Estruturas de Decisão

Disciplina: Algoritmos e Programação
Curso: Engenharia de Computação

Professora: Mariza Miola Dosciatti
mariza@utfpr.edu.br

Objetivos

- Entender o conceito de estrutura de decisão.
- Entender a forma de aninhamento de blocos de comandos.
- Saber utilizar os operadores lógicos e relacionais em estruturas de controle.
- Saber utilizar a estrutura de decisão no desenvolvimento de algoritmos computacionais.

- Item da ementa (Plano de Ensino):
  - Estruturas de controle de fluxo: decisão.

Sumário

1. Estrutura de decisão
    1.1. Operadores relacionais
    1.2. Operadores lógicos
    1.3. Precedência de operadores
    1.4. Estrutura de decisão

1. Estrutura de Decisão

- **Problema:**
  - Fazer a média de duas notas e informar se o valor obtido é maior ou igual a seis ou se é menor que seis.
  - Ações:
    - Entrada
      - Procedimentos de entrada iguais a estrutura sequencial.
    - Processamento
      - Implementar a fórmula para cálculo da média.
      - Verificar se valor obtido maior ou igual a seis ou se é menor que seis (condição).
    - Saída
      - Emitir mensagem informado se valor obtido é maior ou menor que seis.

1. Estrutura de Decisão (cont.)

- **Para verificar uma condição** é necessário realizar um **teste lógico**. Esse teste é realizado por meio **operadores relacionais**.
- **Expressões relacionais** podem ser combinadas utilizando **operadores lógicos**.
- O **resultado** de uma expressão lógica será **verdadeiro** ou **falso**. Independentemente da notação utilizada será sempre a exclusão mútua entre duas opções.

1.1. Operadores relacionais

== igual a

!= diferente de

> maior que

< menor que

>= maior ou igual a

<= menor ou igual a

Var X	Var Y	Teste lógico	Resultado
X=2	Y=3	X == Y	Falso
X=2	Y=3	X != Y	Verdadeiro
X=2	Y=3	X >= Y	Falso
X=2	Y=3	X < Y	Verdadeiro

1.2. Operadores lógicos

! negação lógica

&& "e" lógico, conjunção

|| "ou" lógico, disjunção

Condição1 Var1 = 3	Condição2 Var2 = 3	Condição1 && Condição2	Condição1 | | Condição2	! Condição1
Var1 > 2 **V**	Var2 > 1 **V**	**V**	**V**	**F**
Var1 > 2 **V**	Var2 == 2 **F**	**F**	**V**	**F**
Var1 > 5 **F**	Var2 > 2 **V**	**F**	**V**	**V**
Var1 > 5 **F**	Var2 < 2 **F**	**F**	**F**	**V**

1.2. Operadores lógicos (cont.)

- Em C, o resultado da comparação será o valor **0** se o resultado de um teste lógico é **FALSO** e um valor **1** se resultado de um teste lógico é **VERDADEIRO**. Exemplo:

```c
#include <stdio.h>

int main(void)
{
	int verdadeiro, falso;

	verdadeiro = (15 < 20);
	falso = (15 == 20);

	printf("Verdadeiro: %d\n", verdadeiro);
	printf("Falso: %d\n", falso);

	return 0;
}
```

```
Verdadeiro: 1
Falso: 0
```

1.2. Operadores lógicos (cont.)

- **Condição de controle**
  - Faz o controle de **qual** conjunto de instruções será executado ou **quantas vezes** o conjunto será executado.
  - Uma condição de controle é uma expressão lógica ou aritmética cujo resultado pode ser considerado **verdadeiro** ou **falso**.
  - **Exemplo:** Considere as variáveis:

```
int i=0, j=3;
```

Condição	Valor numérico	Significado lógico
(i == 0)	**1**	Verdadeiro
(i > j)	**0**	Falso
(i)	**0***	Falso
(j)	**1**	Verdadeiro

* 0 é falso e qualquer valor diferente de 0 é verdadeiro.

