ESTRUTURAS CONDICIONAIS EM C

NOTAS DE AULA 02

Uma **estrutura condicional** ou **estrutura de decisão** ou **comando de seleção** define uma condição em um programa, que permite que grupos de comandos sejam executados de maneira condicional, de acordo com o resultado da avaliação de um determinado teste (verdadeiro ou falso). Ou seja, programas utilizam estruturas condicionais para escolher entre cursos alternativos de ações. A seguir, serão apresentadas estruturas condicionais em C e também os operadores lógicos que podem ser utilizados nessas estruturas.

Importante: comandos de seleção são estruturas de controle básicas de qualquer linguagem de programação e devem por isso ser largamente estudadas e praticadas pelos alunos (ou seja, **os alunos são aconselhados a resolverem vários tipos de problemas diferentes usando estruturas condicionais**).

Estrutura condicional simples

A sintaxe de uma estrutura condicional simples em C é:

```
if (condição)
  comando;
```

O comando só será executado se a condição for verdadeira. Uma condição é uma comparação que possui dois valores possíveis: verdadeiro ou falso.

Em C, torna-se obrigatória a utilização de chaves quando existe mais de um comando a executar na cláusula **IF**, bastando para isso que sejam usados os comandos delimitadores de bloco **{** e **}**.

```
if (condição) {
  comando1;
  comando2;
  comando3;
}
```

Os comandos 1, 2 e 3 só serão executados se a condição for verdadeira.

Estrutura condicional composta

A sintaxe de uma estrutura condicional composta em C é:

```
if (condição)
  comando1;
else
  comando2;
```

A leitura do trecho acima é a seguinte: se a condição for verdadeira, será executado o comando1, caso contrário, será executado o comando2.

Assim como na estrutura simples, vários comandos podem ser executados na cláusula **if**, bastando para isso que sejam usados os delimitadores de bloco **{** e **}**.

```
if (condição) {
  comando1;
  comando2;
} else {
  comando3;
  comando4;
}
```

Se condição for verdadeira, os comandos 1 e 2 serão executados; se condição for falsa, os comandos 3 e 4 serão executados.

Operadores Lógicos

Os principais operadores lógicos são: &&, || e !, que significam E, OU e NÃO e são usados para conjunção, disjunção e negação, respectivamente.

Operador	Operação
!	não (negação)
&&	e (conjunção)
II	ou (disjunção)

Exemplos:

Expressão	Resultado
! (1 > 2)	1 (Verdadeiro)
(1 > 2) && (3 > 2)	0 (Falso)
(1 > 2) II (3 > 2)	1 (Verdadeiro)

Na linguagem C, todas as condições devem estar entre parênteses. Veja os exemplos a seguir.

```
if (x == 3)
  printf(“Numero igual a 3”);
```

No exemplo anterior, existe apenas uma condição que, obrigatoriamente, deve estar entre parênteses

.

```
if (x > 5 && x < 10)
  printf(“Numero entre 5 e 10”);
```

No exemplo anterior, existe mais de uma condição, as quais, obrigatoriamente devem estar entre parênteses

.

```
if ((x == 5 && y == 2) || (y == 3))
  printf(“x eh igual a 5 e y eh igual a 2, ou y eh igual a 3”);
```

No exemplo anterior, existe mais de uma condição e mais de um tipo de operador lógico, logo, além dos parênteses que envolvem todas as condições, devem existir ainda parênteses que indiquem a prioridade de execução das condições. Neste exemplo, as condições com o operador &&, ou seja, (x == 5 && y == 2), serão testadas e seu resultado será testado com a condição || (y == 3). No exemplo a seguir, as condições do operador || são testadas e seu resultado testado com a condição &&

.

```
if ((y == 2 || y == 3) && (x == 5))
  printf("x eh igual a 5 e y eh igual a 2, ou y eh igual a 3");
```

É importante notar que os operadores lógicos **e** e **ou** são representados por dois caracteres (**&&** e **||**). Em C existem operadores para manipular valores em nível de bit, chamados de operadores **bitwise**. No caso dos operadores bitwise é usado apenas um caractere para o e e para o ou (**&** e **|**). O uso de operadores bitwise está fora do escopo nesse momento, mas é importante conhecer a diferença de sintaxe entre eles e os operadores lógicos.

Decisão múltipla (Comando SWITCH)

A estrutura condicional **switch** possui uma lógica de funcionamento muito diferente de todas as variantes das estruturas **if** que estudamos até agora. Ela não avalia diversas condições para determinar qual opção executar, tal como era o caso das estruturas **if**. O **switch** é uma construção de múltiplas possibilidades de decisão. Ele compara o valor de uma variável (ou expressão aritmética que resulte em um inteiro) com uma série de valores constantes. A sintaxe desta estrutura é apresentada abaixo.

```
switch (variável) {
  case valor1:
    sentenças;
    ...
  case valor2:
    sentenças;
    ...
  case valor3:
    sentenças;
    ...
  default:
    sentenças;
    ...
}
```

Veja a seguir um exemplo de código que utiliza a estrutura condicional switch. Este programa lê uma variável do tipo *char* chamada *op*, avalia o valor informado e exibe uma mensagem de acordo com o mesmo.

```
#include <stdio.h>
int main(){
        char op;
        printf("Menu principal\n");
        printf("1 - Cadastrar cliente \n");
```

```
        printf("2 - Buscar cliente \n");
        printf("3 - Excluir cliente \n\n");
        printf("Digite uma opcao: ");
        op = getchar();
        switch (op) {
                case '1':
                        printf("Você escolheu cadastrar.\n");
                        break;
                case '2':
                        printf("Você quer buscar por um cliente.\n");
                        break;
                case '3':
                        printf("Você é muito malvado! Quer excluir um cliente!\n");
                        break;
                default:
                        printf("Opção inválida!\n");
        }
     return 0;
}
```

As instruções que pertencem à cláusula **default** são executadas quando o valor da variável de teste não coincide com nenhuma das cláusulas **case** definidas. Observe que no exemplo anterior como a variável de teste (*op*) é do tipo *char,* na estrutura do switch os valores devem ficar entre aspas simples. Se a variável de teste for do tipo *int*, as aspas simples não são necessárias, como é ilustrado no exemplo a seguir.

```
#include <stdio.h>

int main() {
        int x;
        printf("Menu principal");
        printf("1 - Cadastrar cliente \n");
        printf("2 - Buscar cliente \n");
        printf("3 ou 4- Excluir cliente \n");

        printf("Digite uma opcao: ");
        scanf("%d", &x);
        switch (x) {
                case 1:
                        printf("Você escolheu cadastrar.\n");
                        break;
                case 2:
                        printf("Você quer buscar por um cliente.\n");
```

```
                    break;
            case 3:
            case 4:
                    printf("Você é muito malvado! Quer excluir o cliente!\n");
                    break;
            default:
                    printf("Opcao invalida!\n");
        }
        return 0;
}
```

Outro ponto que merece destaque no exemplo anterior é o teste para os valores 3 e 4. Como não foi informada nenhuma instrução para o **case 3**, especialmente a instrução *break*, será executado o bloco de instruções do **case 4**. Assim, se x assumir o valor 3 ou o valor 4 serão executadas as mesmas instruções.