

**Cetec**  
Capacitações

**CENTRO PAULA SOUZA**

GOVERNO DO ESTADO  
**SÃO PAULO**

GOVERNO FEDERAL  
**BRASIL**  
PAÍS RICO É PAÍS SEM POBREZA



NOVA SEDE ADMINISTRATIVA  
DO CENTRO PAULA SOUZA  
E CETEC NOVA LUZ

 **GOVERNO DE  
SÃO PAULO**

Rua Timbiras

Rua Gal. Carlos de Mesquita

# **Atualização Técnica e Pedagógica de Professores no componente de Lógica de Programação com C# (console)**

## **Semana 2**

### **Estruturas de Condição, Seleção e Repetição**

**Prof. Tiago Jesus de Souza**

## Introdução

Nesta semana iremos abordar assuntos fundamentais para o desenvolvimento lógico das atividades do curso.

São comandos/instruções chaves para que seja entendido como um programa realiza o fluxo de dados com tomadas de decisão, como faz para selecionar a opção de um menu e também como repetir instruções por inúmeras vezes até que uma determinada condição seja satisfeita.

Estrutura Condicional: **if / if...else**

Estrutura de Seleção: **switch**

Estruturas de repetição: **for / while / do...while**

## Estrutura Condicional Simples: if (*se*)

Tem a finalidade de tomar uma decisão e efetuar um desvio no processamento do programa, dependendo, é claro, da condição ser Verdadeira ou Falsa.

Sendo a condição Verdadeira, será executada a instrução que estiver escrita na linha logo após da instrução **if**.

Caso seja necessário executar mais de uma instrução para uma condição, elas deverão estar dentro de um bloco, ou seja, devem estar ente “{” e “}”.

**Sintaxe:** if (*condição*)  
          *instrução 1*

          ou

          if (*condição*)  
          {  
          *instrução 1*  
          *instrução 2*  
          }

## Exemplo 1 – Estrutura Condicional Simples

Neste exemplo o usuário deverá digitar dois números inteiros. Será calculada a adição deste números e na sequência existe uma condição, que irá testar se o resultado da soma que está armazenado na variável **X** é maior que 0 (zero).

Sendo verdadeira a condição, exibirá uma mensagem na tela. Caso contrário, simplesmente não será exibida mensagem na tela.

```
static void Main(string[] args)
{
    int A, B, X;
    Console.Write("Informe um valor para a variável A: ");
    A = int.Parse(Console.ReadLine());
    Console.Write("Informe um valor para a variável B: ");
    B = int.Parse(Console.ReadLine());

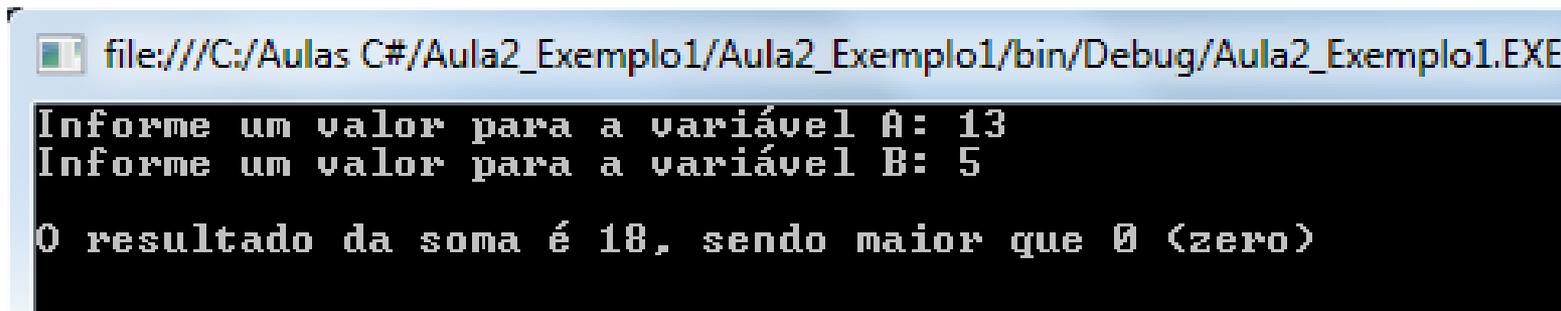
    X = A + B;

    if (X > 0)
        Console.WriteLine("\nO resultado da soma é {0}, sendo maior que 0 (zero)", X);

    Console.ReadKey();
}
```

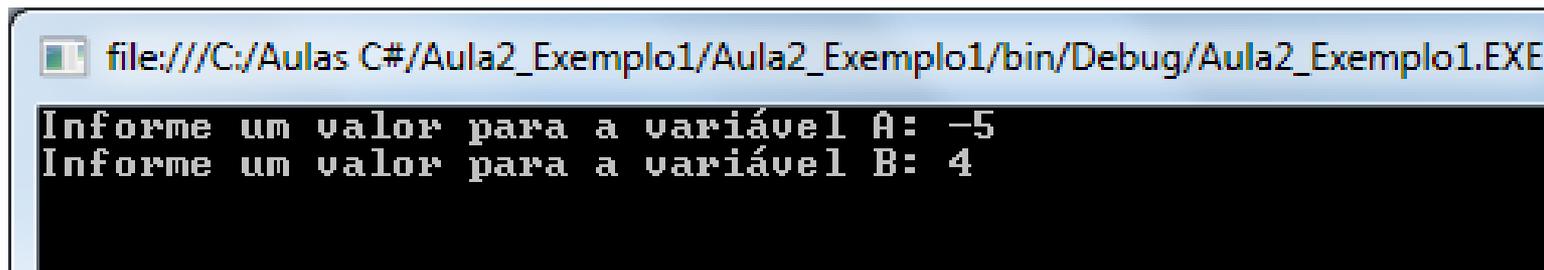
## Exemplo 1 – Estrutura Condicional Simples - Resultado

Observem na imagem a seguir, que a mensagem foi exibida na tela, pois a condição testada é verdadeira.



```
file:///C:/Aulas C#/Aula2_Exemplo1/Aula2_Exemplo1/bin/Debug/Aula2_Exemplo1.EXE
Informe um valor para a variável A: 13
Informe um valor para a variável B: 5
O resultado da soma é 18, sendo maior que 0 (zero)
```

Já no outro teste, não foi exibida nenhuma mensagem, pois a condição testada é falsa.



```
file:///C:/Aulas C#/Aula2_Exemplo1/Aula2_Exemplo1/bin/Debug/Aula2_Exemplo1.EXE
Informe um valor para a variável A: -5
Informe um valor para a variável B: 4
```

## Exemplo 2 – Estrutura Condicional Simples

Considerando o teste da condição falsa, o usuário não entenderá o que ocorreu no programa, mesmo porque, não apareceu nenhuma mensagem na tela e ele pode ficar confuso.

