



UNIVERSIDADE
FEDERAL DO CEARÁ



CK0179 – Programação Computacional para Engenharia: Comandos de Controle Condicional

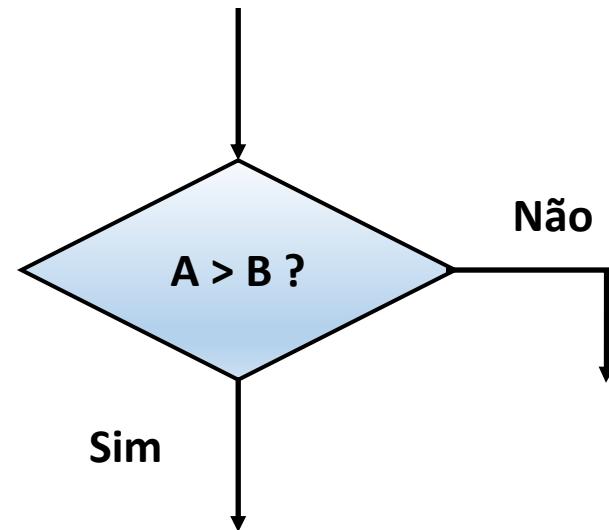
Prof. Maurício Moreira Neto

Objetivo

- Aprender a utilizar os comandos de controle condicional existentes na linguagem C
- Fazer exercícios que utilizem estes comandos de controle condicionais

Fluxogramas

- Condição ou decisão
 - Representados por losangos
 - Normalmente contém uma pergunta cuja a resposta é **Sim** ou **Não** (ou seja, um **teste de Verdadeiro ou Falso**)
 - Gera uma mudança de fluxo



Comando de Decisão - IF

- O comando IF é utilizado quando for necessário escolher entre dois caminhos, ou quando se desejar executar um comando sujeito ao resultado de um teste
- Esta estrutura examina uma ou mais condições e decide quais as instruções devem ser executadas com base na condição
- Em Pseudocódigo:

```
Se (condição) então
    Início
        Instruções
    Fim
```

Comando de Decisão - IF

- A forma geral de um comando **if** é:

```
if (condição) {  
    sequência de comandos;  
}
```

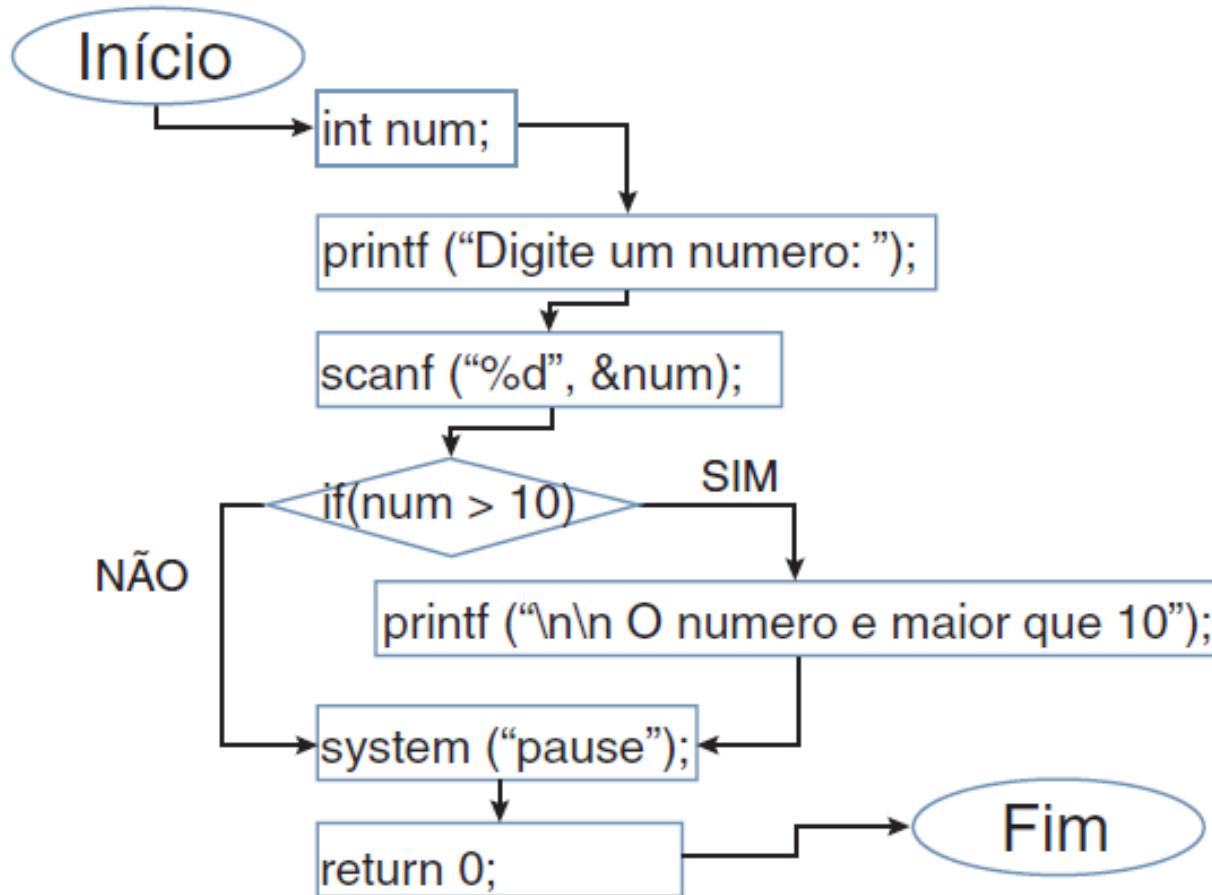
- A condição da expressão será avaliada desta maneira:
 - Se for zero (falsa), a declaração não será executada
 - Se a condição for diferente de zero (verdadeira) a declaração será executada

