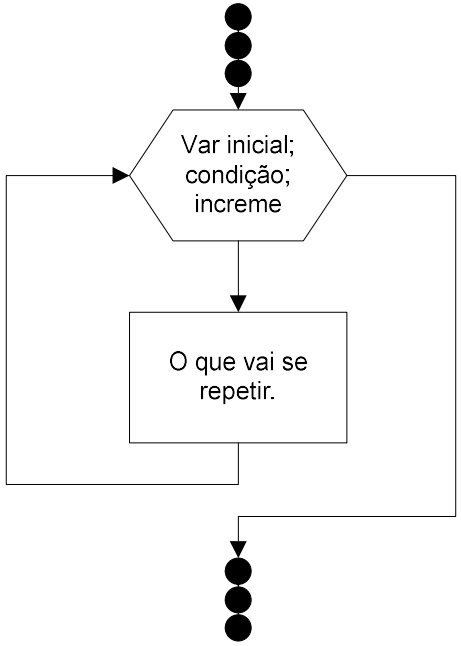
Estruturas de Repetição

- Estruturas de Repetição: são blocos de instruções que alteram o fluxo de execução do código de um programa. Com elas é possível repetir uma série de comandos várias vezes.
- Tipos:
  - While
  - For
  - Do.. While
  - Foreach
- Estrutura de decisão:
  - If – Else
  - Case

Estrutura FOR

A Estrutura For é um laço de repetição / loop, executa repetição de um conjunto de instruções enquanto uma determinada condição é verdadeira.

```
For(inicial; condição; incremento)
{
    //o que vai se repetir
}
```



Estrutura for

```
int c;
Console.WriteLine("A Sequencia de zero a mil:");
for (c = 0; c <= 1000; c++)
{
   Console.WriteLine(c);
}
Console.ReadKey();
```

Primeiro parâmetro: Neste parâmetro dizemos qual vai ser a variável que vai contar o numero de repetições que será feita por esta estrutura. Também definimos aqui o momento inicial da contagem. No nosso exemplo temos a variável ‘c’ igualada a zero, iniciando nossa contagem em zero.

Segundo parâmetro: Definimos a condição de saída ou seja, a condição que, enquanto respeitada, faz com que o laço continue “amarrado”. Neste exemplo vemos que enquanto o valor da variável ‘c’ for menor ou igual a mil, o laço permanece.

Terceiro parâmetro: No terceiro parâmetro temos o incremento, ou seja, o quanto é acrescentando a nossa variável a cada ciclo. Em nosso exemplo temos “c++” que indica o incremento de uma unidade (1) em cada volta.

Estrutura For (Exemplos)

Criar um programa que permita a soma de 10 números digitados pelo usuário, apresentar na tela media destes números.

```
{//INICIO
    int c;
    double num, soma, med;
    Console.WriteLine("digite dez numeros:");
    soma = 0;
    for (c = 1; c <= 10; c++)
    {
        num = Convert.ToDouble(Console.ReadLine());
        soma = soma + num;
    }
    med = soma / 10;
    Console.WriteLine("A media:" + med);
    Console.ReadKey();
}//FIM
```

Estrutura For (Exemplos)

Parecido com o programa apresentado no exemplo anterior, neste exemplo iremos verificar se o numero digitado pelo usuário é par e somente se for par iremos acrescentar ao somador.

```
{//INICIO
    int c;
    double num, soma, med, rest;
    Console.WriteLine("digite dez numeros:");
    soma = 0;
    for (c = 1; c <= 10; c++)
    {
        num = Convert.ToDouble(Console.ReadLine());
        rest = num % 2;
        if (rest == 0)
        {
            soma = soma + num;
        }
    }
    med = soma / 10;
    Console.WriteLine("A media:" + med);
    Console.ReadKey();
}//FIM
```

Exercícios (For)

1 – Crie um programa que permita o usuário digitar um numero qualquer. Imprimir na tela a tabuada deste numero.

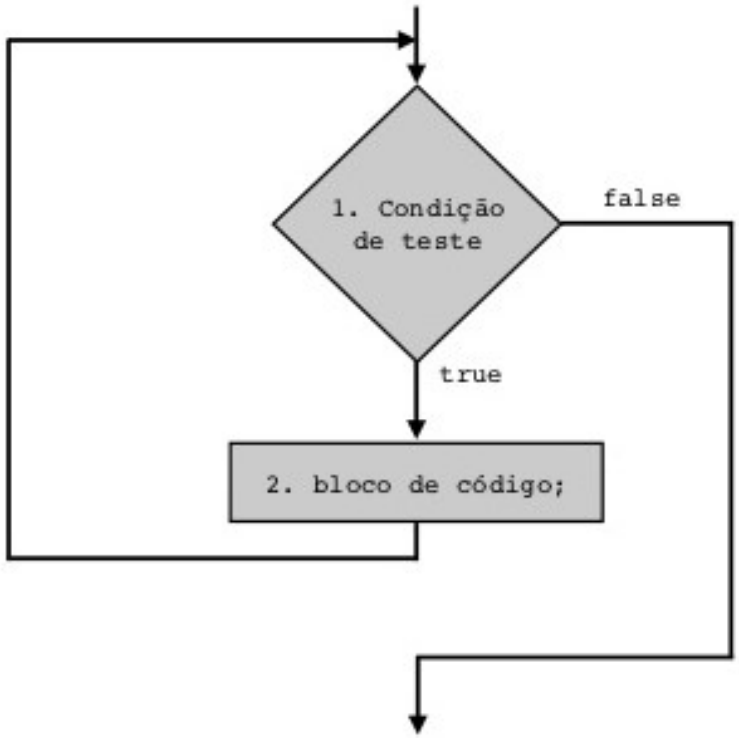
2 - Ler um valor N e imprimir todos os valores inteiros entre 1 (inclusive) e N (inclusive). Considere que o N será sempre maior que ZERO.