1.2. Operadores lógicos (cont.)

- **Operador condicional ternário ? :**
  - Opera sobre três expressões:

    Exp 1 **?** Exp 2 **:** Exp 3;

Exemplo:

```
max = (a > b) ? a : b;
```

A variável que contém maior valor entre `a` e `b` será atribuída a `max`.

```
Valor de max: 3
```

1.2. Operadores lógicos (cont.)

- **Operador condicional ternário ? :**
  Exemplo:

```c
#include <stdio.h>

int main(void)
{
    int numero;

    printf("Informe um numero: ");
    scanf("%d", &numero);

    numero >= 0 ? numero++ : numero--;

    printf("O novo valor de numero eh: %d", numero);

    return 0;
}
```

```
Informe um numero: 6
O novo valor de numero eh: 7
```

1.2. Operadores lógicos (cont.)

- **Operador condicional ternário ? :**
  Exemplo:

```c
#include <stdio.h>

int main(void)
{
    int numero;

    printf("Informe um numero: ");
    scanf("%d", &numero);

    numero % 2 == 0 ? printf("Par") : printf("Impar");

    return 0;
}
```

```
Informe um numero: 5
Impar
```

1.3. Precedência de operadores

- O uso de parênteses altera a precedência dos operadores.
- Operadores de mesma precedência são executados da esquerda para a direita.

Maior precedência	`() []`
	`! ++ -- -(unário) (cast)`
	`* / %`
	`+ -` `< <= >= >`
	`== !=`
	`&&`
	`||`
	`? :`
	`= += -= *= /= %=`
Menor precedência	Avaliação sequencial, da esquerda para a direita

3.1.4. Estrutura de decisão

- Uma **estrutura de decisão** permite decidir se um conjunto de instruções será ou não executado de acordo com determinadas condições.
- A **decisão** é feita com base no resultado de um **teste lógico** que determina a condição.
- As principais estruturas de decisão são:
  - if
  - if ... else
  - switch ... case

1.4. Estrutura de decisão (cont.)

- **if**
  - Sintaxe:

```
if(condição)
{
    //conjunto de instruções;
}
```

  - onde:
    - *condição* é uma expressão com resultado lógico.
    - *conjunto de instruções* são as instruções (bloco de comandos) a serem executadas.
- Esta estrutura permite que se execute (ou não) um conjunto de instruções conforme o valor de uma condição seja verdadeiro ou falso.

1.4. Estrutura de decisão (cont.)

- **if**

Palavra-chave

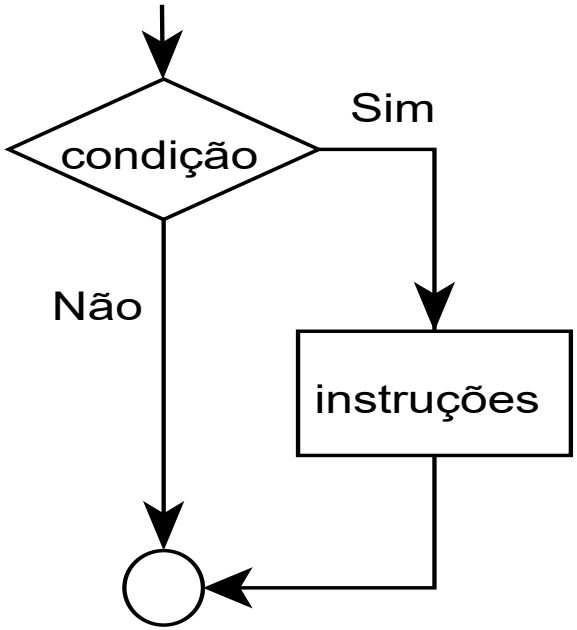
Teste lógico (variáveis, constantes, expressões aritméticas, retorno de função e operadores lógicos e relacionais)

```
if(condição)
{
    /*instruções*/
}
```

O que será realizado se o resultado do teste lógico for verdadeiro

1.4. Estrutura de decisão (cont.)

- **if**



onde:

- *condição* é uma expressão com resultado lógico.
- *instruções são os* comandos a serem executados.