Para solucionar este problema, podemos incluir uma outra condição realizando o teste para saber se o resultado é menor que zero, e assim o usuário terá uma informação como resultado do programa.

```
static void Main(string[] args)
{
    int A, B, X;
    Console.Write("Informe um valor para a variável A: ");
    A = int.Parse(Console.ReadLine());
    Console.Write("Informe um valor para a variável B: ");
    B = int.Parse(Console.ReadLine());

    X = A + B;

    if (X > 0)
        Console.WriteLine("\nO resultado da soma é {0}, sendo maior que 0 (zero)", X);

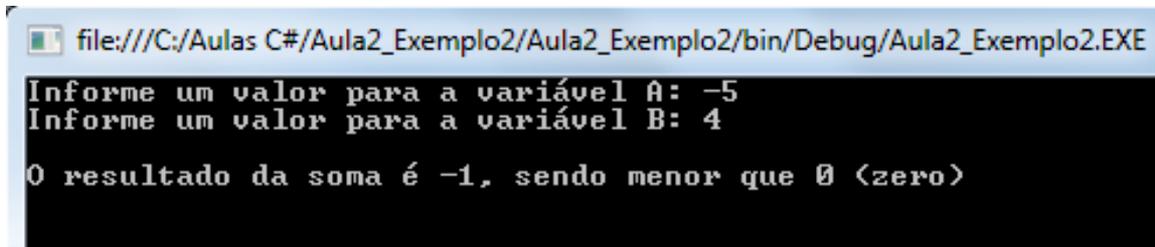
    if (X < 0)
        Console.WriteLine("\nO resultado da soma é {0}, sendo menor que 0 (zero)", X);

    Console.ReadKey();
}
```

## Exemplo 2 – Estrutura Condicional Simples - Resultado

Observem que agora existem dois testes lógicos no programa. E como o programa vai executando os comandos de cima para baixo, primeiramente realizará o teste para identificar se o resultado da soma é maior que zero. Sendo verdadeiro, executará a instrução logo após o teste.

Logo após, obrigatoriamente o programa irá continuar a execução e realizar um outro teste para verificar se o resultado da soma é menor do que zero. Sendo verdadeiro executará a instrução que está logo após este segundo teste.



```
file:///C:/Aulas C#/Aula2_Exemplo2/Aula2_Exemplo2/bin/Debug/Aula2_Exemplo2.EXE
Informe um valor para a variável A: -5
Informe um valor para a variável B: 4
O resultado da soma é -1, sendo menor que 0 (zero)
```

Mas, e se o resultado da soma for igual a zero? Não exibirá nenhuma mensagem.

Para resolver isso, basta incluir mais um teste condicional *if (X==0)* e exibir a mensagem.

## Indentação

É uma prática adotada em programação para organizar o código fonte, tornando a leitura do código muito mais fácil.

Em arquivos pequenos não se percebe tanto a necessidade, mas em arquivos mais extensos é de fundamental importância.

Para qualquer programador, deve ser um critério a ter em conta, principalmente, por aqueles que pretendam partilhar o seu código com outros. A indentação facilita também a modificação, seja para correção ou aprimoramento, do código fonte.

Existem centenas de estilos de indentação, mas, basicamente, consiste na adição de tabulações no início de cada linha na quantidade equivalente ao número de blocos em que cada linha está contida.

## Indentação - Continuação

Notem que os dois códigos apresentados são iguais, o que difere é que um está organizado (indentado) e o outro não, o que dificulta um pouco entender a hierarquia dos comandos, ou seja, um comando será realizado sempre ou só quando satisfazer uma condição?

```
static void Main(string[] args)
{
    int A, B, X;
    Console.Write("Informe um valor para a variável A: ");
    A = int.Parse(Console.ReadLine());
    Console.Write("Informe um valor para a variável B: ");
    B = int.Parse(Console.ReadLine());

    X = A + B;

    if (X > 0)
        Console.WriteLine("\nMaior que 0 (zero)", X);

    if (X < 0)
        Console.WriteLine("\nMenor que 0 (zero)", X);

    if (X == 0)
        Console.WriteLine("\nIgual a 0 (zero)", X);

    Console.ReadKey();
}
```

```
static void Main(string[] args)
{
    int A, B, X;
    Console.Write("Informe um valor para a variável A: ");
    A = int.Parse(Console.ReadLine());
    Console.Write("Informe um valor para a variável B: ");
    B = int.Parse(Console.ReadLine());

    X = A + B;

    if (X > 0)
        Console.WriteLine("\nMaior que 0 (zero)", X);

    if (X < 0)
        Console.WriteLine("\nMenor que 0 (zero)", X);

    if (X == 0)
        Console.WriteLine("\nIgual a 0 (zero)", X);

    Console.ReadKey();
}
```

## Estrutura Condicional Composta: **if...else** (*se...senão*)

Sendo a condição Verdadeira, será executada a instrução que estiver posicionada entre as instruções **if** e **else**. Sendo a condição Falsa, será executada a instrução que estiver posicionada logo após a instrução **else**.

Caso seja necessário executar mais de uma instrução, elas deverão estar dentro de um bloco, ou seja, devem estar ente “{” e “}”.

