode entrar acompanhado. Teremos que trocar "ou" pelo "e", que no C# é escrito da seguinte forma: &&.**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 8 - Condicionais 2");**

**int idadeJoao = 16;**

**int quantidadePessoas = 2;**

**if (idadeJoao >= 18 && quantidadePessoas >= 2)**

**{**

**Console.WriteLine("Pode entrar.");**

**}**

**else**

**{**

**Console.WriteLine("Não pode entrar");**

**}**

**Console.ReadLine();**

**}**

**}**

**Se executarmos a aplicação, veremos a mensagem Não pode entrar. Apesar da quantidade de pessoas ser igual a 2, o valor da idade não atende a condição de ser maior ou igual a 18. Quando usamos &&, as duas condições precisam ser satisfeitas. Se trocarmos a idade do João para 18 e iniciarmos, poderemos entrar.**

**Executando projeto 8 - Condicionais 2**

**Pode entrar.**

**Agora as duas condições foram satisfeitas, vamos prestar atenção em algo: temos a variável quantidadePessoas porque queremos saber se o João está acompanhado ou não.**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 8 - Condicionais 2");**

**int idadeJoao = 18;**

**int quantidadePessoas = 2;**

**if (idadeJoao >= 18 && quantidadePessoas >= 2)**

**{**

**Console.WriteLine("Pode entrar.");**

**}**

**else**

**{**

**Console.WriteLine("Não pode entrar");**

**}**

**Console.ReadLine();**

**}**

**}**

**Nós ainda não falamos sobre um tipo de variável primitiva que guarda um valor true ou false, ou seja, guardar uma expressão que devolve valores verdadeiros ou falsos. Ela será definida como tipo bool e nomeada de acompanhado. Nesta variável, só poderemos atribuir dois valores: true ou false - verdadeiro ou falso.**

**A variável booleana acompanhado poderá receber a expressão utilizada anteriormente quantidadePessoas >= 2. Depois, substituiremos a expressão no if pelo nome da variável que será igual a true.**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 8 - Condicionais 2");**

**int idadeJoao = 18;**

**int quantidadePessoas = 2;**

**bool acompanhado = quantidadePessoas >= 2;**

**if (idadeJoao >= 18 && acompanhado == true)**

**{**

**Console.WriteLine("Pode entrar.");**

**}**

**else**

**{**

**Console.WriteLine("Não pode entrar");**

**}**

**Console.ReadLine();**

**}**

**}**

**O operador que usamos para comparar igualdade será ==. Se usarmos apenas um operador, estaremos atribuindo um valor (aquele da direita está dentro da variável à esquerda), mas se usarmos dois =, o operador será usado para comparação. Se executarmos a operação após as modificações, verificaremos que tudo funciona corretamente.**

**Nós podemos guardar um literal dentro da variável, mostraremos como fazer isso a seguir. Vamos comentar a linha onde declaramos a variável quantidadePessoas e , agora, acompanhado será verdadeiro ou falso. Agora vamos trabalhar com acompanhado como true.**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 8 - Condicionais 2");**

**int idadeJoao = 18;**

**//int quantidadePessoas = 2;**

**//bool acompanhado = quantidadePessoas >= 2;**

**bool acompanhado = true;**

**if (idadeJoao >= 18 && acompanhado == true)**

**{**

**Console.WriteLine("Pode entrar.");**

**}**

**else**

**{**

**Console.WriteLine("Não pode entrar");**

**}**

**Console.ReadLine();**

**}**

**}**

**Se executarmos o código, veremos que ele tem o mesmo comportamento. Da mesma forma que colocamos true, que é verdadeiro, podemos defini-la como false.**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 8 - Condicionais 2");**

**int idadeJoao = 18;**

**//int quantidadePessoas = 2;**

**//bool acompanhado = quantidadePessoas >= 2;**

**bool acompanhado = false;**

**//...**

**Após clicarmos em "Iniciar", desta vez, o retorno será Não pode entrar, porque acompanhado é false e não atende a condição de ser true.**

