



Professor
Maromo

LINGUAGEM DE PROGRAMAÇÃO

Material 003



GitHub
maromo71

C



Agenda



Estruturas de Controle

- Bloco,
- Decisão,
- Seleção

Material: LP_003

Bloco



- Em C, um bloco é uma **coleção de zero ou mais declarações agrupadas entre chaves { }**. Ele é usado para agrupar várias instruções em uma única unidade lógica. Blocos são fundamentais na estruturação do código em C, especialmente quando combinados com estruturas de controle, como loops e condicionais.



Bloco - Características



- **Estruturas de controle:** múltiplas instruções realizadas com base em uma única condição ou controle dos laços.
- **Escopo:** Variáveis declaradas dentro de um bloco têm escopo local ao bloco.
- **Isolamento:** Blocos podem ser usados para isolar certas partes do código, mesmo fora de estruturas de controle,
- **Inicialização de Variáveis:** Em C99 e versões posteriores do padrão C, você pode declarar variáveis em qualquer lugar dentro de um bloco, não apenas no início.



Verdadeiro ou Falso



- **Zero é Falso:** Em C, o valor 0 é considerado falso. Isso se aplica tanto ao literal inteiro 0 quanto ao caractere '\0' (caractere nulo), que também tem valor de 0.
- **Tudo que Não é Zero é Verdadeiro:** Qualquer valor que não seja zero é considerado verdadeiro. Portanto, -1, 1, 100, -100, etc., todos são tratados como valores verdadeiros em contextos booleanos.



Operadores Relacionais em C

Operador	Descrição	Exemplo	Resultado (se $x = 5$ e $y = 10$)
<code>==</code>	Igual a	<code>x == y</code>	false
<code>!=</code>	Diferente de	<code>x != y</code>	true
<code>></code>	Maior que	<code>x > y</code>	false
<code><</code>	Menor que	<code>x < y</code>	true
<code>>=</code>	Maior que ou igual a	<code>x >= y</code>	false
<code><=</code>	Menor que ou igual a	<code>x <= y</code>	true

Operadores Lógicos em C

Operador	Descrição	Exemplo	Resultado (se a = true e b = false)
&&	E lógico (AND)	a && b	false
 	OU lógico (OR)	a b	true
!	NÃO lógico (NOT)	!a	false
		!b	true

Comandos



```
$(window).scrollTop() > header1_initialDistance  
if (parseInt(header1.css('padding-top'), 10) > $(window).scrollTop() - header1_initialDistance) {  
    header1.css('padding-top', '' + header1_initialDistance);  
}  
else {  
    header1.css('padding-top', '' + header1_initialDistance);  
}  
  
$(window).scrollTop() > header2_initialDistance  
if (parseInt(header2.css('padding-top'), 10) > $(window).scrollTop() - header2_initialDistance) {  
    header2.css('padding-top', '' + header2_initialDistance);  
}  
else {  
    header2.css('padding-top', '' + header2_initialDistance);  
}
```

Estruturas de Controle

- if
- operador ternário
- switch

Sintaxe básica

```
if (expressão) {  
    // Comandos a serem executados se a expressão for verdadeira (diferente  
}
```

Sintaxe básica

```
if (expressão) {  
    // Comandos a serem executados se a expressão for verdadeira (diferente  
} else {  
    // Comandos a serem executados se a expressão for falsa (igual a zero)  
}
```


Comando if

Na estrutura if, a "expressão" é avaliada. **Se o resultado for diferente de zero** (o que é considerado "**verdadeiro**" em C), os comandos dentro do bloco if são executados. Se a expressão resultar **em zero (considerado "falso")** e houver uma cláusula else presente, os comandos dentro do bloco else serão executados.

Comando if

Usando apenas if:

Neste exemplo, verificamos se um número é positivo. Se for, imprimimos uma mensagem.

```
#include <stdio.h>

int main() {
    int num = 5; // Exemplo de valor

    if (num > 0) {
        printf("O número é positivo.\n");
    }

    return 0;
}
```


Comando if..else

Usando o if e else:

Neste exemplo, verificamos se um número é par ou ímpar.

```
#include <stdio.h>

int main() {
    int num = 6; // Exemplo de valor

    if (num % 2 == 0) {
        printf("O número é par.\n");
    } else {
        printf("O número é ímpar.\n");
    }

    return 0;
}
```

Ifs aninhados

O termo "**if aninhado**" refere-se à prática de **usar uma instrução if dentro de outra instrução if**. Isso é comumente usado em situações onde há uma série de condições que precisam ser verificadas em sequência ou quando uma condição depende do resultado de outra condição.

```
if (condição1) {  
    // Código a ser executado se condição1 for verdadeira  
  
    if (condição2) {  
        // Código a ser executado se condição1 e condição2 forem verdadeiras  
    }  
}
```


Ifs aninhados – Exemplo

Suponhamos que você queira verificar se uma pessoa é elegível para votar e, em seguida, se ela é elegível para concorrer a um cargo político. A elegibilidade para votar pode ser aos 18 anos, enquanto a elegibilidade para concorrer a um cargo pode ser aos 25 anos.