**Sintaxe:** `if (condição)`  
    *instrução 1*  
`else`  
    *instrução 2*

ou

`if (condição)`  
`{`  
    *instrução 1*  
    *instrução 2*  
`}`  
`else`  
`{`  
    *instrução 3*  
    *instrução 4*  
`}`

## Exemplo 3 - Estrutura Condicional Composta

Neste exemplo o usuário deverá digitar um número inteiro e o programa irá verificar se este número é par ou ímpar.

```
static void Main(string[] args)
{
    int N, R;
    Console.Write("Digite um número: ");
    N = int.Parse(Console.ReadLine());

    R = N % 2; //cálculo do resto da divisão de N por 2

    if (R == 0)
        Console.WriteLine("\n{0} é par", N);
    else
        Console.WriteLine("\n{0} é ímpar", N);

    Console.ReadKey();
}
```

Podemos também realizar o cálculo, na linha da instrução *if*, não precisando desta forma declarar a variável R, inclusive reduzindo o número de linhas.

```
static void Main(string[] args)
{
    int N;
    Console.Write("Digite um número: ");
    N = int.Parse(Console.ReadLine());

    if ((N % 2) == 0)
        Console.WriteLine("\n{0} é par", N);
    else
        Console.WriteLine("\n{0} é ímpar", N);

    Console.ReadKey();
}
```

## Exemplo 3 - Estrutura Condicional Composta - Resultado

Analisando o código, podemos identificar que,, somente será informado que o número é ímpar se a condição (*if*) for falsa. Com isso chegamos a conclusão que o *else* será o responsável por criar um mecanismo onde tratará as ocorrências falsas do *if*.

Exemplificando em algoritmo (português estruturado):

*se (condição for verdadeira) então*

*execute uma instrução*

*senão*

*execute outra instrução*

```
file:///C:/Aulas C#/Aula2_Exemplo3/  
Digite um número: 31  
31 é ímpar
```

```
file:///C:/Aulas C#/Aula2_Exemplo3/  
Digite um número: 56  
56 é par  
-
```

## Exemplo 4 - Estrutura Condicional Composta

Neste outro exemplo, o usuário deverá informar o nome e a idade de duas pessoas. Na sequência será exibido para o usuário o nome e a idade da pessoa mais velha. Será tratado ainda, a condição de idades iguais.

```
static void Main(string[] args)
{
    int idade1, idade2;
    string nome1, nome2;

    Console.Write("Digite nome da 1ª pessoa:");    nome1 = Console.ReadLine();
    Console.Write("Digite a idade da 1ª pessoa: "); idade1 = int.Parse(Console.ReadLine());

    Console.Write("Digite nome da 2ª pessoa: ");    nome2 = Console.ReadLine();
    Console.Write("Digite a idade da 2ª pessoa: "); idade2 = int.Parse(Console.ReadLine());

    if (idade1 > idade2)
    {
        Console.Write("\n{0} é mais velho(a) ", nome1);
        Console.WriteLine("com {0} anos de idade", idade1);
    }else
    {
        if (idade2 > idade1)
        {
            Console.Write("\n{0} é mais velho(a) ", nome2);
            Console.WriteLine("com {0} anos de idade", idade2);
        }
        else{
            Console.WriteLine("\n{0} e {1} têm {2} anos de idade.", nome1, nome2, idade1);
        }
    }

    Console.ReadKey();
}
```

## Exemplo 4 - Estrutura Condicional Composta - Resultado

Notem que realizando a indentação corretamente, fica mais fácil identificar quais instruções devem ser executadas caso as condições sejam verdadeiras ou falsas.

*Resultado da 1ª condição sendo Verdadeira.*

```
file:///C:/Aulas C#/Aula2_Exemplo4/Aula2_Exemplo4/bin/Debug/Aula2_Exemplo4.EXE
Digite nome da 1ª pessoa: Carlos
Digite a idade da 1ª pessoa: 27
Digite nome da 2ª pessoa: Juliana
Digite a idade da 2ª pessoa: 25

Carlos é mais velho(a) com 27 anos de idade
_
```

```
file:///C:/Aulas C#/Aula2_Exemplo4/Aula2_Exemplo4/bin/Debug/Aula2_Exemplo4.EXE
Digite nome da 1ª pessoa: Ricardo
Digite a idade da 1ª pessoa: 30
Digite nome da 2ª pessoa: Daniele
Digite a idade da 2ª pessoa: 32

Daniele é mais velho(a) com 32 anos de idade
_
```

*Resultado da 1ª condição sendo Falsa e a 2ª condição verdadeira.*

*Resultado da 1ª condição sendo Falsa e da 2ª condição também sendo falsa.*

```
file:///C:/Aulas C#/Aula2_Exemplo4/Aula2_Exemplo4/bin/Debug/Aula2_Exemplo4.EXE
Digite nome da 1ª pessoa: Eduardo
Digite a idade da 1ª pessoa: 17
Digite nome da 2ª pessoa: Jamile
Digite a idade da 2ª pessoa: 17

Eduardo e Jamile têm 17 anos de idade.
```

## Exemplo 4 - Estrutura Condicional Composta

Observem que se o primeiro teste condicional for verdadeiro, executará as instruções que estão dentro do bloco (delimitado pelas chaves), e o programa sequência irá para o fim do programa. Ou seja, as demais instruções não serão executadas.

Somente haverá o teste da segunda condição, se a primeira condição for falsa, executando o bloco de instruções do **else**. Na sequência, se a segunda condição for verdade executará as instruções do bloco seguinte, senão executará a instrução do bloco do próximo **else**.

Podemos chamar essa composição de teste encadeado.

```
if (idade1 > idade2)
{
    Console.WriteLine("\n{0} é mais velho(a) ", nome1);
    Console.WriteLine("com {0} anos de idade", idade1);
}else
{
    if (idade2 > idade1)
    {
        Console.WriteLine("\n{0} é mais velho(a) ", nome2);
        Console.WriteLine("com {0} anos de idade", idade2);
    }
    else{
        Console.WriteLine("\n{0} e {1} têm {2} anos de idade.", nome1, nome2, idade1);
    }
}
```

## Testes Simples x Testes Encadeados

Para uma mesma solução, porém usando lógica diferente, poderíamos utilizar três instruções *if* separadamente, como escrito a seguir:

Ambos exibem os mesmos resultados na tela, porém realizando desta forma, haverá um tempo maior de processamento, pois serão necessárias três comparações, mesmo as primeiras sendo verdadeiras.

```
if (idade1 > idade2)
{
    Console.WriteLine("\n{0} é mais velho(a) ", nome1);
    Console.WriteLine("com {0} anos de idade", idade1);
}

if (idade2 > idade1)
{
    Console.WriteLine("\n{0} é mais velho(a) ", nome2);
    Console.WriteLine("com {0} anos de idade", idade2);
}

if (idade2 == idade1)
{
    Console.WriteLine("\n{0} e {1} têm {2} anos de idade.", nome1, nome2, idade1);
}
```

Já no Exemplo 5, se a primeira condição for verdadeira, as outras não serão processadas.

