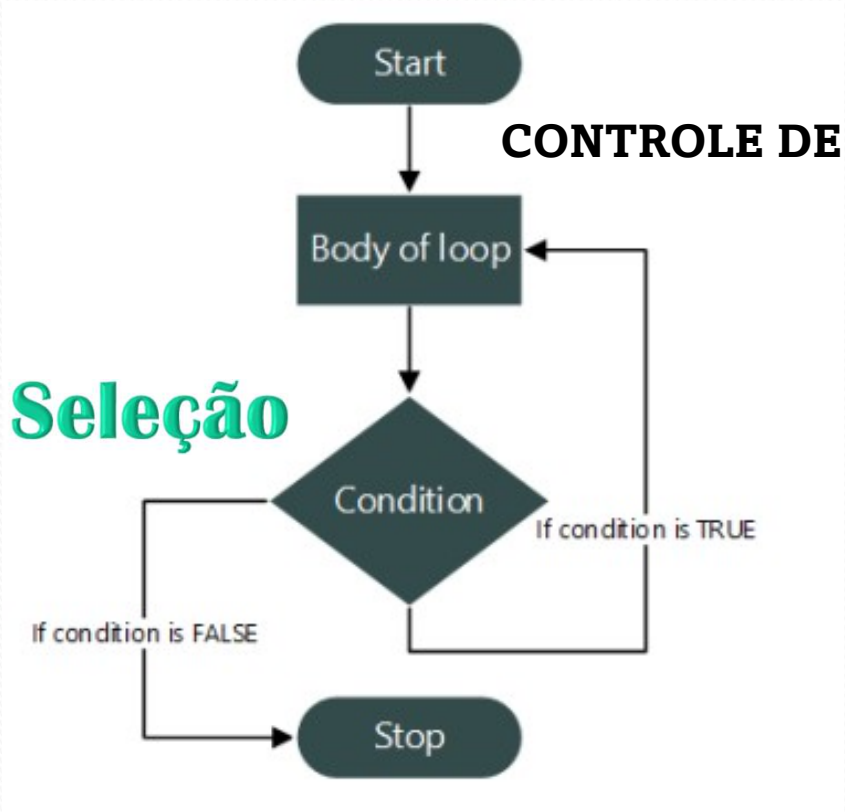
# Disciplina : Programação Estruturada

## CONTROLE DE FLUXO DE PROGRAMA

## Estrutura de Seleção




*Prof. Esp. Adalberto Sanches Munaro*

# Controle de Fluxo de Programa

Decidir se deve ou não executar algum código, dependendo se uma condição é verdadeira e decidir executar algum código repetidamente enquanto uma condição é verdadeira, são blocos de construção básicos na maioria das linguagens de programação.

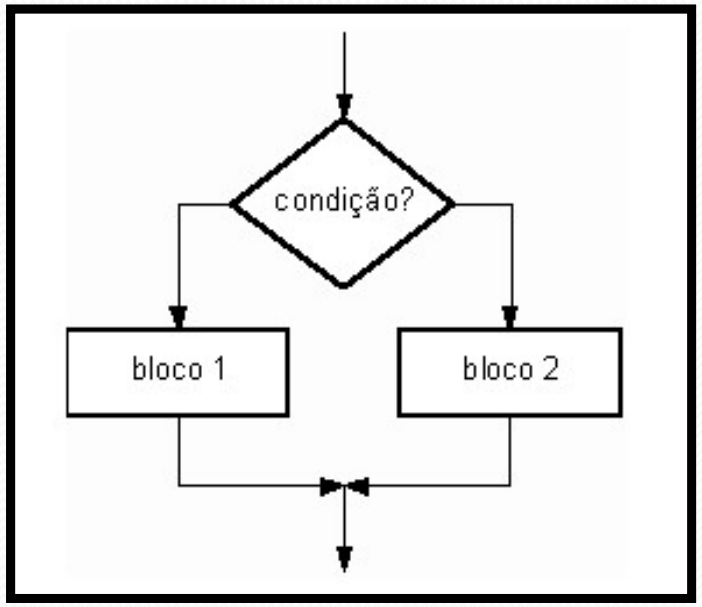
As construções mais comuns que permitem controlar o fluxo de execução do código C, são as expressões if e CASE e laços de repetição (While e For).

As estruturas **de controle de fluxo permitirão** desenvolver algoritmos capazes de testar expressões e, a partir delas, seguir pelas instruções de forma não linear

# Controle de Fluxo de Programa

## Estruturas de controle de fluxo de programa

são aplicadas à seqüência de comandos de maneira a condicionar a execução de trechos específicos de programa, ao resultado de uma expressão lógica.



