

# SI100

## Algoritmos e Programação de Computadores I

1º Semestre - 2018

### Tópico 3: Estruturas de Controle – Desvio Condisional

Unidade 7

Prof. Guilherme Palermo Coelho  
[guilherme@ft.unicamp.br](mailto:guilherme@ft.unicamp.br)

# Roteiro

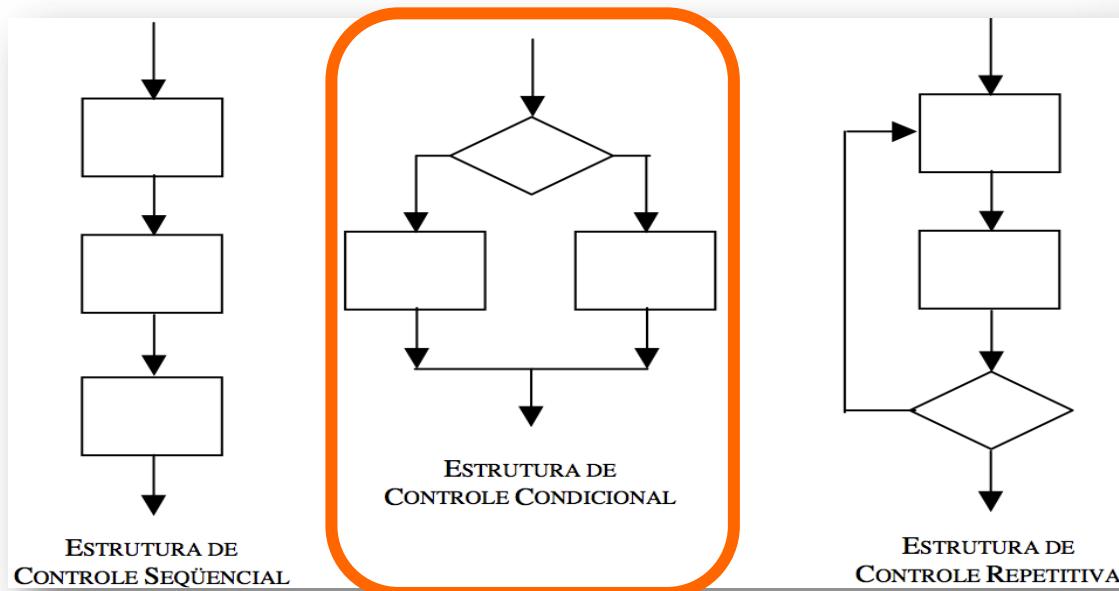
- Introdução;
- O comando *if*;
- O comando *if-else*;
- Operadores lógicos;
- O comando *switch*;
- O operador condicional ternário `?:`;
- Referências.

# INTRODUÇÃO

# Programação Estruturada

- Como vimos nas aulas anteriores, C é uma linguagem de **programação estruturada**:
  - Um programa em C é composto por **blocos elementares** que são interligados através de **três** mecanismos básicos:

- Sequência;
- Seleção;
- Iteração.



# Comandos de Decisão

- Exemplo:
  - 1) Acionar o interruptor;
  - 2) Se a lâmpada **não acender**, então
    - 3) Buscar uma escada;
    - 4) Posicionar a escada embaixo da lâmpada;
    - 5) Pegar uma lâmpada nova;
    - 6) Subir na escada;
    - 7) Retirar a lâmpada velha;
    - 8) Colocar a lâmpada nova;



Como verificar a condição e tomar a decisão?

# O COMANDO *IF*

# O Comando *if*

- Instrui o computador a tomar uma *decisão simples*;
- Formato geral:

```
if (expressão de teste)
    instrução ou bloco de instruções
```

```
/*Teste do comando IF*/
#include <stdio.h>

int main() {
    char ch = getchar();
    if (ch == 's')
        printf("Você pressionou a tecla s.\n");
    return 0;
}
```

# O Comando *if*

```
/*Teste do comando IF*/
#include <stdio.h>

int main() {
    char ch = getchar();
    if (ch == 's')
        printf("Você pressionou a tecla s.\n");

    return 0;
}
```

- A instrução na linha seguinte ao comando **if** só é executada se a **condição** do **if** for **verdadeira**;
  - Ou seja, se `(ch == 's')`
- Note o uso de **indentação** para facilitar a leitura do código.