## Exemplo 5 – Testes usando Operadores Lógicos

Neste exemplo, o usuário deverá digitar um valor real e o programa exibirá uma mensagem informando se este valor digitado está entre 10 e 50.

```
static void Main(string[] args)
{
    double x;

    Console.Write("Digite um valor real (ex.: 34,45): ");
    x = double.Parse(Console.ReadLine());

    if ((x > 10.00) && (x < 50.00))
        Console.WriteLine("{0} está entre 10 e 50", x);
    else
        Console.WriteLine("{0} não está entre 10 e 50", x);

    Console.ReadKey();
}
```

O operador lógico usado foi o **&&** (AND), ou seja, somente será verdadeiro se estiver entre 10 e 50.

## Exemplo 5 – Testes usando Operadores Lógicos - Resultados

Teste sendo executado com a condição sendo verdadeira.

```
file:///C:/Aulas C#/Aula2_Exemplo5/Aula2_Exemplo5/bin/Debug/Aula2_Exemplo5.EXE
Digite um valor real (ex.: 34,45): 12,5
12,5 está entre 10 e 50
-
```

Teste sendo executado com a condição sendo falsa.

```
file:///C:/Aulas C#/Aula2_Exemplo5/Aula2_Exemplo5/bin/Debug/Aula2_Exemplo5.EXE
Digite um valor real (ex.: 34,45): 5,2
5,2 não está entre 10 e 50
```

E se caso o usuário digitar um valor igual a 10 ou 50, qual será a resposta?

## Estrutura de Seleção: **switch...case** (selecione...caso)

A grande maioria dos programas, jogos e páginas web possuem um menu de opções, para deixar com que o usuário escolha o que deseja fazer, ou seja, qual das opções o mesmo deseja selecionar para executar alguma tarefa.

Com base nisso, muitos deles utilizam a estrutura de seleção **switch...case**, onde o usuário deverá informar a opção desejada e o programa executará as instruções pertencentes somente àquele item escolhido.

## Estrutura de Seleção: **switch...case** (selecione...caso)

Neste comando a execução segue os seguintes passos:

- A expressão é avaliada;
- O resultado da expressão é comparado com os valores das constantes que aparecem nos comandos **case**;
- Quando o resultado da expressão for igual a uma das constantes, a execução se inicia a partir do comando associado com esta constante. A execução continua com a execução de todos os comandos até o fim do comando **switch**, ou até que um comando **break** seja encontrado;
- Caso não ocorra nenhuma coincidência o comando **default** é executado.

Sintaxe:

```
switch (variável)
{
    case constante1:  sequencia de comandos;
                    break;

    case constante2:  sequencia de comandos;
                    break;
    .
    .
    .
    case constante_n: sequencia de comandos;
                    break;

    default:        sequencia de comandos;
                    break;
}
```

## Estrutura de Seleção – Pontos importantes

Há alguns pontos importantes que devem ser mencionados sobre o comando ***switch***:

- Notar que caso não apareça um comando de desvio (***break***) todas as instruções seguintes ao teste ***case*** que teve sucesso serão executadas, mesmo as que estejam relacionadas com outros testes ***case***;
- O comando ***switch*** só pode testar igualdade;
- Não podem aparecer duas constantes iguais em um ***case***;

## Estrutura de Seleção: Sintaxe

Neste comando a execução segue os seguintes passos:

- A expressão é avaliada;
- O resultado da expressão é comparado com os valores das constantes que aparecem nos comandos **case**;
- Quando o resultado da expressão for igual a uma das constantes, a execução se inicia a partir do comando associado com esta constante. A execução continua com a execução de todos os comandos até o fim do comando **switch**, ou até que um comando **break** seja encontrado;
- Caso não ocorra nenhuma coincidência o comando **default** é executado.

```
Sintaxe: switch ( variável_opção)  
    {  
        case valor_constante1:  
            instruções  
        break;  
  
        case valor_constante2:  
            instruções  
        break;  
  
        case valor_constanteN:  
            instruções  
        break;  
  
        default: instruções  
        break;  
    }
```

## Exemplo 6 - Estrutura de Seleção

Neste exemplo, em um primeiro momento o que aparecerá na tela para o usuário, será um menu com alguns itens. Ele deverá digitar o número correspondente a opção desejada e pressionar *<enter>*.

Na sequência será executado o *case* de acordo com a opção digitada.

Observem que neste exemplo a opção é do tipo inteiro.

```
static void Main(string[] args)
{
    int opcao;

    Console.WriteLine("1. Inclusão");
    Console.WriteLine("2. Alteração");
    Console.WriteLine("3. Exclusão");

    Console.Write("\nDigite sua opção:");
    opcao = int.Parse(Console.ReadLine());

    switch (opcao)           // início do switch
    {
        case 1: Console.WriteLine("\n Você escolheu inclusão");
                break;

        case 2: Console.WriteLine("\n Você escolheu alteração");
                break;

        case 3: Console.WriteLine("\n Você escolheu exclusão");
                break;

        default:
            Console.WriteLine("\n Opção inválida");
            break;
    }           // fim do switch
    Console.ReadKey();
}
```

## Exemplo 6 - Estrutura de Seleção - Resultado

Na primeira execução, foi digitada a opção 1 e na sequencia foi executada a instrução do *case 1*.

```
file:///C:/Aulas C#/Aula2_Exemplo6/Aula2_Exemplo6/bin/Debug/Aula2_Exemplo6.EXE
1. Inclusão
2. Alteração
3. Exclusão

Digite sua opção:1
Você escolheu inclusão
```

Em um segundo teste, foi digitada a opção 7. Como não existe *case 7*, o *default* será acionado para executar as instruções referentes à opções que não existem *cases*.

```
file:///C:/Aulas C#/Aula2_Exemplo6/Aula2_Exemplo6/bin/Debug/Aula2_Exemplo6.EXE
1. Inclusão
2. Alteração
3. Exclusão

Digite sua opção:7
Opção inválida
```

## Exemplo 7 - Estrutura de Seleção

Neste exemplo, é bem parecido com o Anterior. O que muda é a forma de tratar a opção desejada.