# Controle de Fluxo de Programa

## Operadores Relacionais

| Operadores | Significado | Exemplo | Significado da expressão |
|---|---|---|---|
| > | Maior | a > b | **a** é maior que **b**? |
| >= | Maior ou igual | a >= b | **a** é maior ou igual a **b**? |
| < | Menor | a < b | **a** é menor que **b**? |
| <= | Menor ou igual | a <= b | **a** é menor ou igual a **b**? |
| == | Igual | a == b | **a** é igual a **b**? |
| != | Diferente | a != b | **a** é diferente de **b**? |

As expressões que contenham um operador relacional reduzem-se sempre a um valor lógico TRUE (1) or FALSE (0)

# Controle de Fluxo de Programa

## Operadores Lógicos

Usados nas circunstâncias em que uma condição não é suficiente para tomar uma decisão

| Operadores | Significado |
|---|---|
| && | AND |
| \|\| | OR |
| ! | NOT |

Exemplos:

```
Se c = 'S' || c = 's'
   imprimir ("solteiro");

Se nota >15 && nota < 20
   imprimir ("Bom");

Se nota >15 && !(nota > 20)
   imprimir ("Bom");
```

| Expressão1 | Expresão 2 | Expressão && Expressão 2 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| Expressão1 | Expresão 2 | Expressão \|\| Expressão 2 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

| Expressão | ! Expressão |
|---|---|
| 0 | 1 |
| 1 | 0 |

# Estrutura de Seleção

A **estrutura de seleção** (decisão) sem dúvida, é a mais utilizada na programação.

O **objetivo é identificar** o conteúdo de uma condição e direcionar o fluxo do programa para um determinado cálculo, rotina, desvio, função etc..

# Estrutura de Seleção

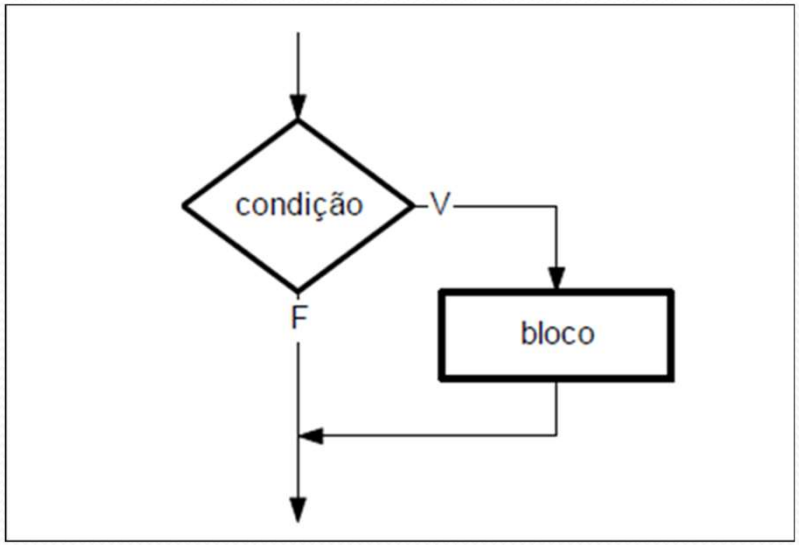
## Estrutura Simples ( if....)

Esta estrutura é aplicada quando um segmento deve ser executado somente se o resultado da expressão for **Verdadeiro**.

O restante da sequencia de comandos será executado normalmente.

O segmento condicionado à estrutura pode ser um comando **simples** ou um comando **composto**.

Neste caso, o comando deve ser delimitado {}.



# Estrutura de Seleção

## Estrutura Simples ( if....)

Sintaxe:

**if** (condição)
***comando;*** ***{ Simples }***

***Com mais de um comando:***

**if** (condição)
{
***comando1;***
***comando2;*** ***{ Composto }***
}

# Estrutura de Seleção

## Estrutura Simples ( if....)

```
/* Comando Simples */
#include <stdio.h>

main ()
{
  int num;
  printf ("Digite um numero: ");
  scanf ("%d",&num);
  if (num>10)
     printf ("\n\nO numero e' maior que 10");
  }
```

# Estrutura de Seleção

## Estrutura Simples ( if....)

```
/* Comando Composto */
#include <stdio.h>
main ()
{
  int num;
  printf ("Digite um numero: ");
  scanf ("%d",&num);
  if (num==10)
     {
       printf ("\n\nVoce acertou!\n");
       printf ("O numero e igual a 10.");
     }
}
```