1.4. Estrutura de decisão (cont.)

- Exemplos:

```
if(n == 5)
{
   printf("%d é igual a 5", n);
}

if(n < 5)
{
   printf("%d é menor que 5", n);
}

if(n <= 5)
{
 printf("%d é menor ou igual a 5", n);
}
```

1.4. Estrutura de decisão (cont.)

- Exemplos:

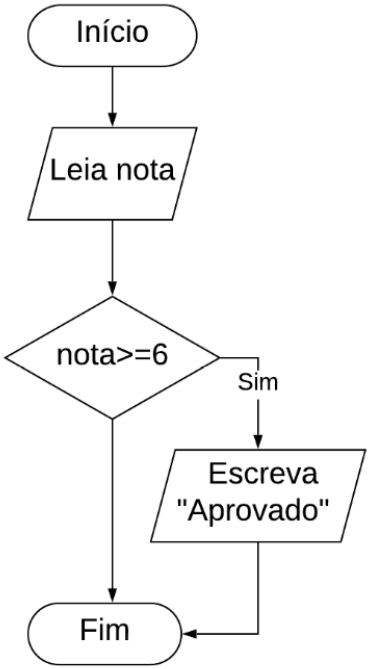
```
if(n < 5 || n > 9)
{
   printf("%d está fora do intervalo entre 5 e 9.", n);
}

if(n >= 5 && n <= 9)
{
   printf("%d pertence ao intervalo entre 5 e 9.", n);
}
```

1.4. Estrutura de decisão (cont.)

- **Exemplo:**
  - Verificar se o valor da variável `nota` é maior ou igual a 6. Se sim, informar `"Aprovado"`.

Algoritmo



Em C:

```c
#include <stdio.h>

int main(void)
{
    float nota;

    printf("Informe a nota: ");
    scanf("%f", &nota);

    if(nota >= 6)
    {
        printf("Aprovado");
    }
    return 0;
}
```

1.4. Estrutura de decisão (cont.)

- **if else**
  - Sintaxe:

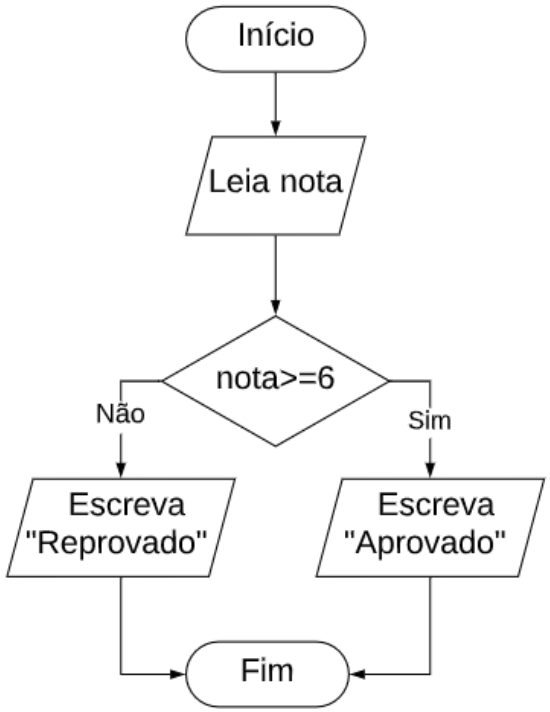
```
if(condição)
{
      //conjunto de instruções 1;
}
else
{
     //conjunto de instruções 2;
}
```

- onde:
  - *condição* – teste lógico realizado.
  - instruções 1 – comandos (instruções) que são executados se o resultado da condição for **verdadeiro**.
  - instruções 2 – comandos (instruções) que são executados se o resultado da condição for **falso.**

1.4. Estrutura de decisão (cont.)

- **Exemplo:**
  - Verificar se o valor da `nota` é maior ou igual a 6. Se sim, informar `"Aprovado"` se não, informar `"Reprovado"`.