Ao invés de números inteiros, são usados caracteres para identificar cada item do menu.

E para isso o tipo da variável *opcao* é *char*.

E os caracteres correspondentes relacionados no *case*, devem estar entre aspas simples ' '.

```
static void Main(string[] args)
{
    char opcao;
    Console.WriteLine("I - Inclusão");
    Console.WriteLine("A - Alteração");
    Console.WriteLine("E - Exclusão");

    Console.Write("\nDigite sua opção:");
    opcao = char.Parse(Console.ReadLine());

    switch (opcao)
    {
        // início do switch
        case 'I': Console.WriteLine("\n Você escolheu inclusão");
                break;

        case 'A': Console.WriteLine("\n Você escolheu alteração");
                break;

        case 'E': Console.WriteLine("\n Você escolheu exclusão");
                break;

        default: Console.WriteLine("\n Opção inválida");
                break;
    } // fim do switch
    Console.ReadKey();
}
```

## Exemplo 7 - Estrutura de Seleção - Resultado

Na primeira execução, foi digitada a opção *I* e na sequencia foi executada a instrução do *case I*.

```
file:///C:/Aulas C#/Aula2_Exemplo7/Aula2_Exemplo7/bin/Debug/Aula2_Exemplo7.EXE
I - Inclusão
A - Alteração
E - Exclusão

Digite sua opção:I
Você escolheu inclusão
```

No teste da segunda execução foi digitada a opção *E*.

```
file:///C:/Aulas C#/Aula2_Exemplo7/Aula2_Exemplo7/bin/Debug/Aula2_Exemplo7.EXE
I - Inclusão
A - Alteração
E - Exclusão

Digite sua opção:E
Você escolheu exclusão
```

## Exemplo 8 - Estrutura de Seleção

Neste exemplo, o que muda também é a forma de tratar a opção desejada.

Ao invés de números inteiros ou simples caracteres, são usadas strings para identificar cada item do menu.

E para isso o tipo da variável *opcao* é *string*.

E as strings correspondentes relacionados no *case*, devem estar entre aspas “ ”.

```
static void Main(string[] args)
{
    string opcao;
    Console.WriteLine("Inclusão");
    Console.WriteLine("Alteração");
    Console.WriteLine("Exclusão");

    Console.Write("\nDigite a ação desejada:");
    opcao = Console.ReadLine();

    switch (opcao)
    {
        // início do switch
        case "Inclusão": Console.WriteLine("\n Você escolheu inclusão");
            break;

        case "Alteração": Console.WriteLine("\n Você escolheu alteração");
            break;

        case "Exclusão": Console.WriteLine("\n Você escolheu exclusão");
            break;

        default: Console.WriteLine("\n Opção inválida");
            break;
    } // fim do switch
    Console.ReadKey();
}
```

## Exemplo 8 - Estrutura de Seleção - Resultado

A opção desejada deve ser digitada exatamente igual a string que consta no case. Caso contrário, serão executadas as instruções do *default*.

```
file:///C:/Aulas C#/Aula2_Exemplo8/Aula2_Exemplo8/bin/Debug/Aula2_Exemplo8.EXE
Inclusão
Alteração
Exclusão

Digite a ação desejada:Alteração

Você escolheu alteração
```

```
file:///C:/Aulas C#/Aula2_Exemplo8/Aula2_Exemplo8/bin/Debug/Aula2_Exemplo8.EXE
Inclusão
Alteração
Exclusão

Digite a ação desejada:alteracao

Opção inválida
_
```

## Estruturas de Repetição

Em muitas situações durante o desenvolvimento de programas, existe a necessidade de repetir por inúmeras vezes um determinado trecho de código, ou seja, repetir por várias vezes algumas instruções.

Anda em outras palavras, os comandos de repetição permitem que um conjunto de instruções seja executado até que satisfaça uma determinada condição.

Sendo assim, devemos utilizar estruturas de repetição para que isso seja possível.

As estruturas de repetição são:

- **for** (para)
- **while** (Enquanto)
- **do...while** (faça...enquanto)

## Estrutura de Repetição: Comando FOR

É uma estrutura muito utilizada principalmente quando se sabe quantas vezes deve ser executado um conjunto de instruções.

A forma geral do comando *for* é:

```
for (inicialização ; condição ; incremento)  
{  
    Intruções;  
}
```

Normalmente, a **inicialização** é realizada através de um comando de atribuição, que é usado para colocar um valor na variável de controle do laço. A **condição** determina quando a repetição acaba. O **incremento ou decremento** define como a variável de controle do laço varia cada vez que o laço é repetido.

## Exemplo 9 - Estrutura de Repetição: Comando FOR (incremento)

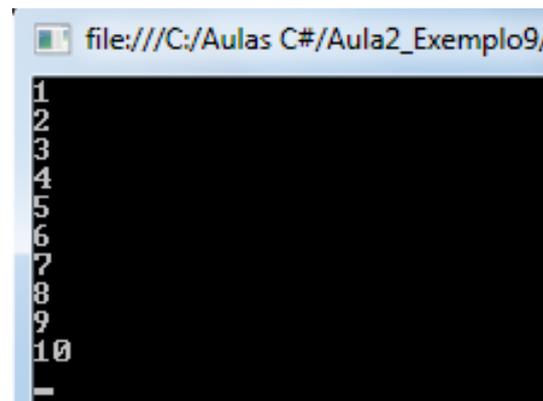
Neste exemplo serão exibidos na tela, todos os números de 1 a 10 em ordem crescente.

```
static void Main(string[] args)
{
    int x;

    for (x = 1; x <= 10; x++)
    {
        Console.WriteLine("{0}", x);
    }

    Console.ReadKey();
}
```