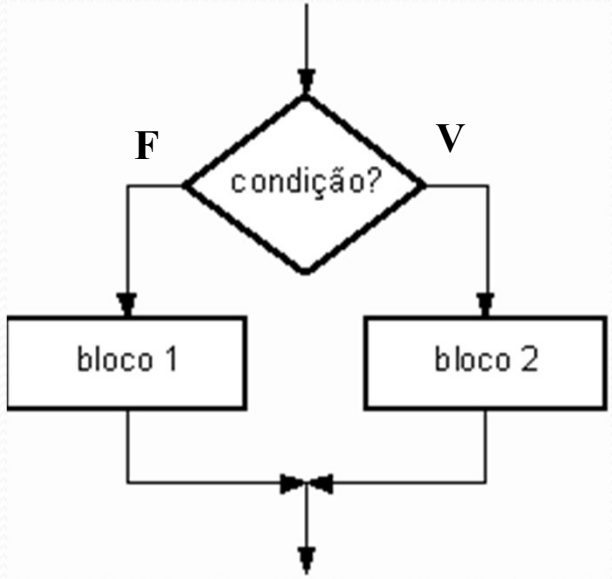
# Estrutura de Seleção

## Estrutura Composta ( if .. else )

Esta estrutura é aplicada quando dois segmentos de programa serão executados individualmente associados, um ao resultado **Verdadeiro** e outro ao resultado **Falso** da condição proposta.

O restante do programa mantém sua execução inalterada.

Vale também aqui, a definição de comandos simples e compostos.



# Estrutura de Seleção

## Estrutura Composta ( if .. else )

Sintaxe:

```
if (condição)
        comando1;    { verdadeiro }
   else
     {
        comando2;
                                  { falso }
        comando3;
     }
```

# Estrutura de Seleção

## Estrutura Composta ( if .. else )

```
/* Comando Composto */
#include <stdio.h>
main ()
{
  char sexo;
  printf ("Digite o sexo da pessoa F/M  ");
  fflsuh(stdin);
  sexo= getchar();
  if (sexo ==‘F’ || sexo ==‘f’ )
     printf ("\n\n Pessoa do sexo Feminino ");
  else
    printf ("\n\n Pessoa do sexo Masculino ");
  }
```

# Estrutura de Seleção

## Estrutura Encada ( if .. Else if )

Chamamos de estruturas de decisão encadeadas (**if´s aninhados**), quando uma estrutura de decisão está localizada dentro do lado falso da outra. Este tipo de estrutura também é conhecida como seleção "aninhada" ou seleção "encaixada".

Qualquer que seja o termo usado para identificar a estrutura, o importante é que esse formato com uma estrutura de seleção dentro da outra permite fazer a escolha de apenas um entre vários comandos possíveis.

```
if (expressao)
    comando1;
else if (expressao)
    comando2;
else if(expressao)
    comando3;
```

# Controle de Fluxo de Programa

```
/* if´s aninhados */
#include <stdio.h>
main ()
{
  int num;
  printf ("Digite um numero: ");
  scanf ("%d",&num);
  if (num>10)
     printf ("\n\nO numero e' maior que 10");
  else if (num< 10)
      printf ("\n\nO numero e' menor que 10");
   else
         printf ("O numero e’ igual a 10.");
  }
```

# Estrutura de Seleção

## A DECLARAÇÃO switch .. case

Ainda que o encadeamento ***if-else-if possa realizar testes de múltipla*** escolha, ele quase nunca é elegante.

O código pode ser muito difícil de acompanhar e pode confundir até mesmo o seu autor.

Por esse motivo, a linguagem C tem internamente uma declaração de decisão de múltipla escolha chamada de ***switch.***

# A DECLARAÇÃO switch .. case

A forma geral da declaração switch é:

```
switch (variável) {
   case constante1:
      commandos;
      break;
   case constante2:
      commandos;
      break;
   case constante3:
      commandos;
      break;

default
    commandos;
}
```

# Estrutura de Seleção

## A DECLARAÇÃO switch .. case

```
main ()
 {
int num;
printf ("Digite um numero: ");
scanf ("%d",&num);
switch (num)
 {
   case 9:
     printf ("\n\nO numero e igual a 9.\n");
     break;
   case 10:
     printf ("\n\nO numero e igual a 10.\n");
     break;
   case 11:
     printf ("\n\nO numero e igual a 11.\n");
     break;
   default:
     printf ("\n\nO numero nao e nem 9 nem 10 nem 11.\n");
  }
}
```