Comando de Decisão - IF

- Exemplo do uso do IF

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int num;
    printf("Digite um número: ");
    scanf("%d", &num);
    if (num > 10){
        printf("O número que digitou é maior que 10.\n");
    }
    return 0;
}
```

Comando de Decisão - IF



Comando de Decisão - IF

- A condição pode ser uma expressão usando operadores matemáticos, lógicos e relacionais
 - + , - , * , / , %
 - && , ||
 - > , < , >= , <= , == , !=
- Exemplo:
 - $(x > 10 \&\& y \leq x-1)$

Comando de Decisão - IF

- Tabela Verdade

- Os termos a e b representam o resultado de duas expressões relacionais

a	b	!a	!b	a && b	a b
0	0	1	1	0	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	1	1

Comando de Decisão - IF

- Pode-se usar chaves {} para delimitar o bloco de instruções que pertence ao **if**

```
if (num > 10){  
    printf("O numero que voce digitou e maior que 10.\n");  
}
```

- As chaves devem ser usadas no caso de mais de uma instrução:

```
if (nota >= 60){  
    printf("A nota é maior ou igual a 60.\n");  
    printf("O aluno está aprovado!");  
}
```

- As chaves podem ser ignoradas se a instrução for única

```
if (num > 10)  
    printf("O numero que voce digitou e maior que 10.\n");
```

Exercício

- Dada o valor da nota de um aluno, monte a expressão if que verifica se ele precisará fazer a prova final. O aluno deverá fazer prova final se sua nota for maior ou igual a 30 e menor do que 60.

Exercício – Resolução

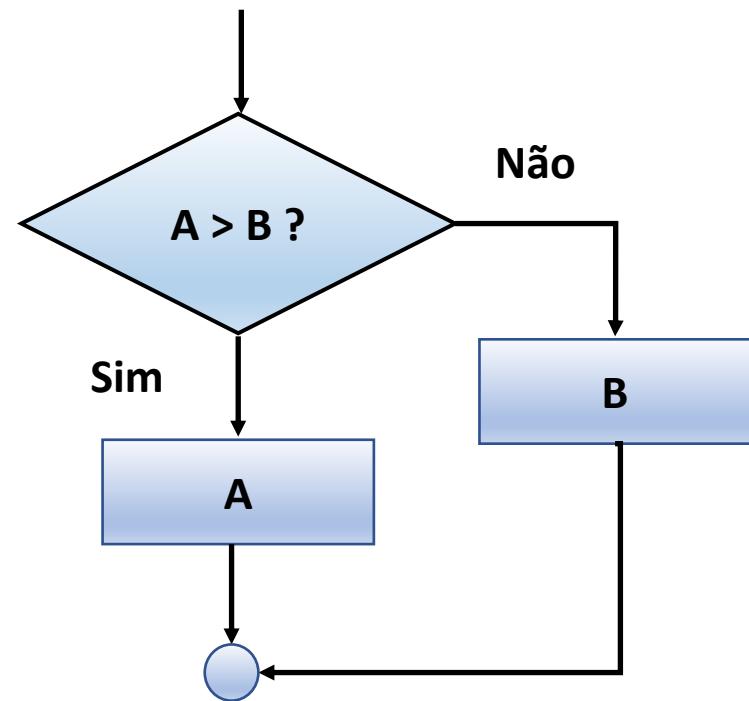
```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int nota;
    printf("Digite o valor da nota: ");
    scanf("%d", &nota);