3 – Crie um programa que permita que o usuário digite 10 números, some apenas os números digitados maiores do que 10.

4 –Criar um programa que permita o usuário entrar com vinte números, após o termino da digitação, mostrar na tela qual o maior e menor numero que o usuário digitou.

WHILE

O while trata-se da estrutura de repetição mais utilizada quando programamos com C#. Com ela, enquanto a condição for verdadeira o bloco de código será executado. Primeiramente o sistema testa essa condição. Caso verdadeira, executa as linhas declaradas dentro do while; do contrário, sai do loop.



WHILE

```
1 while (condição)
2 {
3         //bloco de código
4 }
```

Essa estrutura permite avaliar a condição para iniciar sua execução. Isso para o caso da condição for verdadeira.

Exemplo WHILE

```
{//INICIO
    int num, c, resp;
    Console.WriteLine("Digite o numero da tabuada:");
    num = Convert.ToInt32(Console.ReadLine());
    c = 0;
    while (c <= 10)
    {
        resp = c * num;
        Console.WriteLine(resp);
        c = c + 1;
    }
    Console.ReadKey();

}//FIM
```

Tratamento de Erro

O erro no programa pode acontecer por uma entrada indevida, alguns erros , a execução do programa simplesmente continua, em outras ela pode ser interrompida abruptamente.

Por estes motivos, devemos dar atenção muito especial à detecção e manipulação de erros em nossas aplicações. No C#, contamos com um mecanismo sofisticado que nos auxilia a produzir códigos de manipulação organizados e muito eficientes, que é a manipulação de exceções.

Tratamento de Erro

O manipulador de exceções é um código que captura a exceção lançada e a manipula. Sua principal vantagem é que precisamos escrever apenas uma vez o código de manipulação de uma exceção que pode ocorrer em uma região controlada.

O bloco de código onde as exceções podem ocorrer é chamado de região protegida. Ele é indicado pelo comando try. Para associarmos uma determinada exceção a um bloco de código que a manipulará, usamos uma ou mais cláusulas catch. Veja um exemplo:

Tratamento de Erro

```
try{
catch{
finally{
```

```
try{
  //Comando de execução normal
}
catch{
  //manipulação do erro
}
finally{
  //execução de finalização
}
```

Tratamento de Erro

```
{//INICIO
    Console.WriteLine("Entrar com um numero: ");
    try
    {
        int numero = Convert.ToInt32(Console.ReadLine());
    }
    catch
    {
        Console.WriteLine("Digite um valor válido");
    }
    Console.ReadKey();
}//FIM
```

Tratamento de Erro

Podemos também capturar o erro e uma instância da classe Exception (o que é geralmente usado nas aplicações em geral):

```
{//INICIO

    Console.WriteLine("Entrar com um numero: ");
    try
    {

        int numero = Convert.ToInt32(Console.ReadLine());
    }
    catch (Exception ex)
    {
        Console.WriteLine("Digite um valor válido");
        Console.WriteLine(ex);

    }
    Console.ReadKey();
}//FIM
```

Tratamento de Erro

No C#, contamos com um recurso muito útil, que é um bloco que nos permite liberar recursos que foram alocados na execução do programa e que, após a execução do try, não precisam mais ser utilizados. Trata-se do bloco finally: quando usado em conjunto com o bloco try, os comandos que estiverem entre o bloco definido por finally{} sempre serão executados, independentemente se foram geradas exceções ou não no bloco try.

Tratamento de Erro

```
{//INICIO
    Console.WriteLine("Entrar com um numero: ");
    try
    {
        int numero = Convert.ToInt32(Console.ReadLine());
    }
    catch
    {
        Console.WriteLine("Digite um valor válido");
    }
    finally{
        Console.WriteLine("Obrigado!");
    }
    Console.ReadKey();
}//FIM
```

Exercícios(While)

1 - Elabore um programa que exiba na tela uma sequencia de números naturais começando em I, terminando em F.

2 - Faça um algoritmo que exiba quantas pessoas possuem mais de 18 anos. O algoritmo deverá ler a idade de 10 pessoas.

3 - Ler 10 valores e escrever quantos desses valores lidos são NEGATIVOS.

4 - Escreva um programa que imprima as seguintes sequencias de números: (1, 1 2 3 4 5 6 7 8 9 10) (2, 1 2 3 4 5 6 7 8 9 10) (3, 1 2 3 4 5 6 7 8 9 10) (4, 1 2 3 4 5 6 7 8 9 10) e assim sucessivamente, até que o primeiro número (antes da vírgula), também chegue a 10.