### Resultado



```
file:///C:/Aulas C#/Aula2_Exemplo9/
1
2
3
4
5
6
7
8
9
10
```

Onde:

- **x = 1** é a inicialização da variável, que começa o laço de repetição valendo **1**.
- **x <= 10** é a condição de parada do laço de repetição, onde enquanto **x** for menor ou igual a **10**, o bloco de instruções será executado.
- **x++** é o incremento, onde a cada iteração a variável **x** passa a ser incrementada

## Exemplo 10 - Estrutura de Repetição: Comando FOR (decremento)

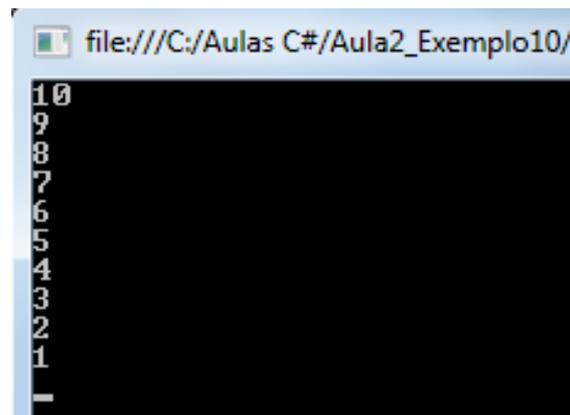
Neste exemplo serão exibidos na tela, todos os números de 1 a 10 em ordem decrescente.

```
static void Main(string[] args)
{
    int x;

    for (x = 10; x >= 1; x--)
    {
        Console.WriteLine("{0}", x);
    }

    Console.ReadKey();
}
```

### Resultado



```
file:///C:/Aulas C#/Aula2_Exemplo10/
10
9
8
7
6
5
4
3
2
1
```

Onde:

- **x = 10** é a inicialização da variável, que começa o laço de repetição valendo **10**.
- **x >= 1** é a condição de parada do laço de repetição, onde enquanto **x** for maior ou igual a **1**, o bloco de instruções será executado.
- **x--** é o decremento, onde a cada iteração a variável **x** passa a ser decrementada

## Estrutura de Repetição: Comando WHILE

Esta estrutura de laço de repetição caracteriza-se por efetuar um teste lógico no início do laço de repetição, verificando se é permitido executar o trecho de instruções subordinado a ele.

Sintaxe:

```
while (<condição>)  
{  
    Instruções para condição verdadeira;  
}
```

## Estrutura de Repetição: Comando WHILE

A estrutura *while* tem o seu funcionamento controlado por condição. Desta forma, pode executar um determinado conjunto de instruções enquanto a condição verificada permanecer *Verdadeira*. No momento em que esta condição se torna *Falsa*, o processamento da rotina é desviado para fora do laço de repetição, ou seja, a execução do bloco de instruções é encerrada.

Caso seja a condição *Falsa* logo no início do laço de repetição, as instruções contidas nele são ignoradas.

Caso seja necessário executar mais de uma instrução para uma condição verdadeira dentro de um laço, elas devem estar definidas dentro de um bloco por meio dos símbolos de chaves.

## Exemplo 11 - Estrutura de Repetição: Comando WHILE

Neste exemplo serão exibidos na tela, todos os números de 1 a 20.

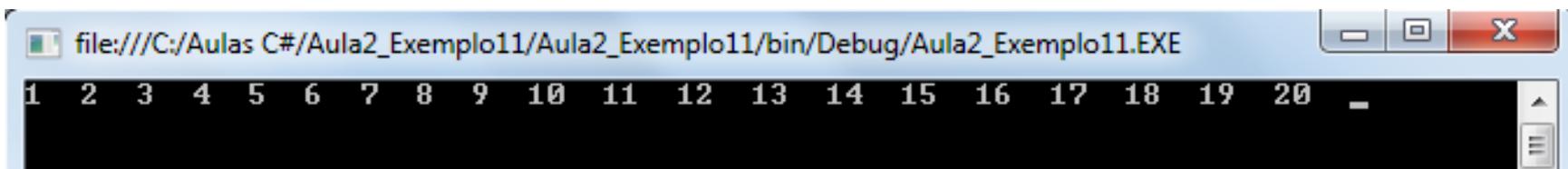
O valor de `num` inicial é **1**. Na sequência será realizado o teste condicional do comando **while**, que será verdadeiro, pois o valor de **num** é menor ou igual a **20**. Sendo assim, será exibido o valor de **num** na tela e em seguida haverá um incremento de **num** que passa a valer **2**, e retorna para o teste condicional, que sendo verdadeiro, continuará a execução do bloco de instruções. Caso contrário será encerrada a execução do bloco de instruções.

```
static void Main(string[] args)
{
    int num = 1;

    while (num <= 20)
    {
        Console.WriteLine("{0} ", num);
        num = num + 1;    // ou num++
    }

    Console.ReadKey();
}
```

## Resultado



```
file:///C:/Aulas C#/Aula2_Exemplo11/Aula2_Exemplo11/bin/Debug/Aula2_Exemplo11.EXE
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 _
```

## Exemplo 12 - Estrutura de Repetição: Comando WHILE

Neste exemplo o usuário deverá digitar um número inteiro e exibir todos os números pares em ordem crescente menores que 20, a partir no número digitado pelo usuário.

```
static void Main(string[] args)
{
    int n;

    Console.Write("Digite um número: ");
    n = int.Parse(Console.ReadLine());

    while (n < 20)
    {
        if (n % 2 == 0)
        {
            Console.WriteLine("{0}", n);
        }
        n++;
    }
    Console.ReadKey();
}
```

## Exemplo 12 - Estrutura de Repetição: Comando WHILE - Resultado

Observem que quando usuário digita um número menor que 20, a condição testada é verdadeira e assim é executado o bloco de instruções até que a condição seja falsa.

```
file:///C:/Aulas C#/Aula2_Exemplo12/A
Digite um número: 3
4
6
8
10
12
14
16
18
-
```

```
file:///C:/Aulas C#/Aula2_Exemplo12/A
Digite um número: -2
-2
0
2
4
6
8
10
12
14
16
18
```

```
file:///C:/Aulas C#/Aula2_Exemplo12/A
Digite um número: 11
12
14
16
18
-
```

Já se o número digitado não for menor do que 20 nenhum resultado será exibido na tela, pois a condição inicial não foi verdadeira.

```
file:///C:/Aulas C#/Aula2_Exemplo12/A
Digite um número: 51
-
```

## Estrutura de Repetição: Comando DO...WHILE

Esta estrutura faz um teste lógico no final de um laço de repetição. Ela é parecida com a estrutura **while**. Seu funcionamento é controlado também por condição. Esse tipo de laço executa um conjunto de instruções pelo menos uma vez antes de verificar a validade da condição estabelecida.