Neste próximo exemplo, se a idade for, digamos, **20**, a saída será apenas "**Você é elegível para votar.**". Se a idade for **26**, ambas as mensagens serão exibidas.

Ifs aninhados – Exemplo

```
#include <stdio.h>

int main() {
    int idade = 26; // exemplo de idade

    if (idade >= 18) {
        printf("Você é elegível para votar.\n");

        if (idade >= 25) {
            printf("Você também é elegível para concorrer a um cargo político.\n");
        }
    } else {
        printf("Você não é elegível para votar.\n");
    }

    return 0;
}
```


Considerações ao se usar ifs aninhados



1. Legibilidade

Aninhar **muitas instruções if** pode tornar o código difícil de ler e seguir. É uma boa prática limitar a profundidade do aninhamento sempre que possível.



2. Alternativas

Em muitos casos, o uso de operadores lógicos (como **&&**, **||**) ou a instrução **switch** pode oferecer uma alternativa mais clara ao uso excessivo de if aninhados.



3. Indentação

Manter **uma boa indentação** é crucial ao usar estruturas aninhadas, pois ajuda a visualizar a hierarquia das condições e a lógica geral do código.



Operador Ternário ? :

O operador ternário em C, muitas vezes chamado de "operador condicional" ou simplesmente "ternário", é uma forma concisa de realizar uma verificação if-else. Ele é chamado de "ternário" porque envolve três operandos.

Sintaxe básica

```
condição ? valor_se_verdadeiro : valor_se_falso
```

- **condição:** É a expressão que será avaliada.
- **valor_se_verdadeiro:** É o valor que a expressão retorna se a condição for verdadeira.
- **valor_se_falso:** É o valor que a expressão retorna se a condição for falsa.

Comando switch

O comando **switch** é uma estrutura de controle de decisão em C que permite a um programa testar a igualdade de uma variável ou expressão com múltiplos valores. Em vez de escrever vários **if-else if**, você pode usar o **switch** para tornar o código mais organizado quando estiver lidando com várias opções fixas.

Comando switch

Sintaxe:

```
switch (expressão) {  
    case valor1:  
        // comandos para valor1  
        break;  
  
    case valor2:  
        // comandos para valor2  
        break;  
  
    ...  
  
    default:  
        // comandos se nenhum dos valores anteriores for correspondido  
}
```


Comando switch

Sobre a Sintaxe

- **expressão:** É a variável ou expressão cujo valor você quer verificar.
- **case:** Representa um valor específico que a expressão pode ter.
- **break:** É usado para sair do switch depois de um caso ter sido correspondido e executado.
- **default:** É opcional e pode ser usado para um bloco de comandos que é executado quando nenhum dos casos é correspondido.

Exemplo

Suponha que você tenha um programa que deve exibir o nome do dia da semana com base em um número (1 para domingo, 2 para segunda-feira, etc.):



```
#include <stdio.h>

int main() {
    int dia = 4; // Exemplo

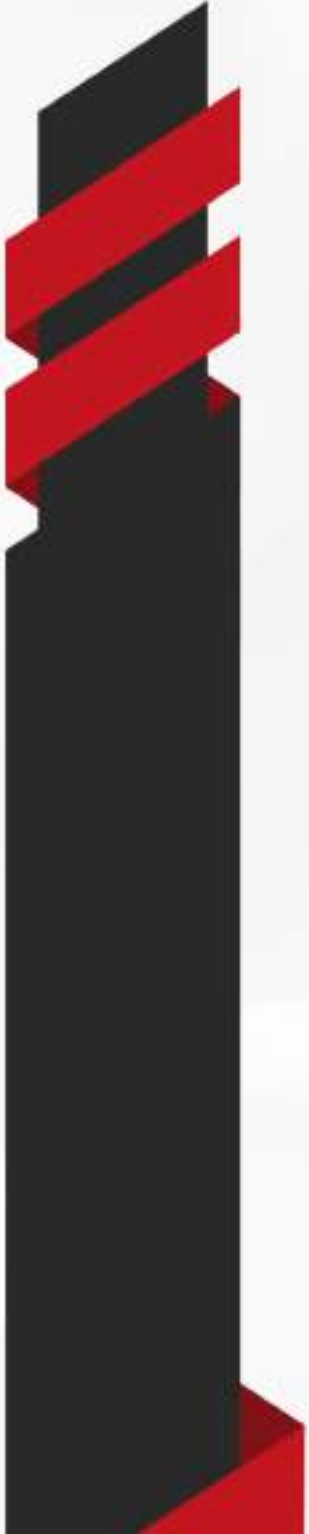
    switch (dia) {
        case 1:
            printf("Domingo");
            break;
        case 2:
            printf("Segunda-feira");
            break;
        case 3:
            printf("Terça-feira");
            break;
```