    if (nota >= 30 && nota < 60) {
        printf("O aluno deverá fazer a prova final\n");
    }
    system("pause");
    return 0;
}
```

Comando ELSE

- O comando **else** pode ser entendido como sendo um complemento do comando **if**
 - Se o **if** diz o que fazer quando a condição é verdadeira, o **else** tratará da condição falsa



Comando ELSE

- O comando if-else tem a seguinte forma geral:

```
if /* condition */  
{  
    // Sequencia de comandos A  
} else {  
    // Sequencia de comandos B  
}
```

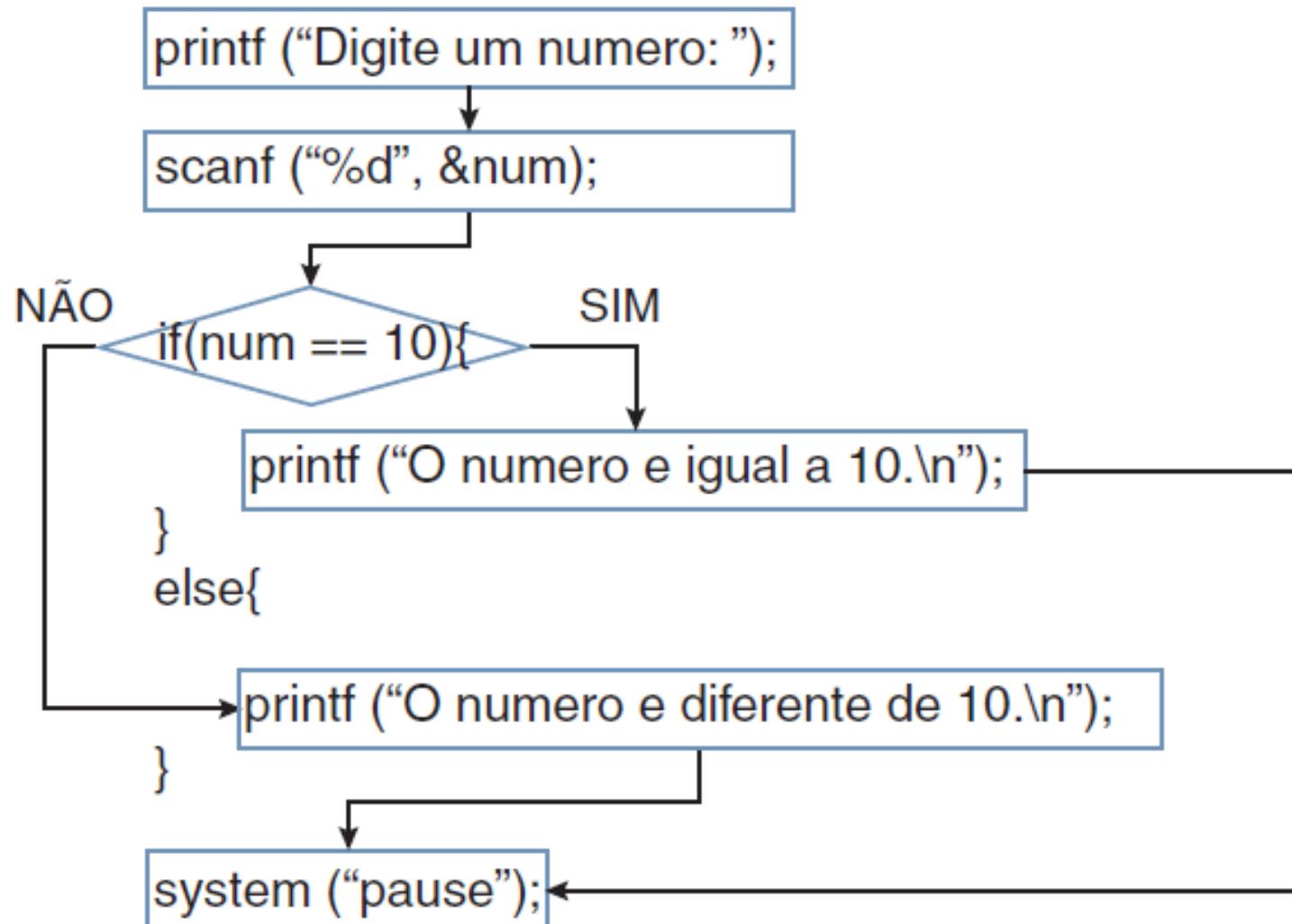
Comando ELSE

- A expressão da condição será avaliada:
 - Se for diferente de zero (verdadeira), a sequência de comandos A será executada
 - Se for zero (falso) a sequência de comandos B será executada
- Note que quando usamos a estrutura **if-else**, uma das duas declarações será executada
- Não há obrigatoriedade em usar o **else**

Exemplo IF-ELSE

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int numero;
    printf("Digite um numero: ");
    scanf("%d", &numero);
    if (numero == 10) {
        printf("O numero inserido e igual a 10\n");
    } else {
        printf("O numero inserido e diferente de 10\n");
    }
    return 0;
}
```

