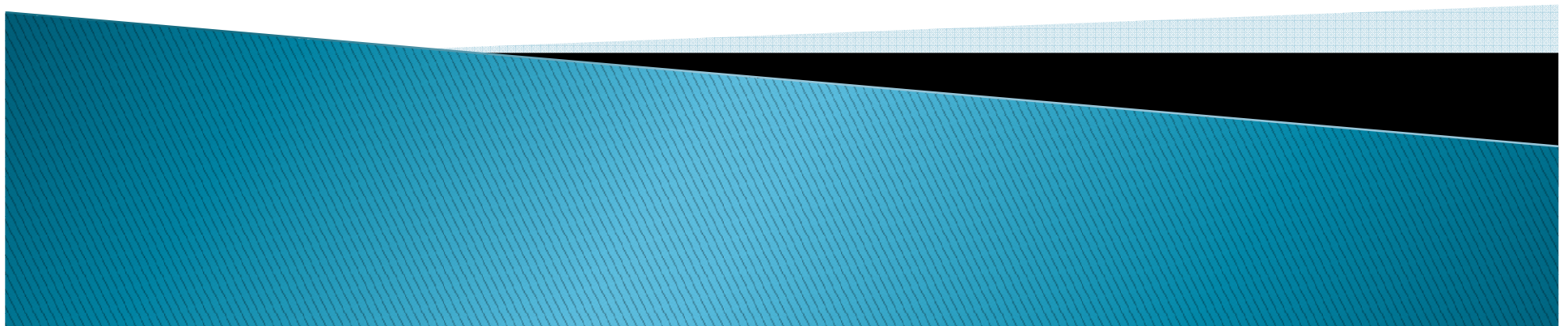


Instituto Federal da Bahia-IFBa
Curso: Engenharia Elétrica
Curso da Linguagem C

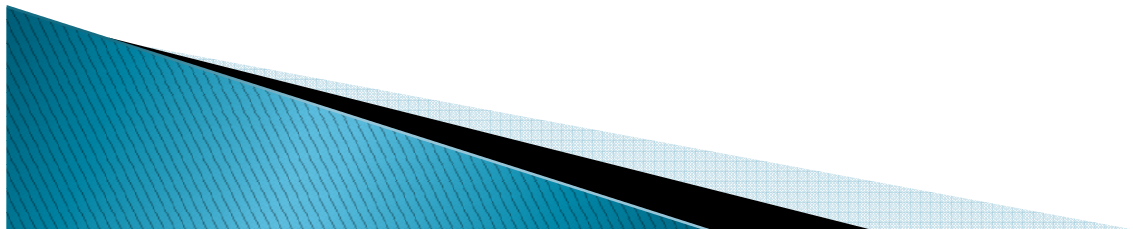
Estruturas Iteração

Prof. Luiz Cláudio Machado



Estruturas Iteração

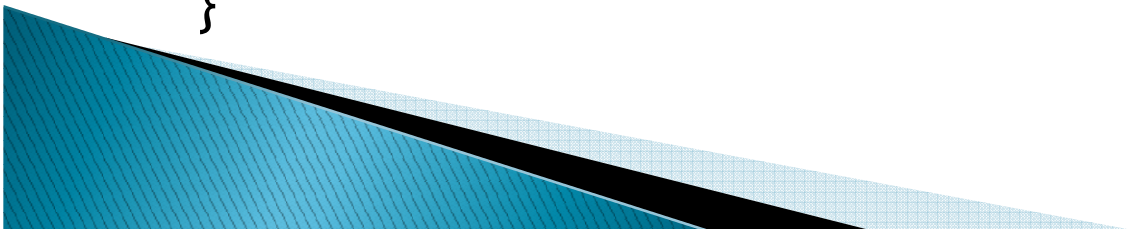
- ▶ Conhecidas também como comando de repetição e/ou laços, loops.
- ▶ Basicamente existem três tipos de estruturas de iteração na linguagem C:
 - ▶ FOR (para/variando),
 - ▶ WHILE (enquanto/faça)
 - ▶ DO ... WHILE (repita/até).



Estruturas Iteração– FOR

- ▶ A ideia básica do comando for é que você execute um conjunto de comandos, um número fixo de vezes, enquanto uma variável de controle é incrementada ou decrementada a cada passagem pelo laço.

