



Aula 04 - Linguagem C: Condicionais

Prof. Me. Claudiney R. Tinoco

`profclaudineytinoco@gmail.com`

Faculdade de Computação (FACOM)

Bacharelado em Ciência da Computação (BCC)

Bacharelado em Sistemas de Informação (BSI)

Programação Procedimental (PP)

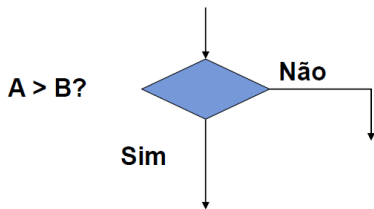
GBC014 - GSI002



FLUXOGRAMAS

○ Condição ou Decisão

- Representado por losangos
- Normalmente contém uma pergunta do tipo Sim/Não ou um teste de Verdadeiro/Falso.
- Mudança no fluxo





COMANDO IF

- Em linguagem C, o comando **if** é utilizado quando for necessário escolher entre dois caminhos, ou quando se deseja executar um comando sujeito ao resultado de um teste.



COMANDO IF

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

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

- A expressão, na condição, será avaliada:
 - Se ela for zero (falsa), a declaração não será executada;
 - Se a condição for diferente de zero (verdadeira) a declaração será executada.



EXEMPLO IF

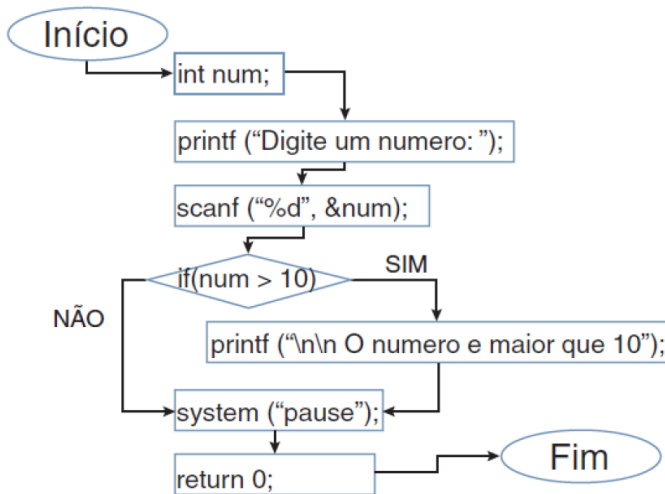
```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int num;
    printf("Digite um numero: ");
    scanf("%d",&num);

    if(num > 10){
        printf("O numero eh maior do que 10.\n");
    }

    return 0;
}
```



EXEMPLO IF





CONDIÇÃO DO IF

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



CONDIÇÃO DO 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 IF – USO DAS CHAVES {}

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

```
if (num > 10) {  
    printf ("\n\n O numero eh maior que 10");  
}
```

- 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 ("\n\n O numero e maior que 10") ;
```



EXERCÍCIO

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



EXERCÍCIO

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

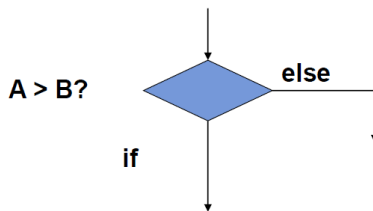
```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int num;
    printf ("Digite a nota: ");
    scanf ("%d", &num);
    if (num > 30 && num < 60)
        printf("O aluno deve fazer a prova sub \n");

    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 verdadeiro, o **else** trata da condição falsa.