Algoritmo:



Em C:

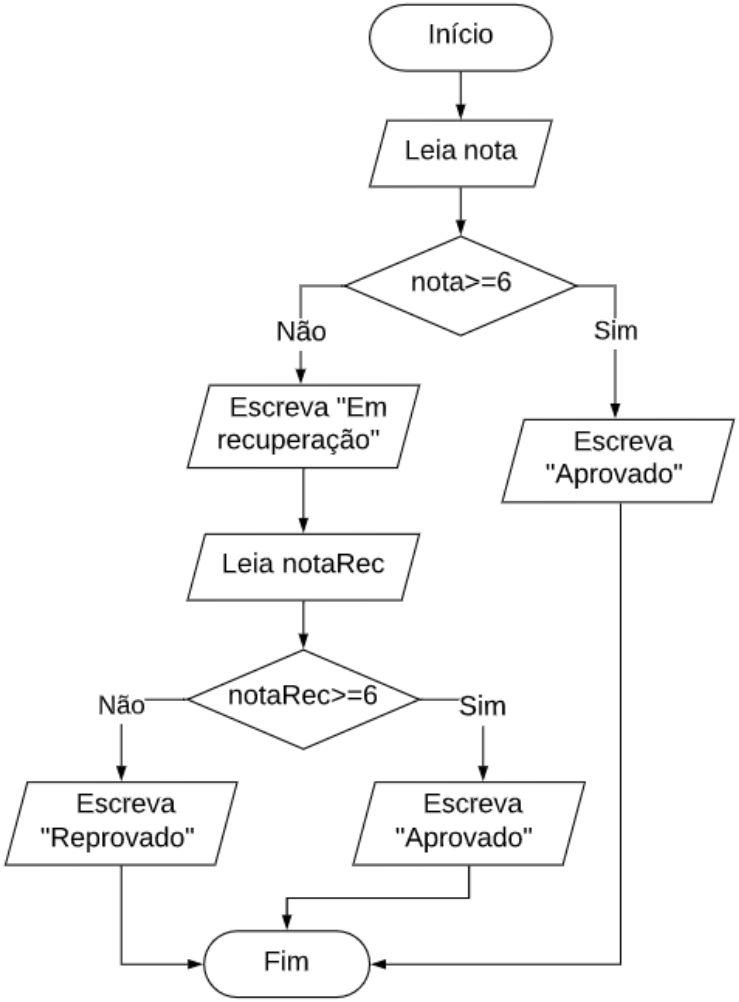
```c
#include <stdio.h>
int main(void)
{
    float nota;

    printf("Informe a nota: ");
    scanf("%f", &nota);

    if(nota >= 6)
    {
        printf("Aprovado");
    }
    else
    {
        printf("Reprovado");
    }
    return 0;
}
```

1.4. Estrutura de decisão (cont.)

Algoritmo:



Em C:

```c
#include <stdio.h>
int main(void)
{
    float nota, notaRec;
    printf("Informe a nota: ");
    scanf("%f", &nota);

    if(nota >= 6)
    {
        printf("Aprovado\n");
    }
    else
    {
        printf("Em recuperacao\n");
        printf("Informe nota de recuperacao: \n");
        scanf("%f", &notaRec);

        if(notaRec >= 6)
        {
            printf("Aprovado apos recuperacao\n");
        }
        else
        {
            printf("Reprovado");
        }
    }
    return 0;
```

1.4. Estrutura de decisão (cont.)

- **Exemplo**
  - if ... else if .. else

Verificar se o valor da nota:

a) É maior ou igual a 6

b) Está entre 5 e 6

c) É menor que 5

Em C:

```c
#include <stdio.h>

int main(void)
{
    float nota;

    printf("Informe a nota: ");
    scanf("%f", &nota);

    if(nota >= 6)
    {
        printf("Aprovado\n");
    }
    else if(nota > 5 && nota < 6)
    {
        printf("Em recuperacao");
    }
    else
    {
        printf("Reprovado");
    }
    return 0;
}
```

1.4. Estrutura de decisão (cont.)

- **switch...case** - Permite a execução de um conjunto de instruções conforme o resultado de uma expressão de controle. O resultado desta expressão é comparado com o valor de cada um dos rótulos, e as instruções são executadas a partir deste rótulo.