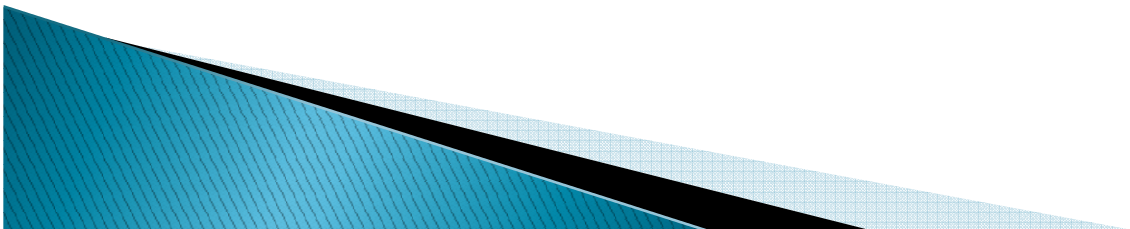
```
#include <stdio.h>
#include <conio.h>
main ( )
{
    int i;
    clrscr( );
    for ( i = 1; i <= 10; i ++ )
        printf ("%d\n", i);
}
```



Estruturas Iteração– FOR

- ▶ FOR (inicialização da variável de controle; condição; o incremento ou decremento da variável)

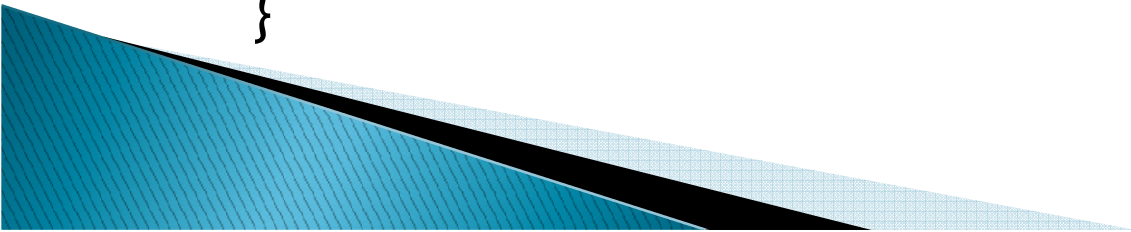
```
#include <stdio.h>
main ( )
{
    int i = 1;
    for ( ; 1 <= 10; i ++ )
        printf ("%d\n", i);
}
```



Estruturas Iteração– FOR

- ▶ Trata-se de um recurso que permite ao usuário utilizar uma estrutura if dentro de outra obtendo, assim, diversas respostas possíveis.

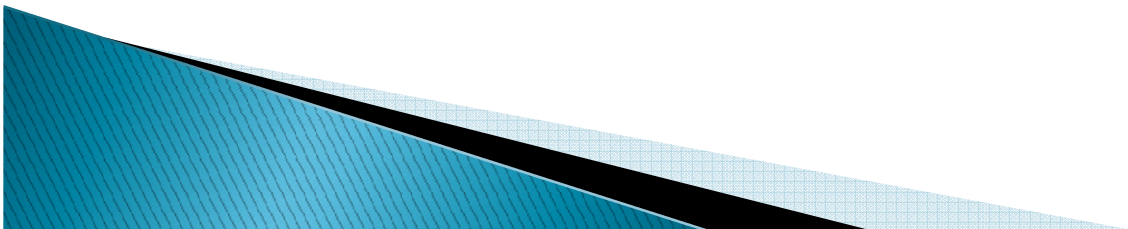
```
#include "stdio.h"
#include "conio.h"
main ( )
{
    int i, j;
    clrscr( );
    for ( i =1; i < 10; i ++ )
        for ( j = 1; j < 10; j ++ )
            printf ( "\n%d x %d = %d", i, j, i * j);
}
```



Estruturas de Iteração– While

- ▶ É o mais genérico dos três e pode ser usado para substituir os outros dois; em outras palavras, o laço while supre todas as necessidades. Vamos analisar o exemplo a seguir:

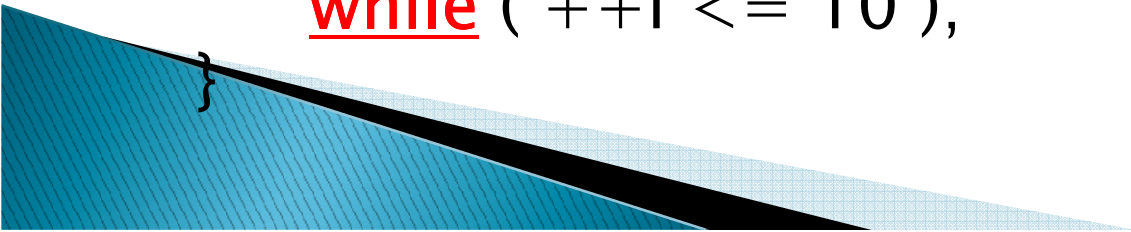
```
#include <stdio.h>
#include <conio.h>
main ( )
{
    int x, y;
    x = y = 0;
    while ( y < 10 )
        x += ++y;
    clrscr( );
    printf ("\nx = %d\n y = %d\n", x, y);
}
```