**Seguindo com o assunto de condicionais e a variável *booleana* , abordaremos outro tópico. Para isto, manteremos a organização e criaremos outro projeto que receberá o nome 9-Escopo. Novamente, copiaremos o código do arquivo anterior e iremos renomeá-lo.**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 9 - Escopo");**

**int idadeJoao = 18;**

**bool acompanhado = false;**

**if (idadeJoao >= 18 && acompanhado == true)**

**{**

**Console.WriteLine("Pode entrar.");**

**}**

**else**

**{**

**Console.WriteLine("Não pode entrar");**

**}**

**Console.ReadLine();**

**}**

**}**

**O novo assunto será "Escopo", observe que removemos as linhas comentadas. Além de sabermos que o João "pode entrar" quando executamos o primeiro bloco de código, podemos incluir ainda uma mensagem informando se ele está acompanhado ou não. Para isto, criaremos uma variável do tipo string, onde guardaremos o texto.**

**Um truque é adicionarmos if seguido de else para verificar acompanhado.**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 9 - Escopo");**

**int idadeJoao = 18;**

**bool acompanhado = false;**

**if (acompanhado == true)**

**{**

**}**

**//...**

**Ao incluirmos o if(acompanhado == true) veremos que ele ficará colorido com o fundo amarelo. Isto ocorreu porque escolhemos este if e pressionamos a tecla "tab", automaticamente, o VS coloca os cursos dentro do parênteses e pinta o que foi escrito. Se pressionarmos "Enter", o cursor vira direto para dentro do bloco e o trecho não estará mais sinalizado. Trata-se de um recurso valioso, quando começamos a escrever intensamente.**

**Se o João estiver acompanhado, adicionaremos no bloco a variável mensagemAdicional, esta receberá a mensagem João está acompanhado!, e abaixo, incluiremos o bloco referente ao else.**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 9 - Escopo");**

**int idadeJoao = 18;**

**bool acompanhado = false;**

**if (acompanhado == true)**

**{**

**mensagemAdicional = "João está acompanhado!";**

**}**

**else**

**{**

**mensagemAdicional = "João não está acompanhado";**

**}**

**if (IdadeJoao >= 18 && acompanhado == true)**

**{**

**Console.WriteLine("Pode entrar.")**

**Console.WriteLine(mensagemAdicional);**

**}**

**else**

**{**

**Console.WriteLine("Não pode entrar");**

**}**

**Console.ReadLine();**

**}**

**//...**

**Dentro do else guardamos a mensagem João não está acompanhado. Atribuímos valores tanto para o caso em que ele estiver acompanhado ou não, depois, vamos executar. Primeiramente, teremos apenas a mensagem Não pode entrar. No caso, ele não pode entrar apesar de ter 18 anos, porque o valor de acompanhado é false - e nós usamos o operado lógico &&. Vamos alterar o valor de acompanhado para true para então executarmos o bloco do segundo if, no qual as duas condições são atendidas.**

**Em seguida, executaremos o bloco.**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 9 - Escopo");**

**int idadeJoao = 18;**

**bool acompanhado = true;**

**if (acompanhado == true)**

**{**

**mensagemAdicional = "João está acompanhado!";**

**}**

**else**

**{**

**mensagemAdicional = "João não está acompanhado";**

**}**

**if (IdadeJoao >= 18 && acompanhado == true)**

**{**

**Console.WriteLine("Pode entrar.")**

**Console.WriteLine(mensagemAdicional);**

**}**

**else**

**{**

**Console.WriteLine("Não pode entrar");**

**}**

**Console.ReadLine();**

**}**

**//...**

**Agora no console, teremos outra mensagem.**

**Executando projeto 9 - Escopo**

**Pode entrar.**

**João está acompanhado!**

**Queremos verificar outro caso, para isto, vamos adotar o operador ||novamente. Queremos pensar uma situação em que João pode entrar na primeira condição de ser maior ou igual a 18, mas não queremos que seja adicionada uma mensagem se ele não estiver acompanhado, por isso, removeremos o trecho referente em else.**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 9 - Escopo");**