Exemplo IF-ELSE



Comando IF-ELSE

- Como no caso do comando **if**, as chaves podem ser ignoradas se a instrução contida no **else** for única

```
if(num == 10) {
    printf("O numero eh igual a 10.\n");
}else // else sem usar chaves
    printf("O numero eh diferente de 10.\n");
```

```
if(num == 10) {
    printf("O numero eh igual a 10.\n");
}else{ // else com chaves
    printf("O numero eh diferente de 10.\n");
}
```

Comando IF-ELSE

- O comando do if é independente do comando do else

```
if(num == 10) //if sem usar chaves
    printf("O numero eh igual a 10.\n");
else // else sem usar chaves
    printf("O numero eh diferente de 10.\n");
```

```
if(num == 10){ //if com chaves
    printf("O numero eh igual a 10.\n");
}else // else sem usar chaves
    printf("O numero eh diferente de 10.\n");
```

```
if(num == 10){ //if com chaves
    printf("O numero eh igual a 10.\n");
}else{ // else com chaves
    printf("O numero eh diferente de 10.\n");
}
```

```
if(num == 10) //if sem usar chaves
    printf("O numero eh igual a 10.\n");
else{ // else com chaves
    printf("O numero eh diferente de 10.\n");
}
```

Comando IF-ELSE

Certo

```
if (condicao) {  
    sequência de comandos;  
} else {  
    sequência de comandos;  
}
```

Errado

```
if (condicao) {  
    sequência de comandos;  
else  
    sequência de comandos;  
}
```



A sequência de comandos de **if** é independente da sequência de comandos de **else**. Cada comando tem o seu próprio conjunto de chaves ({}).

Aninhamento de IF

- O IF aninhado é simplesmente um **if** dentro da declaração de um outro **if** externo
 - A estrutura **if-else-if** é apenas uma extensão da estrutura **if-else**
- O único cuidado que devemos ter é o de saber exatamente a qual **if** um determinado **else** está ligado

Aninhamento de IF

```
if(condição){  
    instrução 1;  
    ...  
    instrução N;  
}else{  
    if(condição){  
        instrução 1;  
        ...  
        instrução N;  
    }else{  
        instrução 1;  
        ...  
        instrução N;  
    }  
}
```

```
if(condição){  
    if(condição){  
        instrução 1;  
        ...  
        instrução N;  
    }else{  
        instrução 1;  
        ...  
        instrução N;  
    }  
}else{  
    instrução 1;  
    ...  
    instrução N;  
}
```