COMANDO ELSE

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

```
if(condição) {  
    sequência de comandos 1;  
  
} else{  
    sequência de comandos 2;  
  
}
```



COMANDO ELSE

- A expressão da condição será avaliada:
 - Se ela for diferente de zero (verdadeiro), a seqüência de comandos 1 será executada.
 - Se for zero (falso) a seqüência de comandos 2 será executada.
- Note que quando usamos a estrutura if-else, uma das duas declarações será executada.
- Não há obrigatoriedade em usar o else

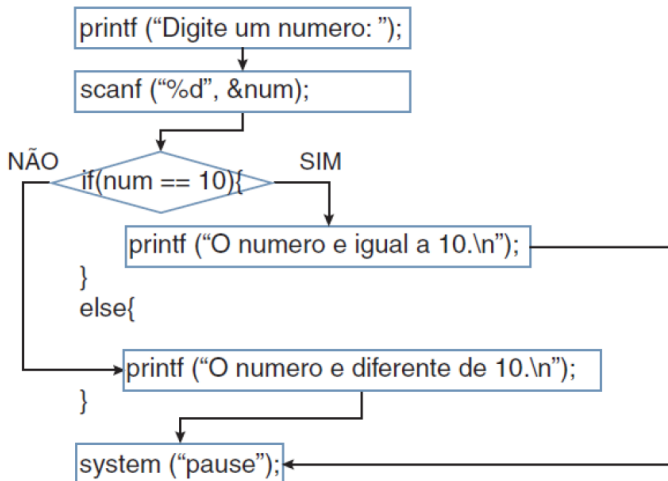


EXEMPLO IF-ELSE

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int num;
    printf("Digite um numero: ");
    scanf("%d",&num);
    if(num == 10){
        printf("O numero eh igual a 10.\n");
    }else{
        printf("O numero eh 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 ele
 - executa a sequência de comandos correspondente.
 - Só uma sequência de comandos será executada, ou seja, só será executada a sequência de comandos equivalente à primeira condição que der diferente de zero.
 - A última sequência 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 ANINHAMENTO

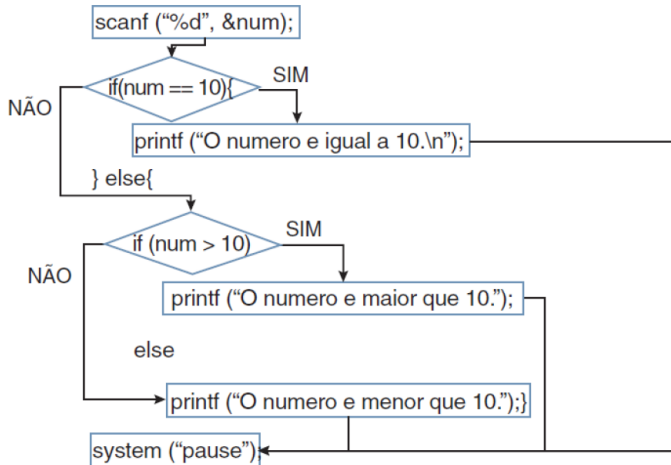
```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int num;
    printf("Digite um numero: ");
    scanf("%d", &num);

    if(num == 10) {
        printf("O numero eh igual a 10.\n");
    } else {
        if(num > 10)
            printf("O numero eh maior do que 10.\n");
        else
            printf("O numero eh menor do que 10.\n");
    }

    return 0;
}
```



EXEMPLO ANINHAMENTO





ANINHAMENTO DE IF

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

```
if (cond1)
```

```
    if (cond2)
```

```
        comando if2;
```

```
    else
```

```
        comando if1;
```

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

```
if (cond1) {
```

```
    if (cond2)
```

```
        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 sub.



EXERCÍCIO

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int d;
    printf("Digite a nota: ");
    scanf("%d",&d);
    if (d >= 60)
        printf("Aluno aprovado \n");
    else
        if(d < 30)
            printf("Aluno reprovado \n");
        else
            printf("O aluno deve fazer a prova sub \n");

    return 0;
}
```



EXERCÍCIO

- Construir a seqüência 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 = 60) {  
    printf("Você passou raspando!!");  
}
```

- Ao fazer **Nota = 60** (“Nota recebe 60”) estamos atribuindo um valor inteiro à variável Nota.
- O valor atribuído **60 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 condicional “?:” é uma simplificação do if-else utilizada tipicamente para atribuições condicionais



O OPERADOR ?

- Uma expressão como

```
if (a > 0)
    b = -150;
else
    b = 150;
```

- pode ser simplificada usando-se o operador ? da seguinte maneira:

```
b = a > 0 ? -150 : 150;
```



EXERCÍCIO

- Dado dois números 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 x:");
    scanf("%d", &x);
    printf("Digite 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 x:");
    scanf("%d", &x);
    printf("Digite 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.
 - Parecido com if-else-if, porém não aceita expressões, **apenas constantes**.
 - O switch testa a variável e executa a declaração cujo “case” corresponda 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 valores associados às cláusulas **case** em sequência;
 - 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**.



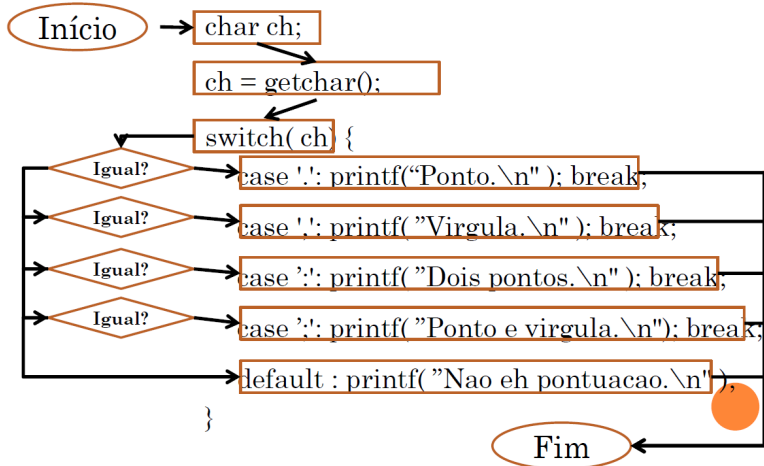
O COMANDO SWITCH

```
#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 SWITCH

○ O comando break

- Faz com que o switch seja interrompido assim que uma das sequência de comandos 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.



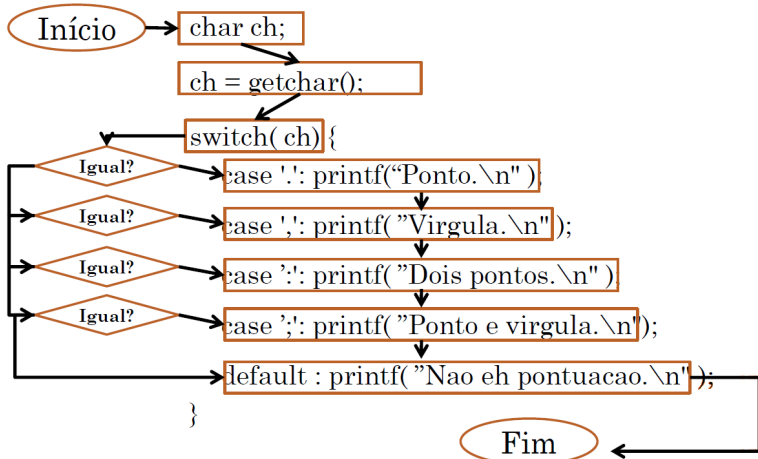
O COMANDO SWITCH SEM BREAK

```
#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





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

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

```
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;  
}
```



Referências

✓ Básica

- BACKES, André. *“Linguagem C: completa e descomplicada”*. Elsevier Brasil, 2013.
- DAMAS, Luís. *“Linguagem C”*. Grupo Gen-LTC, 2016.
- MIZRAHI, Victorine V. *“Treinamento em linguagem C”*, 2a. ed., São Paulo, Pearson, 2008.

✓ Extra

- BACKES, André. *“Programação Descomplicada Linguagem C”*. Projeto de extensão que disponibiliza vídeo-aulas de C e Estruturas de Dados. Disponível em: <https://www.youtube.com/user/progdescomplicada>. Acessado em: 25/04/2022.

✓ Baseado nos materiais do professor:

- Prof. André Backes (UFU)

Dúvidas?

Prof. Me. Claudiney R. Tinoco
profclaudineytinoco@gmail.com

Faculdade de Computação (FACOM)
Universidade Federal de Uberlândia (UFU)