**int idadeJoao = 18;**

**bool acompanhado = true;**

**string mensagemAdicional;**

**Console.WriteLine(mensagemAdicional);**

**if (acompanhado == true)**

**{**

**mensagemAdicional = "João está acompanhado!";**

**}**

**if (IdadeJoao >= 18 && acompanhado == true)**

**{**

**Console.WriteLine("Pode entrar.")**

**Console.WriteLine(mensagemAdicional);**

**}**

**else**

**{**

**Console.WriteLine("Não pode entrar");**

**}**

**Console.ReadLine();**

**}**

**//...**

**Mas VS já começará a sinalizar com vermelho a variável mensagemAdicional. Por que isto aconteceu? Isto aconteceu porque não podemos usá-la antes de atribuir um valor, entenderemos melhor isso, adicionaremos Console.WriteLine(mensagemAdicional). Em seguida, o editor exibirá a seguinte mensagem: Uso de variável local não atribuída "mensagemAdicional". Por enquanto, ela ainda não está recebendo nenhum valor.**

**No entanto, estamos atribuindo um valor dentro do if quando acompanhadoseja igual a true. O compilador tem a capacidade de verificar todos os blocos condicionais. Então ele verifica e testa se existe a possibilidade da mensagem, de uma variável qualquer, ser usada sem ter um valor atribuído. Nosso código permite a possibilidade de cairmos na linha Console.WriteLine(mensagemAdicional), sem atribuirmos um valor para variável - no caso em que acompanhado for igual a false.**

**Como o compilador percebeu que isto poderia acontecer e nos antecipou com a sinalização em vermelho, sendo assim, incluiremos um bloco else adicionando o caso em que João não está acompanhado.**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 9 - Escopo");**

**int idadeJoao = 18;**

**bool acompanhado = true;**

**string mensagemAdicional;**

**if (acompanhado == true)**

**{**

**mensagemAdicional = "João está acompanhado!";**

**}**

**else**

**{**

**mensagemAdicional = "João NÃO está acompanhado";**

**}**

**if (IdadeJoao >= 18 && acompanhado == true)**

**{**

**Console.WriteLine("Pode entrar.")**

**Console.WriteLine(mensagemAdicional);**

**}**

**else**

**{**

**Console.WriteLine("Não pode entrar");**

**}**

**Console.ReadLine();**

**}**

**Eliminamos a possibilidade de que chegamos a um ponto do código, sem que a variável tenha recebido um valor e o compilador ficou satisfeito. Mostramos anteriormente uma forma mais simples de atribuir e declarar uma variável simultaneamente. Removeremos a linha na qual declaramos o tipo de mensagemAdicional como string e faremos essa parte dentro dos blocos de código.**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 9 - Escopo");**

**int idadeJoao = 18;**

**bool acompanhado = true;**

**if (acompanhado == true)**

**{**

**string mensagemAdicional = "João está acompanhado!";**

**}**

**else**

**{**

**string mensagemAdicional = "João NÃO está acompanhado";**

**}**

**if (IdadeJoao >= 18 && acompanhado == true)**

**{**

**Console.WriteLine("Pode entrar.")**

**Console.WriteLine(mensagemAdicional);**

**}**

**else**

**{**

**Console.WriteLine("Não pode entrar");**

**}**

**Console.ReadLine();**

**}**

**Atribuímos um valor para a variável em ambos os casos, porém, não podemos usá-la quando esta cair dentro do if com as duas condições. Por isso, o VS vai sinalizar a variável novamente, e veremos a mensagem O nome "mensagemAdicional" não existe no contexto atual- ou seja, ela não foi declarada. Parece um retorno inesperado.**

**Acontece que as variáveis possuem um escopo onde "elas vivem" e podem ser usadas. Quando abrimos e fechamos o bloco, significa que a variável só existirá somente dentro dele. Dentro da nossa mensagemAdicional, incluiremos Console.WriteLine(mensagemAdicional). Desta vez, o compilador não vai reclamar.**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 9 - Escopo");**