Aninhamento de IF

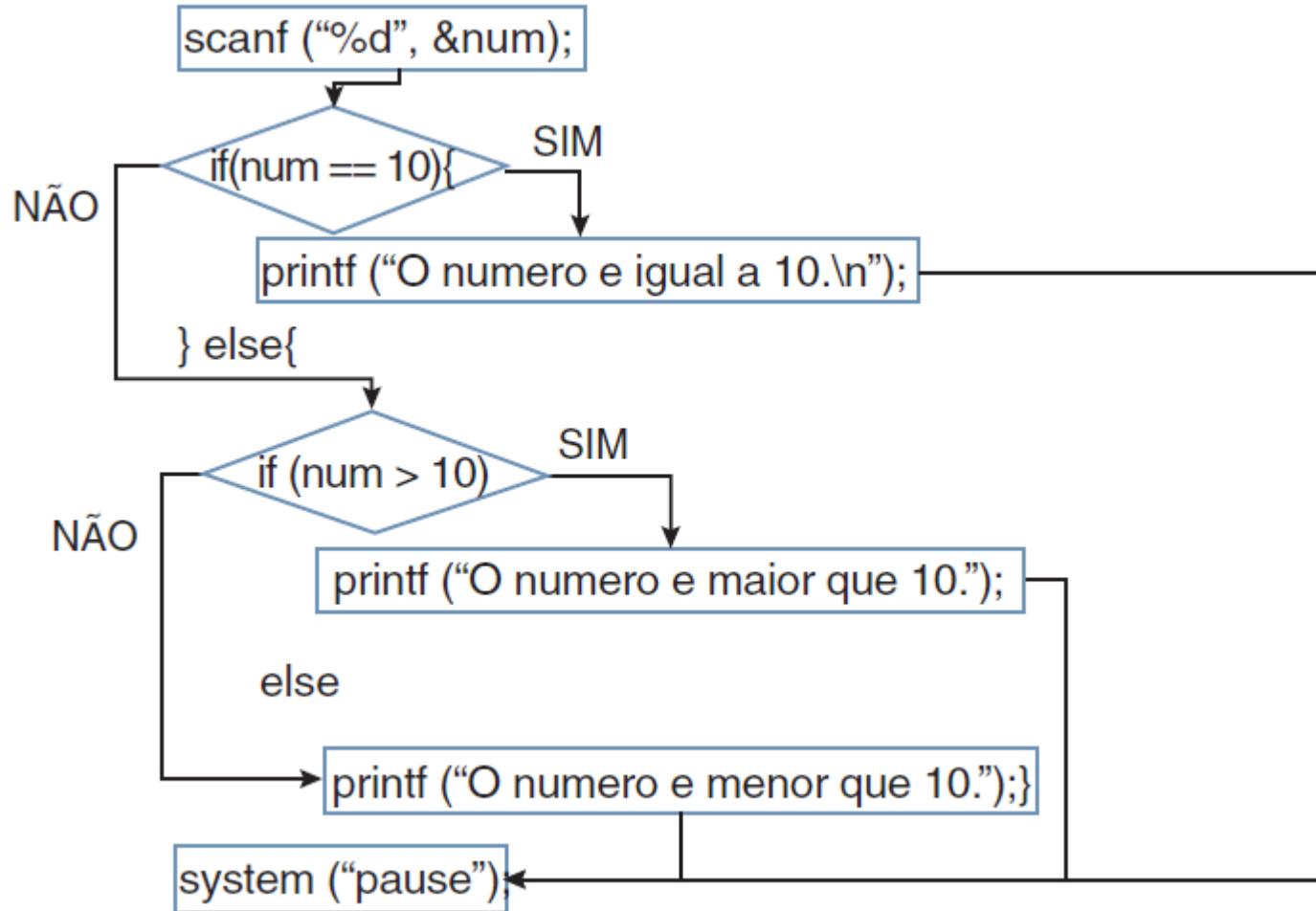
- O programa começa a testar as condições começando pela 1 e continua a testar até que ele ache uma expressão cujo resultado dê diferente de zero (verdadeiro)
- Neste caso:
 - Executa a sequencia de comandos correspondentes
 - Só uma sequencia de comandos será executada, ou seja, só será executada a sequencia de comandos equivalente a primeira condição que der diferente de zero
 - A última sequencia de comandos (default) é a que será executada no caso de todas as condições darem zero (falso) e é opcional

```
if(condição) {  
    instrução 1;  
    ...  
    instrução N;  
} else {  
    if(condição) {  
        instrução 1;  
        ...  
        instrução N;  
    } else {  
        instrução 1;  
        ...  
        instrução N;  
    }  
}
```

Exemplo de Aninhamento

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int numero;
    printf("Digite um numero: ");
    scanf("%d", &numero);
    if (numero == 10) {
        printf("O numero inserido e igual a 10\n");
    } else {
        if (numero < 10) {
            printf("O numero inserido e menor que 10.\n");
        } else {
            printf("O numero inserido e maior que 10.\n");
        }
    }
    return 0;
}
```

Exemplo de Aninhamento



Aninhamento de IF

- Observe sempre a correspondência entre **if's** e **else's**