Desta forma **do...while** irá processar um conjunto de instruções, no mínimo uma vez, até enquanto a condição for *verdadeira*.

Sintaxe:

```
do{  
    instruções enquanto verdadeiras;  
}  
while (condição);
```

## Exemplo 13 - Estrutura de Repetição: Comando DO...WHILE

Neste exemplo, o bloco de repetição será executado pelo menos uma vez. Sendo iniciado o usuário digitará um número inteiro. Se este número for diferente de zero, exibirá o resultado do dobro deste número digitado. Senão informará o fim do programa. Ou seja, será executado o bloco de instruções enquanto a condição do **while** for verdadeira.

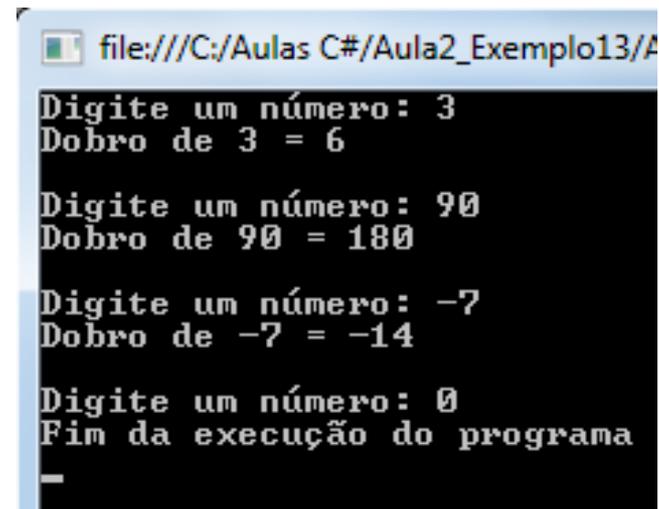
```
static void Main(string[] args)
{
    int A;

    do
    {
        Console.Write("Digite um número: ");
        A = int.Parse(Console.ReadLine());

        if (A != 0)
            Console.WriteLine("Dobro de {0} = {1} \n", A, (A * 2));
        else
            Console.WriteLine("Fim da execução do programa");
    } while (A != 0);

    Console.ReadKey();
}
```

### Resultado:



```
file:///C:/Aulas C#/Aula2_Exemplo13/A
Digite um número: 3
Dobro de 3 = 6

Digite um número: 90
Dobro de 90 = 180

Digite um número: -7
Dobro de -7 = -14

Digite um número: 0
Fim da execução do programa
_
```

## Estruturas de Repetição, qual usar?

Conforme conseguimos observar nos exemplos, ambas as estruturas possuem a mesma finalidade, que é executar um bloco de instruções várias vezes. Mas como saber qual utilizar? É simples, não existe uma regra que diz qual é a melhor forma. O que pode existir é um estudo sobre o desempenho de cada estrutura para cada problema a ser resolvido.

O que podemos concluir em linhas gerais para auxiliar na tomada de decisão em qual estrutura utilizar, é analisar o problema e usar:

- **for:** quando se sabe quantas vezes um bloco de instruções deverá ser executado.
- **while:** quando desejar que um bloco de instruções seja executado por várias vezes somente se a condição inicial for verdadeira (teste no início).
- **do...while:** quando desejar que um bloco de instruções seja executado pelo menos uma vez (teste no final).

## Exemplo 14 – Utilizando Diversas Estruturas

Escreva um programa que exiba o seguinte menu na tela:

- 1 – Par/Ímpar
- 2 – Decrescente até 0 (zero)
- 3 - Sair

Para cada item deverá ser implementado o que se pede, sendo que se o usuário escolher a opção:

- 1, deverá ser digitado um número e o programa informará se este número é par ou ímpar.
- 2, deverá ser digitado um número e o programa exibirá todos os números em ordem decrescente até 0 (zero)
- 3, deverá encerrar o programa.

## Exemplo 14 – Utilizando Diversas Estruturas - Resolução

```
static void Main(string[] args)
{
    int op, n;
    do
    {
        Console.Clear();
        Console.WriteLine("1 - Par/ímpar");
        Console.WriteLine("2 - Decrescente até 0 (zero)");
        Console.WriteLine("3 - Sair");
        Console.Write("\nInforme a opção desejada: "); op = int.Parse(Console.ReadLine());

        switch (op)
        {
            case 1: Console.Write("Digite um número: "); n = int.Parse(Console.ReadLine());
                Console.WriteLine(n + " é " + (n % 2 == 0 ? "par" : "ímpar"));
                break;

            case 2: Console.Write("Digite um número: "); n = int.Parse(Console.ReadLine());
                while (n >= 0)
                    Console.Write("{0} ", n--);
                break;

            case 3: Console.WriteLine("Fim da execução do programa");
                break;

            default: Console.WriteLine("Opção Inválida - Tente novamente");
                break;
        }
        Console.WriteLine("\nTecla algo para continuar");
        Console.ReadKey();
    } while (op != 3);
}
```

## Exemplo 14 – Utilizando Diversas Estruturas – Comentários

Neste exemplo foram utilizadas várias estruturas e instruções já vistas no curso até o momento:

switch...case, while, do...while, operadores ternários e decremento

Este foi apenas um exemplo de resolução deste exercício, pois poderia ainda ter usado **if...else**, em vez de usar operadores ternários. Poderia utilizar **for**, em vez de **while**.