**int idadeJoao = 18;**

**bool acompanhado = true;**

**if (acompanhado == true)**

**{**

**string mensagemAdicional = "João está acompanhado!";**

**Console.WriteLine(mensagemAdicional);**

**}**

**else**

**{**

**string mensagemAdicional = "João NÃO está acompanhado";**

**}**

**if (IdadeJoao >= 18 && acompanhado == true)**

**{**

**Console.WriteLine("Pode entrar.")**

**Console.WriteLine(mensagemAdicional);**

**}**

**else**

**{**

**Console.WriteLine("Não pode entrar");**

**}**

**Console.ReadLine();**

**}**

**//...**

**Mas no momento em que fecharmos o bloco e entrarmos no else, a variável não existirá mais, inclusive ela será declarada novamente com o mesmo nome. Se cairmos dentro do bloco do if, novamente, mensagemAdicional deixará de existir. Se incluirmos uma linha com Console.WriteLine() antes, a variável ainda não existirá - estando indisponível na lista do *intellisense*.**

**Então, o código funcionará mal dessa maneira, devemos criar a variável logo no início do código, para que ela exista no bloco maior. Ela será declarada na raiz do bloco e os blocos filhos, ela poderá ser utilizada também. Observe que quando tentamos declarar a variável dentro destes blocos, somos alertados pelo editor que não podemos, porque já existe mensagemAdicional nesse contexto, definida no escopo pai.**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 9 - Escopo");**

**int idadeJoao = 18;**

**bool acompanhado = true;**

**string mensagemAdicional;**

**if (acompanhado == true)**

**{**

**mensagemAdicional = "João está acompanhado!";**

**Console.WriteLine(mensagemAdicional);**

**}**

**else**

**{**

**mensagemAdicional = "João NÃO está acompanhado";**

**}**

**if (IdadeJoao >= 18 && acompanhado == true)**

**{**

**Console.WriteLine("Pode entrar.")**

**Console.WriteLine(mensagemAdicional);**

**}**

**else**

**{**

**Console.WriteLine("Não pode entrar");**

**}**

**Console.ReadLine();**

**}**

**Não podemos declarar um variável novamente com o mesmo nome. Após fazermos as alterações e entendermos qual é o comportamento de escopo de uma variável, e onde elas existem, tanto if como else possuem as chaves de abre e fecha bloco. Temos um caso no qual elas não são necessárias: quando temos uma única instrução para a execução. Por exemplo, o bloco else em que a mensagem exibida é "João NÂO está acompanhado", em uma situação assim, podemos trabalhar sem as chaves.**

**//...**

**if (acompanhado == true)**

**{**

**mensagemAdicional = "João está acompanhado!";**

**Console.WriteLine(mensagemAdicional);**

**}**

**else**

**mensagemAdicional = "João NÃO está acompanhado";**

**if (IdadeJoao >= 18 && acompanhado == true)**

**{**

**Console.WriteLine("Pode entrar.")**

**Console.WriteLine(mensagemAdicional);**

**}**

**else**

**{**

**Console.WriteLine("Não pode entrar");**

**}**

**Console.ReadLine();**

**}**

**O compilador vai enxergar o novo bloco no qual existe uma única instrução e faz parte do else. Mas no caso, voltaremos com as chaves, porque se o bloco tiver mais de uma linha, como veremos abaixo:**

**//...**

**if (acompanhado == true)**

**{**

**mensagemAdicional = "João está acompanhado!";**

**Console.WriteLine(mensagemAdicional);**

**}**

**else**

**mensagemAdicional = "João NÃO está acompanhado";**

**Console.WriteLine(mensagemAdicional);**

**if (IdadeJoao >= 18 && acompanhado == true)**

**{**

**Console.WriteLine("Pode entrar.")**

**Console.WriteLine(mensagemAdicional);**

**}**

**else**

**{**

**Console.WriteLine("Não pode entrar");**

**}**

**Console.ReadLine();**

**}**

**Se usarmos o recurso de formatação do VS, veremos que o editor vai alinhar Console.WriteLine(mensagemAdicional); no mesmo nível do else, por não considerar que a linha faz parte do bloco else.**

**Então se temos mais de uma linha em um bloco, devemos utilizar as chaves. As chaves serão opcionais apenas quando tivermos uma instrução.**

* **como usar as operações lógicas AND (&&) e OR (||);**

**Switch**

**Vimos como fazer testes com o if, mas e se precisarmos fazer vários testes? Por exemplo, temos uma variável mes e precisamos testar o seu número e imprimir o seu mês correspondente. Vamos ter que fazer 12 ifs?**