```
if (condicao1)
    if (condicao2)
        comando if2;
else
    comando if1;
```

Errado, pois o comando if1 está associado ao segundo if, e não ao primeiro

```
if (condicao1) {
    if (condicao2)
        comando if2;
} else
    comando if1;
```

Correto, agora o comando if1 está associado ao primeiro if

Aninhamento de IF

- Não existe aninhamento de else's
 - Para cada else deve existir um if anterior, mas nem todo if precisa ter um else

```
if (cond1)
    comando if1;
else
    comando else1;
else
    comando else2;
```



Exercício

- Dada o valor da nota de um aluno, monte o conjunto de if's e else's que verifica se ele foi aprovado, reprovado ou precisará fazer a final.

Exercício

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int nota;
    printf("Digite a nota do aluno: ");
    scanf("%d", &nota);
    if (nota >= 70) {
        printf("Aluno esta aprovado\n");
    } else {
        if (nota < 40) {
            printf("O aluno reprovado\n");
        } else {
            printf("O aluno vai para final\n");
        }
    }
    return 0;
}
```

Exercício

- Construir a sequência de **if-else** para escrever o nome do dígito lido
 - ‘0’ -> “zero”;
 - ‘1’ -> “um”;
 - etc.

Exercício

- Construir a sequência de **if-else** para escrever o nome do dígito lido
 - ‘0’ -> “zero”;
 - ‘1’ -> “um”;
 - etc.

```
char ch;
scanf("%c", &ch);
if (ch == '0') printf("Zero");
else if (ch=='1') printf("Um");
else if (ch=='2') printf("Dois");
else if ...
else if (ch=='9') printf("Nove");
else printf("Nao era um digito!");
```

Expressão Condicional

- Quando o compilador avalia uma condição, ele quer um valor de retorno para poder tomar a decisão
- Esta expressão não necessita ser uma expressão no sentido convencional
- Uma variável sozinha pode ser uma “expressão” e esta retornar o seu próprio valor

Expressão Condicional

- Isto quer dizer que teremos as seguintes expressões

```
int num;  
if (num!=0)  
if (num==0)
```

- Equivalem a

```
int num;  
if (num)  
if (!num)
```

Importante

- Símbolo de atribuição = é diferente, muito diferente, do operador relacional de igualdade ==

```
int Nota;  
Nota == 60; // Nota é igual a 60?  
Nota = 50; // Nota recebe 50  
// Erro comum em C:  
// Teste se a nota é 60  
// Sempre entra na condição  
if (Nota = 60) {  
    printf("Você passou raspando!!");  
}  
// Versão Correta  
if (Nota == 60) {  
    printf("Você passou raspando!!");  
}
```

Importante

- Símbolo de atribuição = é diferente, muito diferente, do operador relacional de igualdade ==
- Por que sempre entra na condição?

```
if (Nota = 70) {  
    printf("Passou na conta hein!!");  
}
```

- Ao fazer **Nota = 70** (“Nota recebe 70”) estamos atribuindo um valor inteiro a variável Nota
- O valor atribuído **70 é diferente de Zero**. Como em C os booleanos são números inteiros, então vendo **Nota** como booleano, essa assume true, uma vez que é diferente de zero

O operador ?

- Também conhecido como operador ternário
- A expressão condição “? :” é uma simplificação do **if-else** utilizada tipicamente para atribuições condicionais

O operador

- Uma expressão como

```
if (a > 0) {  
    b = -199;  
} else {  
    b = 199;  
}
```

- Pode ser simplificada usando-se o operador ?
Da seguinte maneira

```
b = a > 0 ? -199 : 199;
```

Exercício

- Dado dois número x e y, retorne o maior na variável z:
 - Usando if-else
 - Usando o operador ternário

Exercício

Usando if-else

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int x, y, z;
    printf("Digite valor x: ");
    scanf("%d", &x);
    printf("Digite valor y: ");
    scanf("%d", &y);
    if (x > y)
    {
        z = x;
    } else {
        z = y;
    }
    printf("Maior = %d\n", z);
    return 0;
}
```

Usando operador ternário

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int x, y, z;
    printf("Digite valor x: ");
    scanf("%d", &x);
    printf("Digite valor y: ");
    scanf("%d", &y);
    z = x > y ? x : y;
    printf("Maior = %d\n", z);
    return 0;
}
```

O Operador ?