- **Sintaxe:**

```
switch(variável)
{
     case rotulo1:
        //conjunto de instruções 1;
    case rotulo2:
        //conjunto de instruções 2;
     case rotulo3:
         //conjunto de instruções 3;
     default:
         //conjunto de instruções 4;
}
```

- **onde**:
  - *variável*
  - *rótulo é uma* constante inteira ou um caractere.
  - *conjunto de instruções são as* instruções (bloco de comandos) a serem executadas.

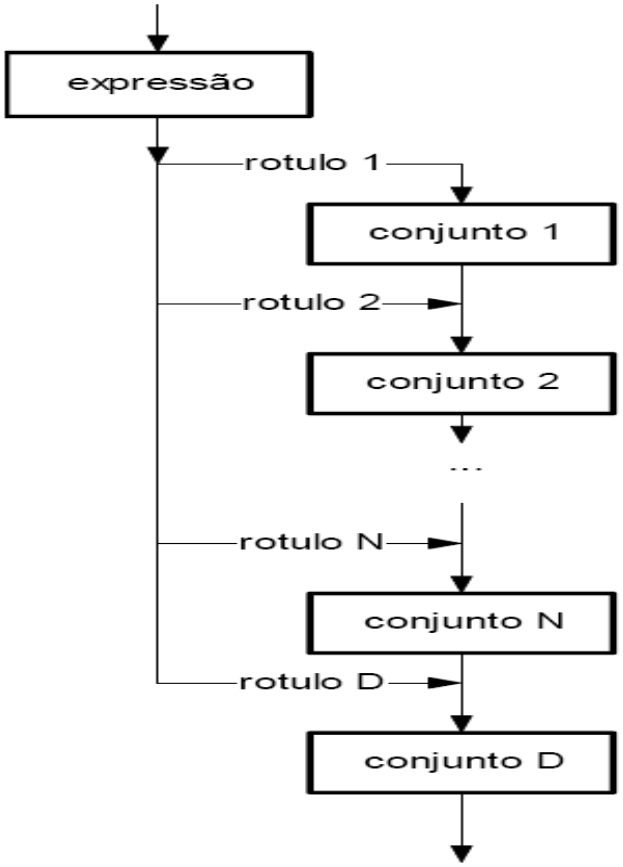
1.4. Estrutura de decisão (cont.)

- **switch...case**
  - O valor de *variável* é avaliado e o fluxo lógico será desviado para o conjunto cujo *rótulo* é igual ao resultado da variável e todas as instruções seguintes a este rótulo serão executadas.
  - Caso o resultado da variável seja diferente de todos os valores dos rótulos, o *conjunto* ***default*** é executado.
  - Os rótulos devem ser **diferentes** entre si.
  - ***default*** é opcional.

1.4. Estrutura de decisão (cont.)

- Exemplo da estrutura switch ... case

Algoritmo



Em C:

```c
#include <stdio.h>

int main(void)
{
    char tipo;

    printf("Selecione o sabor da pizza: ");
    scanf("%c", &tipo);

    switch(tipo)
    {
        case 'M':
            printf("Muzzarela");
            break;
        case 'C':
            printf("Calabreza");
            break;
        default:
            printf("Opcao incorreta");
    }
    return 0;
}
```

Referências

- DEITEL, P. J. DEITEL, H. M. **Como programar em C**. São Paulo: LTC, 1990.
- SCHILDT, H. **C Completo e total**, 3ª ed. São Paulo: Makron Books, 1996.
- MIZRAHI, V. V. **Treinamento em linguagem C: curso completo - módulo 1**. São Paulo: McGraw-Hill, 1990.
- Material baseado no conteúdo disponibilizado pela professora Beatriz Borsoi.

Dúvidas

- ???