**Para esses casos, existe o comando switch, onde podemos colocar todas as opções ou rumos que o nosso programa pode tomar. Ele funciona da seguinte maneira:**

**switch (variavelASerTestada) {**

**case opção1:**

**comando(s) caso a opção 1 tenha sido escolhida**

**break;**

**case opção2:**

**comando(s) caso a opção 2 tenha sido escolhida**

**break;**

**case opção3:**

**comando(s) caso a opção 3 tenha sido escolhida**

**break;**

**default:**

**comando(s) caso nenhuma das opções anteriores tenha sido escolhida**

**}**

**O código que será executado, que no nosso caso será a impressão do nome do mês, será o código em que a condição for verdadeira:**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**class Programa**

**{**

**static void Main(String[] args)**

**{**

**int mes = 10;**

**switch (mes)**

**{**

**case 1:**

**Console.WriteLine("O mês é Janeiro");**

**break;**

**case 2:**

**Console.WriteLine("O mês é Fevereiro");**

**break;**

**case 3:**

**Console.WriteLine("O mês é Março");**

**break;**

**case 4:**

**Console.WriteLine("O mês é Abril");**

**break;**

**case 5:**

**Console.WriteLine("O mês é Maio");**

**break;**

**case 6:**

**Console.WriteLine("O mês é Junho");**

**break;**

**case 7:**

**Console.WriteLine("O mês é Julho");**

**break;**

**case 8:**

**Console.WriteLine("O mês é Agosto");**

**break;**

**case 9:**

**Console.WriteLine("O mês é Setembro");**

**break;**

**case 10:**

**Console.WriteLine("O mês é Outubro");**

**break;**

**case 11:**

**Console.WriteLine("O mês é Novembro");**

**break;**

**case 12:**

**Console.WriteLine("O mês é Dezembro");**

**break;**

**default:**

**Console.WriteLine("Mês inválido");**

**break;**

**}**

**Console.ReadLine();**

**}**

**}**

**O break irá interromper a execução do caso que o contém, para os outros não sejam executados. Se nenhuma condição for aceita, o código do default é que será executado. Por exemplo:**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**class Programa**

**{**

**static void Main(String[] args)**

**{**

**int mes = 13;**

**switch (mes)**

**{**

**case 1:**

**Console.WriteLine("O mês é Janeiro");**

**break;**

**case 2:**

**Console.WriteLine("O mês é Fevereiro");**

**break;**

**case 3:**

**Console.WriteLine("O mês é Março");**

**break;**

**case 4:**

**Console.WriteLine("O mês é Abril");**

**break;**

**case 5:**

**Console.WriteLine("O mês é Maio");**

**break;**

**case 6:**

**Console.WriteLine("O mês é Junho");**

**break;**

**case 7:**

**Console.WriteLine("O mês é Julho");**

**break;**

**case 8:**

**Console.WriteLine("O mês é Agosto");**

**break;**

**case 9:**

**Console.WriteLine("O mês é Setembro");**

**break;**

**case 10:**

**Console.WriteLine("O mês é Outubro");**

**break;**

**case 11:**

**Console.WriteLine("O mês é Novembro");**

**break;**

**case 12:**

**Console.WriteLine("O mês é Dezembro");**

**break;**

**default:**

**Console.WriteLine("Mês inválido");**

**break;**

**}**

**Console.ReadLine();**

**}**

**}**

**A impressão será Mês inválido. Então, o switch é uma solução para os ifsencadeados.**

**projeto while :**

**Resolveremos a questão em um novo projeto, que receberá o nome de 10-CalculaPoupanca. Observe que estamos no projeto 10, e no gerenciador de soluções, ele vai ficar logo abaixo do projeto 1. Se deixássemos desta forma, perderíamos a organização. Para preservá-la, incluiremos o P no início do nome do arquivo: P10-CalculaPoupanca.**

**Agora ele ficará listado depois do 9. O novo arquivo começará da seguinte forma:**

**using System;**

**using System.Collection.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace P10\_CalculaPoupanca**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**}**