Tente fazer o mesmo programa de outras formas, porém devendo chegar ao mesmo resultado.

**Console.Clear();** instrução para limpar a tela.

*Obs.: Nas linhas de instrução para o usuário digitar valores, foi colocado o comando de entrada de dados na mesma linha, apenas para colocar em um só slide o código fonte.*

## Exemplo 14 – Utilizando Diversas Estruturas - Resultado

Observem que ao iniciar o programa já é exibido um menu.

Escolhendo a opção 1, será solicitado ao usuário que digite um número e o resultado será par ou ímpar.

Na sequência irá retornar ao menu, e o usuário deverá escolher outra opção, ou seja, somente será encerrado o programa, quando o usuário escolher a opção 3.

Realize todos os testes.

```
file:///C:/Aulas C#/Aula2_Exemplo14/Aula2_Exemplo14/bin/Debug/Aula2_Exemplo14.EXE
1 - Par/ímpar
2 - Decrescente até 0 (zero)
3 - Sair

Informe a opção desejada: 1
Digite um número: 55
55 é ímpar

Tecle algo para continuar
_
```

```
file:///C:/Aulas C#/Aula2_Exemplo14/Aula2_Exemplo14/bin/Debug/Aula2_Exemplo14.EXE
1 - Par/ímpar
2 - Decrescente até 0 (zero)
3 - Sair

Informe a opção desejada: 2
Digite um número: 9
9 8 7 6 5 4 3 2 1 0

Tecle algo para continuar
_
```

```
file:///C:/Aulas C#/Aula2_Exemplo14/Aula2_Exemplo14/bin/Debug/Aula2_Exemplo14.EXE
1 - Par/ímpar
2 - Decrescente até 0 (zero)
3 - Sair

Informe a opção desejada: 3
Fim da execução do programa

Tecle algo para continuar
_
```

## Exemplo 15 – Contagem do número de elementos de um conjunto

Neste exemplo, o usuário terá que digitar vários números inteiros e ao final o programa irá informar a quantidade de números que foram digitados. A condição de parada deste laço de repetição, será quando o usuário digitar o valor 0 (zero), que não deverá entrar para a contagem dos números válidos digitados.

```
static void Main(string[] args)
{
    int q = 0, n;
    do
    {
        Console.WriteLine("\n\nPara sair digite 0 (zero) ");
        Console.Write("\nDigite um número: ");
        n = int.Parse(Console.ReadLine());
        if (n != 0)
            q++;

    } while (n != 0);

    Console.WriteLine("\nForam digitados {0} números inteiros", q);
    Console.WriteLine("\n\nTecla algo para continuar");
    Console.ReadKey();
}
```

## Exemplo 15 – Contagem do número de elementos de um conjunto

A variável  $n$  serviu para armazenar o número digitado pelo usuário.

A variável  $q$  serviu para a contagem dos números válidos digitados, que no caso ela inicia com valor 0 (zero) e na sequência entra no laço de repetição.

Ao passo que o usuário digita um número, é realizada uma verificação para saber se o número digitado é diferente de 0 (zero).

Sendo verdadeira a condição será realizado um incremento na variável  $q$ .

O programa se encerrado somente quando o usuário digitar o valor 0 (zero).

```
file:///C:/Aulas C#/Aula2_Exemplo15/Aula2_Exemplo15,
Para sair digite 0 <zero>
Digite um número: 23
Para sair digite 0 <zero>
Digite um número: 98
Para sair digite 0 <zero>
Digite um número: 12
Para sair digite 0 <zero>
Digite um número: 0
Foram digitados 3 números inteiros
Tecla algo para continuar
```

## Exemplo 15 – Acúmulo de resultados parciais

Neste exemplo, o usuário terá que digitar vários números reais e ao final o programa irá informar a soma dos valores que foram digitados. A condição de parada deste laço de repetição, será quando o usuário digitar o valor 0 (zero).

```
static void Main(string[] args)
{
    double soma = 0, n;
    do
    {
        Console.WriteLine("Para sair digite 0 (zero) ");
        Console.Write("\nDigite um número: ");
        n = double.Parse(Console.ReadLine());
        soma += n;           // ou soma = soma + n
        Console.WriteLine("\nAcumulo da Soma = {0}", soma);
        Console.WriteLine("-----");
    } while (n != 0);

    Console.WriteLine("\n\nTecle algo para continuar");
    Console.ReadKey();
}
```

## Exemplo 15 – Acúmulo de resultados parciais - Resultado

A variável  $n$  serviu para armazenar o número digitado pelo usuário.

A variável soma serve para armazenar o acúmulo do resultado da soma de todos os valores digitados pelo usuário. Esta variável inicia valendo 0 (zero).

O usuário digitará um valor e na sequência a variável soma receberá o resultado da adição da variável soma com a variável  $n$ .

Ou seja, enquanto o usuário não digitar 0 (zero), o usuário irá digitar um valor e será acumulada na variável soma o acúmulo da soma dos números digitados pelo usuário.

```
file:///C:/Aulas C#/Aula2_Exemplo16/Aula2_Exemplo16
Para sair digite 0 <zero>
Digite um número: 3
Acumulo da Soma = 3
-----
Para sair digite 0 <zero>
Digite um número: 5
Acumulo da Soma = 8
-----
Para sair digite 0 <zero>
Digite um número: 75
Acumulo da Soma = 83
-----
Para sair digite 0 <zero>
Digite um número: 0
Acumulo da Soma = 83
-----
Tecle algo para continuar
_
```

## Bibliografia

- Manzano, José Augusto N. G., **Estudo Dirigido de Microsoft Visual C# 2010 Express.** São Paulo, SP, Editora Érica, 2010.
- MSDN, Microsoft. **Guia de Programação C#.** Disponível: [http://msdn.microsoft.com/pt-br/library/67ef8sbd\(v=vs.80\).aspx](http://msdn.microsoft.com/pt-br/library/67ef8sbd(v=vs.80).aspx). Acesso em 22 jul 2013
- Wikipedia. **Indentação.** Disponível: <http://pt.wikipedia.org/wiki/Indenta%C3%A7%C3%A3o> >;. Acessado em 22 jul 2013