- O operador ? é limitado
 - Não atende a uma gama muito grande de casos
- Mas pode ser usado para simplificar expressões complicadas. Uma aplicação interessante é a do contador circular
 - `index = (index == 3) ? Index = 0: ++index;`

O comando SWITCH

- O comando SWITCH é próprio para se testar uma variável em relação a diversos valores pré-estabelecidos
 - Parecidos com **if-else-if**, porém não aceitas expressões, **apenas constantes**.
 - O switch testa a variável e executa a declaração cujo “case” corresponde ao valor atual da variável

O comando SWITCH

- Forma geral do comando switch

```
switch (expressão) {  
    case valor 1:  
        sequência de comandos 1;  
        break;  
    case valor k:  
        sequência de comandos k;  
        break;  
    ...  
    default:  
        sequência de comandos padrão;  
        break;  
}
```

O comando SWITCH

- O comando switch
 - Avalia o valor da **expressão** com os valor associados as cláusulas **case** em sequencia
 - Quando o valor associado a uma cláusula é igual ao valor da **expressão** os respectivos comandos são executados até encontrar um **break**
- A declaração **default** é opcional e será executada apenas se a **expressão** que está sendo testada não for igual a nenhuma das constantes presentes nos **case**

O comando SWITCH

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    char ch;
    printf("Digite um simbolo de pontuacao: ");
    ch = getchar();
    switch( ch ) {
        case '.':
            printf("Ponto.\n"); break;
        case ',':
            printf("Virgula.\n"); break;
        case ':':
            printf("Dois pontos.\n"); break;
        case ';':
            printf("Ponto e virgula.\n"); break;
        default :
            printf("Nao eh pontuacao.\n");
    }

    return 0;
}
```

O comando SWITCH

- O comando break

- Faz com que o switch seja interrompido assim que uma das sequencias de comando seja executada
- Não é essencial. Se após a execução da declaração não houver um break, o programa continuará executando o próximo comando case
- Isto pode ser útil em algumas situações, mas tenha cuidado

O comando SWITCH – sem break

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    char ch;
    printf("Digite um simbolo de pontuacao: ");
    ch = getchar();
    switch( ch ) {
        case '.':
            printf("Ponto.\n");
        case ',':
            printf("Virgula.\n");
        case ':':
            printf("Dois pontos.\n");
        case ';':
            printf("Ponto e virgula.\n");
        default :
            printf("Nao eh pontuacao.\n");
    }
    return 0;
}
```

O comando SWITCH – sem break

```
int num;
scanf ("%d", &num);
switch( num ) {
    case 0: printf ("0"); /* 0123456789 */
    case 1: printf ("1"); /* 123456789 */
    case 2: printf ("2"); /* 23456789 */
    case 3: printf ("3"); /* 3456789 */
    case 4: printf ("4"); /* 456789 */
    case 5: printf ("5"); /* 56789 */
    case 6: printf ("6"); /* 6789 */
    case 7: printf ("7"); /* 789 */
    case 8: printf ("8"); /* 89 */
    case 9: printf ("9"); /* 9 */
}
```

Exercício

- Construir o switch para escrever o nome do dígito lido
 - 0 -> “zero”;
 - 1 -> “um”;
 - etc

Exercício

```
switch(num) {  
    case 0: printf("Zero"); break;  
    case 1: printf("Um"); break;  
    case 2: printf("Dois"); break;  
    case 3: printf("Tres"); break;  
    case 4: printf("Quatro"); break;  
    case 5: printf("Cinco"); break;  
    case 6: printf("Seis"); break;  
    case 7: printf("Sete"); break;  
    case 8: printf("Oito"); break;  
    case 9: printf("Nove"); break;  
}
```

Dúvidas?



Referências

- André Luiz Villar Forbellone, Henri Frederico Eberspächer, Lógica de programação (terceira edição), Pearson, 2005, ISBN 9788576050247.
- Ulysses de Oliveira, Programando em C - Volume I - Fundamentos, editora Ciência Moderna, 2008, ISBN 9788573936599.
- **Slides baseados no material do site "Linguagem C Descomplicado"**
 - <https://programacaodescomplicada.wordpress.com/complementar/>