**}**

**}**

**O próximo passo será adicionar Console.WriteLine() e Console.ReadLine().**

**namespace P10\_CalculaPoupanca**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 10 - Calcula poupança");**

**Console.ReadLine();**

**}**

**}**

**}**

**Vamos criar a variável valorInvestido igual a 1000. Apesar do valor na declaração dar a impressão de que se trata de uma variável do tipo inteiro, este não é o caso - nós teremos que nos preocupar com casas decimais. Começaremos a projetar o valor para um mês, então, incluiremos no cálculo o valor de rendimento.**

**Para chegarmos a esse resultado, somaremos o valor investido com o valor investido multiplicado pelo fator de rendimento. Imagine que o valor de rendimento da poupança é 0.36% nesta semana. Mas quando falamos de números percentuais e eles são trabalhados em expressões aritméticas, precisamos dividir por 100.**

**Por exemplo, se calcularmos 10% de R$ 100, o resultado será igual a 10. Se queremos encontrar 10% de um determinado valor, significa que multiplicaremos o valor total por 0.1. Ou seja, dividimos 10 por 100. Então, faremos o mesmo com o valor 0.36% para usá-lo em um expressão aritmética.**

**namespace P10\_CalculaPoupanca**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 10 - Calcula poupança");**

**double valorInvestido = 1000;**

**//0.36% = 0.0036**

**valorInvestido = valorInvestido + valorInvestido \* 0.0036;**

**Console.WriteLine("Após 1 mês, você terá R$" + valorInvestido);**

**Console.ReadLine();**

**}**

**}**

**}**

**O valor 0.36% será usado como 0.0036. Incluímos um comentário explicando como chegamos a esse número, além de adicionarmos Console.WriteLine()para que este seja exibido na tela.**

**Ao executarmos nosso código, teremos o seguinte retorno:**

**Executando projeto 10 - Calcula poupança**

**Após 1 mês, você terá R$1003,6**

**Nós começamos com R$ 1000, mas daqui um mês, teremos R$1003,6guardados na poupança. Observe que o número só possui uma casa decimal, isto acontece porque estamos trabalhando com a conversão mais básica com um double sendo transformado em texto (ou seja, uma *string*). Por padrão, C# não adiciona o zero nesta segunda casa decimal.**

**O próximo passo será realizar o cálculo para dois meses, basta adicionarmos uma nova linha, refazendo o cálculo de valorInvestido e incluirmos a mensagem com o valor total após dois meses.**

**namespace P10\_CalculaPoupanca**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 10 - Calcula poupança");**

**double valorInvestido = 1000;**

**//0.36% = 0.0036**

**valorInvestido = valorInvestido + valorInvestido \* 0.0036;**

**Console.WriteLine("Após 1 mês, você terá R$" + valorInvestido);**

**valorInvestido = valorInvestido + valorInvestido \* 0.0036;**

**Console.WriteLine("Após 2 meses, você terá R$" + valorInvestido);**

**Console.ReadLine();**

**}**

**}**

**}**

**No fim, concatenamos o valor investido com a string, em seguida, vamos executar a aplicação.**

**Executando projeto 10 - Calcula poupança**

**Após 1 mês, você terá R$1003,6**

**Após 2 meses, você terá R$1007,21296**

**O valor total após dois meses será de R$1007. Para chegarmos a esse valor, na segunda vez em que multiplicarmos o juro 0,36%, ou seja, 0.0036, ele será aplicado no valor 1003,6. Não podemos multiplicar o fator de rendimento por 2 ou por 3 para projetar vários meses, nós precisamos calcular mês a mês.**

**No entanto, ficou trabalhoso realizar o cálculo dos 12 meses, apesar de só estarmos repetindo o mesmo código. Enquanto não alcançarmos os 12 meses, a única alteração será no contador do mês exibido na mensagem.**

**Vamos encontrar uma forma de simplificar o nosso trabalho, explicando que enquanto o contador de mês for menor que 12, o mesmo código será executado. Para isto, escreveremos esta ideia no editor, sem nos preocuparmos com o processo de compilação.**