# O Comando *if*

- No exemplo anterior, um **único** comando deveria ser executado caso a condição do *if* fosse verdadeira;
- Em programas reais, muitas vezes precisamos executar **mais de uma** instrução após um comando condicional.
- Isto pode ser feito através da definição de um **bloco** de instruções;
  - Bloco de instruções: conjunto de instruções dentro de “{” “}”

```
if (expressão de teste) {  
    instrução 1;  
    instrução 2;  
    instrução 3;  
    ...  
}
```

# O Comando *if*

```
/*Teste do comando IF com blocos*/
#include <stdio.h>

int main() {
    char ch = getchar();
    if (ch == 's') {
        printf("Você pressionou a tecla s.\n");
        printf("A letra seguinte a %c eh %c.\n", ch, ch+1);
    }
    return 0;
}
```

- **Atenção:** esquecer de colocar a instruções de um bloco entre “{” “}” é um *erro lógico*;
  - O programa irá compilar mas não gerará o resultado esperado.

# Comandos *if* aninhados

- Um comando *if* pode estar dentro do corpo de *outro* comando *if*;
  - Nestes casos, dizemos que o *if interno* está *aninhado*.

```
/*IFs aninhados*/
#include <stdio.h>

int main() {
    char ch;
    printf("Digite uma letra de \'a\' a \'z\':");
    ch = getchar();
    if (ch >= 'a')
        if (ch <= 'z')
            printf("\nVocê digitou certo!\n");
    return 0;
}
```

# O COMANDO *IF-ELSE*

# O Comando *if-else*

- Nos exemplos anteriores: um único comando (ou bloco de comandos) será executado se a condição do *if* for verdadeira;
- E se quisermos executar alguma outra instrução quando a *expressão de teste* for *falsa*?
  - Para isto existe o comando *else*, que deve ser associado a um *if*.

```
if (expressão de teste) {  
    instrução 1;  
    ...  
}  
else {  
    instrução k;  
    ...  
}
```

Executado se (expressão de teste)  
for *verdadeira*

Executado se (expressão de teste)  
for *falsa*

# O Comando *if-else*

```
/*Teste do comando IF-ELSE*/
#include <stdio.h>

int main() {
    char ch = getchar();
    if (ch == 's') {
        printf("Você pressionou a tecla s.\n");
    }
    else {
        printf("Você não pressionou s.\n");
    }
    return 0;
}
```

# O Comando *if-else* - Exemplo

- Suponha um programa que deva:
  - Ler dois valores inteiros ‘a’ e ‘b’;
  - Escrever na tela qual é o **maior** valor lido (caso sejam diferentes) ou dizer que os valores são **iguais** (caso eles sejam).
- Como será visto nos próximos slides, é possível implementar este programa de formas diferentes, usando os comandos de decisão vistos até agora.

# O Comando *if-else* - Exemplo

```
/* Implementação SEM o uso de ELSE*/
#include <stdio.h>

int main() {
    int a,b;
    printf("Digite dois números inteiros:");
    scanf("%d %d", &a, &b);
    if (a > b) {
        printf("O maior valor é %d.\n", a);
    }
    if (b > a) {
        printf("O maior valor é %d.\n", b); }
    if (a == b) {
        printf("Os valores são iguais.\n"); }
    return 0;
}
```

# O Comando *if-else* - Exemplo

- Na implementação anterior (sem o uso de *else*):
  - **TODAS** as comparações serão feitas;
    - $a > b$ ;
    - $b > a$ ;
    - $a == b$ ;
  - Mesmo que uma das anteriores seja verdadeira.
- O código anterior **não** é eficiente!
  - Faz comparações desnecessárias.

# O Comando *if-else* - Exemplo

```
/* Implementação COM o uso de ELSE*/
#include <stdio.h>

int main() {
    int a,b;
    printf("Digite dois números inteiros:");
    scanf("%d %d", &a, &b);
    if (a > b)
        printf("O maior valor é %d.\n", a);
    else
        if (b > a)
            printf("O maior valor é %d.\n", b);
        else
            printf("Valores são iguais.\n");
    return 0;
}
```

# O Comando *if-else* - Exemplo

- Na implementação anterior (com o uso de *else*):
  - As comparações seguintes somente são feitas se as anteriores forem *falsas*;
  - O código **evita comparações desnecessárias** → é *mais eficiente*.