Estruturas de Iteração– For While

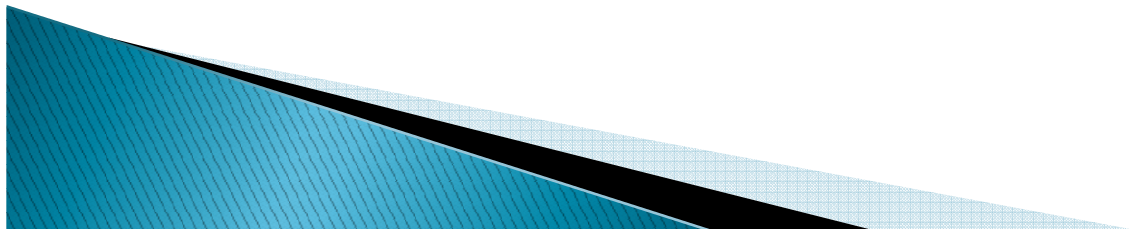
O comando do ... while é semelhante ao comando while. A diferença está no momento da avaliação da expressão, o que ocorre sempre após a execução do comando. Isto faz com que o comando do laço do ... while sempre execute pelo menos uma vez antes de realizar tal teste.

```
#include <stdio.h>
main ( )
{
    int i = 1;
    do
        printf ("%d\n", i);
    while ( ++i <= 10 );
}
```



Comandos de interrupção

- ▶ **Break**: quando utilizado em um bloco de comandos, associado a um for, while ou do ... while, faz com que o laço seja imediatamente interrompido, transferindo o processamento para o primeiro comando seguinte do laço.
- ▶ **Continue**: funciona de forma semelhante ao comando break. Ao invés de interromper a execução do laço, como o comando break, o comando continue pula as instruções que tiverem abaixo e força a próxima iteração do laço.
- ▶ **Goto**: provoca o desvio da execução do programa para algum outro ponto dentro do código fonte. Como, hoje em dia, todos os programas seguem as técnicas de programação estruturadas, o uso do goto não é recomendável.

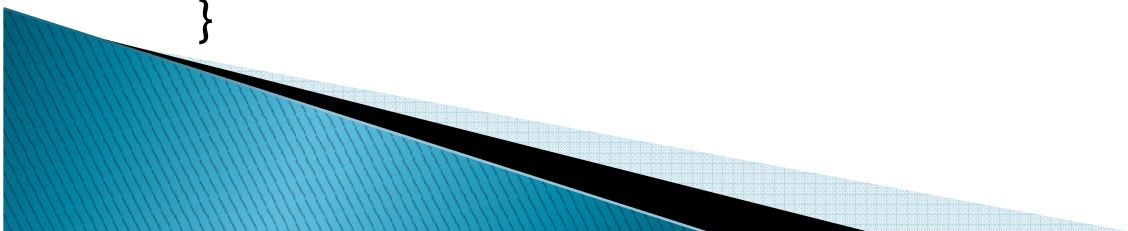


Comandos de interrupção

Exemplos

```
#include <stdio.h>
#include <conio.h>
main ( )
{
    char ch;
    int i;

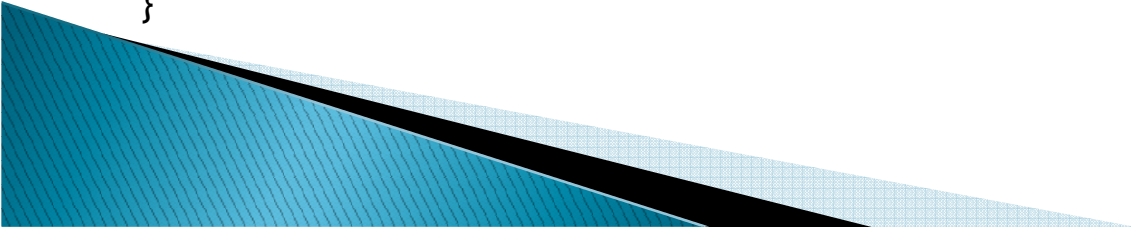
    for ( i = 0; i < 10; i ++ )
    {
        ch = getch( );
        if ( ch == '\x1b' ) /* character escape ESC */
            break;
        printf ( "\n%c", ch );
    }
    puts ( "\nAcabou" );
    getch( );
}
```



Comandos de Interrupção

Exemplos

```
#include <stdio.h>
#include <conio.h>
main ( )
{
    char
    ch;
    int
    i;
    for ( i = 0; i < 10; i ++ )
    {
        ch = getch( );
        if ( ch == '\x1b' )
            continue;
        printf ( "\n%c", ch );
    }
    clrscr( );
    puts ( "\n Acabou" );
    getch( );
}
```



Comandos de Interrupção

Exemplos

```
#include <stdio.h>
/* Simulação do uso do comando goto */
main ( )
{
    .....
    while( )
    {
        .....
        if (erro )
            goto ERRO;
        .....
    }
    .....
}
ERRO:
/* representa o rótulo do desvio goto */
.....
/* rotina de tratamento de erro */
```