**namespace P10\_CalculaPoupanca**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 10 - Calcula poupança");**

**double valorInvestido = 1000;**

**(contadorMes <= 12)**

**{**

**valorInvestido = valorInvestido + valorInvestido \* 0.0036;**

**Console.WriteLine("Após 1 mês, você terá R$" + valorInvestido);**

**}**

**//0.36% = 0.0036**

**valorInvestido = valorInvestido + valorInvestido \* 0.0036;**

**Console.WriteLine("Após N mês, você terá" + valorInvestido);**

**//...**

**Aproveitamos o trecho com o qual realizávamos o cálculo, depois, veremos como é possível trocar o N da mensagem. Diremos ao C# que o código deve ser repetido enquanto atender a condição de maior ou igual a 12 meses. O termo "enquanto" existe no C#, em inglês, a palavra reservada será *while* - ela será utilizada no cálculo. Para que o código seja totalmente válido, vamos declarar a variável contadorMes, que será do tipo inteiro, porque não precisaremos trabalhar com casas decimais.**

**namespace P10\_CalculaPoupanca**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 10 - Calcula poupança");**

**double valorInvestido = 1000;**

**int contadorMes = 1;**

**while (contadorMes <= 12)**

**{**

**valorInvestido = valorInvestido + valorInvestido \* 0.0036;**

**Console.WriteLine("Após N mês, você terá" + valorInvestido);**

**}**

**Console.ReadLine();**

**}**

**}**

**}**

**Este código é válido e ficou bem semelhante ao if, considerando que temos a palavra reservada seguida da expressão booleana contadorMes <= 12. E dentro do bloco, temos o trecho que será executado.**

**Agora que adotamos a palavra while, vamos ver o que será exibido na tela. Ao clicarmos em iniciar, teremos a mensagem retornando infinitamente.**

**Após N mês, você terá201875,657000311**

**Após N mês, você terá202602,409365512**

**Após N mês, você terá203331,778039228**

**Após N mês, você terá204063,772440169**

**Após N mês, você terá204798,402020954**

**Após N mês, você terá205535,676268229**

**Se teclarmos "Enter", a linha do ReadLine() não será executada e, por isso, a aplicação não para.**

**Vamos analisar o que aconteceu. Temos contadorMes, que é igual 1 e enquanto essa condição for true, o bloco será executado. No fim, reavaliamos a expressão booleana, porém, contadores continuará sendo igual a 1, que será menor ou igual a 12.**

**Como nunca incrementamos o valor do contadorMes, nunca conseguiremos sair do while. Vamos incrementar e aumentar +1 o valor da variável contadorMes.**

**namespace P10\_CalculaPoupanca**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 10 - Calcula poupança");**

**double valorInvestido = 1000;**

**int contadorMes = 1;**

**while (contadorMes <= 12)**

**{**

**valorInvestido = valorInvestido + valorInvestido \* 0.0036;**

**Console.WriteLine("Após N mês, você terá" + valorInvestido);**

**contadorMes = contadorMes + 1;**

**}**

**Console.ReadLine();**

**}**

**}**

**}**

**Se executarmos o código, veremos que o N ainda não é sustituido, mas resolveremos isso a seguir. Mas observe a execução do cálculo 12 vezes.**

**Executando projeto 10 - Calcula poupança**

**Após N mês, você terá1003,6**

**Após N mês, você terá1007,21296**

**Após N mês, você terá1010,838926656**

**Após N mês, você terá1014,47794679196**

**Após N mês, você terá1018,13006740041**

**Após N mês, você terá1021,79533564305**

**Após N mês, você terá1025,47379885137**

**Após N mês, você terá1029,16550452723**

**Após N mês, você terá1032,87050034353**

**Após N mês, você terá1036,58883414477**

**Após N mês, você terá1040,32055394769**

**Após N mês, você terá1044,0657079419**

**Isto porque estamos somando +1 e temos a condição que em determinado ponto, quando o contador chega a 13, ele deixa de executar o bloco while e chegamos ao Console.ReadLine(). Em seguida, vamos modificar N pelo valor do contadorMes. Teremos que fazer ajustes na mensagem para fazer a concatenação com a variável.**