Uso do break

Importante lembrar de incluir o comando **break** após cada bloco case, caso contrário, o programa continuará a executar os blocos subsequentes até encontrar um break ou até o final do switch.

O **switch em C só pode testar a igualdade**. Não pode ser usado para testes baseados em intervalos ou outras condições. Se você precisar de verificações mais complexas, terá que usar **if-else if-else**.

Comando switch aninhado



O comando **switch aninhado** em C **envolve colocar uma instrução switch dentro de outro bloco case de uma instrução switch externa**. É uma maneira de adicionar outra camada de decisão baseada em diferentes critérios. Embora tecnicamente possível, o uso de switch aninhado pode **complicar a legibilidade do código**, por isso deve ser usado com cautela.

Comando switch aninhado

Exemplo parcial



```
#include <stdio.h>

int main() {
    int escolhaPrincipal = 1; // 1 para bebida fria, 2 para bebida quente
    int escolhaEspecificas = 2; // Exemplo: 1 para água, 2 para refrigerante

    switch (escolhaPrincipal) {
        case 1:
            printf("Você escolheu uma bebida fria.\n");

            switch (escolhaEspecificas) {
                case 1:
                    printf("Você escolheu água.\n");
                    break;
                case 2:
                    printf("Você escolheu refrigerante.\n");
                    break;
                default:
                    printf("Opção inválida para bebida fria.\n");
            }
            break;

        case 2:
            // ...
    }
}
```


Considerações do aninhamento do switch

Legibilidade: Como mencionado anteriormente, o **uso excessivo de switch aninhado pode tornar o código mais difícil de ler e seguir.** Em muitos casos, pode ser útil considerar outras abordagens, como usar funções separadas para diferentes níveis de decisão.

Indentação: A **indentação correta é crucial** ao usar **switch** aninhado para visualizar claramente a estrutura e a hierarquia das decisões.

Complexidade: O uso de **switch** aninhado **pode aumentar a complexidade do código**, principalmente se muitos casos precisarem ser considerados em cada nível. Pode ser útil considerar a refatoração ou outras estratégias de design para manter o código gerenciável.



Exercícios - Aula 03

Exercício 1: Descontos baseados na idade

Enunciado: Escreva um programa que leia a idade de um cliente e determine o desconto em uma compra. Se a idade for menor que 18 anos, ele recebe 5% de desconto. Se tiver entre 18 e 60 anos, recebe 10% de desconto. Se for maior que 60 anos, recebe 15% de desconto. Use a estrutura if-else.

Exercício 2: Determinando o dia da semana

Enunciado: Crie um programa que solicite ao usuário que insira um número de 1 a 7 e exiba o dia da semana correspondente (1 para Domingo, 2 para Segunda-feira, etc.). Se o usuário inserir um número fora desse intervalo, exiba uma mensagem de erro. Use a estrutura switch.



Exercícios - Aula 03

Exercício 3: Avaliando desempenho acadêmico

Enunciado: Desenvolva um programa que leia a nota final de um aluno (de 0 a 100) e determine sua classificação:

- Abaixo de 40: Reprovado
- 40 a 59: Suficiente
- 60 a 79: Bom
- 80 a 89: Muito Bom
- 90 a 100: Excelente Use a estrutura if-else para determinar a classificação.

Exercício 4: Determinando o período do dia

Enunciado: Escreva um programa que leia a hora atual (um número de 0 a 23) e determine se é manhã (5 a 11), tarde (12 a 17), noite (18 a 22) ou madrugada (23 a 4). Use o operador ternário ? :.



Exercícios - Aula 03

Exercício 5: Tipo de veículo baseado em rodas

Enunciado: Crie um programa que solicite ao usuário o número de rodas de um veículo e, usando a estrutura switch, determine se é uma bicicleta (2 rodas), carro (4 rodas), ou "Outro tipo" para qualquer outro número de rodas.

Referências

DAMAS, L. M. D. Linguagem C. LTC, 2007.

HERBERT, S. C completo e total. 3a. ed. Pearson, 1997.