# OPERADORES LÓGICOS

# Operadores Lógicos

- A linguagem C oferece três operadores chamados **lógicos**:
  - **Operador lógico E (&&):**
    - **(exp1 && exp2):** é **verdadeira** se tanto **exp1** quanto **exp2** forem verdadeiras.
  - **Operador lógico OU (||):**
    - **(exp1 || exp2):** é **verdadeira** se **exp1** ou **exp2** forem verdadeiras e se **exp1** e **exp2** forem verdadeiras.
  - **Operador lógico de negação (NOT):**
    - É um operador **únario**;
    - **(!exp1)** é verdadeira se **exp1** for falsa e vice-versa.

# Operadores Lógicos - Exemplo

- Dadas as 3 notas de um aluno, caso **todas sejam maiores que 5.0** o programa deve exibir a mensagem “**Aprovado**” em conjunto com a média. Caso contrário, deve exibir a mensagem “**Reprovado**”:

```
#include <stdio.h>

int main() {
    float a, b, c;
    printf("Digite as notas:");
    scanf("%f,%f,%f", &a, &b, &c);

    if ((a > 5.0) && (b > 5.0) && (c > 5.0))
        printf("Aprovado - Media: %.2f.\n", (a+b+c)/3);
    else
        printf("Reprovado.\n");

    return 0;
}
```

# Precedência de Operadores

Maior Precedência

Operadores	Tipo dos operadores
! - + <sup>+</sup> - <sup>-</sup>	Unários: não lógico, menos aritmético e incremento/decremento
* / %	Aritméticos
+ -	Aritméticos
< > <= >=	Relacionais
== !=	Relacionais
&&	Lógico E
	Lógico OU
= += -= *= /= %=	Aritméticos de atribuição

Menor Precedência

# O COMANDO *SWITCH*

# Comando *switch*

- **If-else** facilita a escrita de programas que devem escolher uma entre duas alternativas;
- No entanto, em muitas situações precisamos escolher entre várias alternativas:
  - **Opção 1:** usar múltiplos **if-else** aninhados
    - Resolve o problema, mas deixa o código confuso e pouco elegante.
  - **Opção 2:** usar o comando **switch**
    - Similar a **if-else** aninhado, mas com maior flexibilidade e com formato mais limpo e claro.

# Comando *switch*

- Sintaxe:

```
switch (expressão constante) {  
    case constante1:  
        instruções; //opcional  
    case constante2:  
        instruções; //opcional  
    ...  
    default:           //opcional  
        instruções; //opcional  
}
```

- O comando ***switch*** avalia a expressão entre parênteses e compara o resultado com os rótulos dados após os termos “**case**” (constante1, constante2, ...);
  - A expressão entre parênteses deve resultar em **valor inteiro** ou **caractere**.

# Comando *switch*

- Sintaxe:

```
switch (expressão constante) {  
    case constante1:  
        instruções; //opcional  
    case constante2:  
        instruções; //opcional  
    ...  
    default:           //opcional  
        instruções; //opcional  
}
```

- Cada **case** deve ser rotulado por uma constante do tipo **inteiro** ou **caractere** e terminado por “:” (dois pontos);
  - **NÃO** é possível usar variáveis ou expressões para os rótulos.

# Comando *switch*

- Sintaxe:

```
switch (expressão constante) {  
    case constante1:  
        instruções; //opcional  
    case constante2:  
        instruções; //opcional  
    ...  
    default:           //opcional  
        instruções; //opcional  
}
```

- Caso múltiplas instruções sigam um **case**, não é necessário envolvê-las em chaves.
  - O corpo do switch **deve** estar entre chaves.

# Comando *switch*

- Sintaxe:

```
switch (expressão constante) {  
    case constante1:  
        instruções; //opcional  
    case constante2:  
        instruções; //opcional  
    ...  
    default:           //opcional  
        instruções; //opcional  
}
```

- Se o rótulo de um **case** for igual à expressão, a execução começa nele.
- Se não houver nenhum rótulo igual e existir um **default**, a execução começa nele.