**namespace P10\_CalculaPoupanca**

**{**

**class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Executando projeto 10 - Calcula poupança");**

**double valorInvestido = 1000;**

**int contadorMes = 1;**

**while (contadorMes <= 12)**

**{**

**valorInvestido = valorInvestido + valorInvestido \* 0.0036;**

**Console.WriteLine("Após " + contadorMes + " meses, você terá R$" + valorInvestido);**

**contadorMes = contadorMes + 1;**

**}**

**Console.ReadLine();**

**}**

**}**

**}**

**Quando executamos o código, veremos que o valores serão listados de uma forma mais organizada.**

**Executando projeto 10 - Calcula poupança**

**Após 1 mês, você terá R$1003,6**

**Após 2 mês, você terá R$1007,21296**

**Após 3 mês, você terá R$1010,838926656**

**Após 4 mês, você terá R$1014,47794679196**

**Após 5 mês, você terá R$1018,13006740041**

**Após 6 mês, você terá R$1021,79533564305**

**Após 7 mês, você terá R$1025,47379885137**

**Após 8 mês, você terá R$1029,16550452723**

**Após 9 mês, você terá R$1032,87050034353**

**Após 10 mês, você terá R$1036,58883414477**

**Após 11 mês, você terá R$1040,32055394769**

**Após 12 mês, você terá R$1044,0657079419**

**Obtivemos o resultado esperado. Existe outra coisa que podemos modificar no código para deixá-lo mais elegante. Estamos somando +1 a variável contadorMes, mas isto nos permite escrever um código diferente. Vamos manter a linha com o código atual para entendermos que se trata de uma versão equivalente, em seguida, adicionaremos a variável seguida do +=, este tipo de junção com = serve para qualquer operador aritmético.**

**while (contadorMes <= 12)**

**{**

**valorInvestido = valorInvestido + valorInvestido \* 0.0036;**

**Console.WriteLine("Após " + contadorMes + " meses, você terá R$" + valorInvestido);**

**// contadorMes = contadorMes + 1;**

**contadorMes += 1;**

**}**

**Console.ReadLine();**

**}**

**}**

**}**

**Quando usamos +=, o compilador vai somar +1 com a variável à esquerda e guardará o novo valor em contadorMes. Se executarmos o código, teremos o mesmo comportamento do código.**

**Nós estamos somando apenas 1, mas também existe um forma mais apropriada de escrevermos isso, substituindo +1 por ++. Com isso repetimos a tarefa de incremetar +1 e novamente, salvamos na variável.**

**Continuaremos tendo o mesmo comportamento do código, porém, quando incrementamos +1 é estranho escrevermos desta forma. No entanto, ao somarmos uma quantidade maior, torna-se importante usar +=, como +2 ou +**

**Utilize um laço do tipo for para imprimir todos os múltiplos de 3, entre 1 e 100.**

**Dica: Existem duas maneiras tradicionais de resolver este problema. Uma delas consiste em fazer o for, e utilizar numero % 3 para descobrir o resto da divisão de um número por 3 (o operador % se chama de *módulo*). Se o resto for zero, ele é divisível por 3. Como em:**

**if(numero % 3 == 0)**

**{**

**// faça algo**

**}**

**Outra abordagem é fazer um laço um pouco diferente, que pula já direto pelos múltiplos de três. Existem outras abordagens, escolha a sua e faça a implementação em uma nova classe!**

**VER OPINIÃO DO INSTRUTOR**

### **Opinião do instrutor**

**A abordagem que utiliza o resto da divisão é:**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**class Programa**

**{**

**static void Main(String args[])**

**{**

**for (int i = 1; i < 100; i++ )**

**{**

**if (i % 3 == 0)**

**{**

**Console.WriteLine(i);**

**}**

**}**

**}**

**}**

**ou, entre outras tantas opções, a mais simples:**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**class Programa**

**{**

**static void Main(String args[])**

**{**

**for (int i = 3; i < 100; i += 3 )**

**{**

**Console.WriteLine(i);**

**}**

**}**

**}**

* **A sintaxe do while e for;**
* **O operador +=;**
* **O operador ++;**
* **Laços aninhados;**
* **A funcionalidade do break.**