# Comando *switch* - Exemplo

```
#include <stdio.h>

int main() {
    int diaSemana;
    scanf("%d", &diaSemana);

    switch (diaSemana) {
        case 1:
            printf("Domingo\n");
            break;
        case 2:
            printf("Segunda-feira\n");
            break;
        ...
        default:
            printf("Dia inválido.\n");
    }
    return 0;
}
```

# Comando *switch*

- Quando ocorre uma **correspondência** entre um rótulo de um **case** e o resultado da expressão, **todas** as instruções a seguir deste rótulo são executadas;
- No entanto, nós queremos que apenas as instruções associadas a um **case** sejam executadas;
  - Para que isto ocorra, é necessário adicionar um comando **break** ao final do bloco de instruções de cada **case**.

```
switch (expressão) {  
    ...  
    case constante2:  
        instrução1;  
        instrução2;  
        break;  
    ...  
}
```

**Break** causa uma saída  
imediata do *switch*

# O OPERADOR CONDICIONAL TERNÁRIO ?:

( 32 )

# Operador condicional ternário ?:

- C possui um outro operador condicional que permite representar, ***de forma compacta***, uma instrução ***if-else***;
  - Este operador é chamado de ***operador condicional***.

- Formato:

```
condição ? expressao_1 : expressao_2
```

- **condição**: expressão lógica que deve ser avaliada em **verdadeiro** ou **falso**.
- **expressao\_1**: valor que será resultante da condicional caso “**condição == verdadeiro**”;
- **expressao\_2**: valor que será resultante da condicional caso “**condição == falso**”;

# Operador condicional ternário ?:

- Exemplos:
  - `max = (num1 > num2) ? num1 : num2;`
  - `valorAbsoluto = (num < 0) ? -num : num;`
- É importante lembrar que o comando ***if-else*** pode substituir todas as expressões escritas com o operador condicional;
- No entanto, o uso do operador condicional deixa tais expressões mais “compactas”.

# EXERCÍCIOS

# Exercícios

1. Escreva um programa que leia um número inteiro e imprima na tela se ele é **par** ou **ímpar**.
2. Escreva um programa que leia dez números inteiros e imprima na tela quantos destes números são pares e quantos são ímpares.
3. Escreva um programa que leia uma data no formato **dd/mm/aaaa** e escreva esta data com o mês por extenso (utilize **apenas** comandos ***if-else*** para comparações):
  - Ex.:
    - Entrada: 22/07/1980
    - Saída: 22 de julho de 1980.
4. Reescreva o programa do exercício 3 usando o comando ***switch***.

# Exercícios

5. Escreva um programa que leia um caractere do teclado, fornecido pelo usuário, e imprima na tela:
  - O caractere em letra maiúscula, caso ele esteja em letra minúscula;
  - O próprio caractere, caso contrário.
6. Em um determinado semestre de um curso foram aplicadas três provas com pesos iguais. Escreva um programa que leia estas três notas e exiba a **média** do aluno e seu ***status*** no curso:
  - Se as três notas forem maiores ou iguais a 6.0, o aluno está **aprovado**;
  - Se uma das notas for inferior a 2.0, o aluno está **reprovado**.
  - Caso contrário, o aluno deveria ter feito exame e sua média será dada pela soma da média das provas com a nota do exame dividida por dois. O programa deve **ler a nota do exame** neste caso. A aprovação ocorrerá se a média for **maior ou igual a 5.0**.

# Exercícios

7. Escreva um programa que simule o funcionamento de uma calculadora simples com quatro operações (+, -, \* e /). Este programa deverá ler os **operandos** e o **operador**, como ilustrado abaixo, e exibir o resultado da operação (repetindo os operandos e o operador fornecidos pelo usuário). Use **apenas** comandos ***if-else***.
  - Ex. de entrada e saída:
    - Entrada:  $4 + 2$
    - Saída:  $4 + 2 = 6$
8. Reescreva o programa do exercício 7 usando apenas o operador ***switch***.

# Exercícios

- Observação:
  - Os seguintes exercícios devem ser entregues via SuSy:
    - Exercício 1;
    - Exercício 4;
    - Exercício 7.
  - Veja os enunciados atualizados no site do sistema:
    - <https://susy.ic.unicamp.br:9999/si100a> (Turma A);
    - <https://susy.ic.unicamp.br:9999/si100b> (Turma B);

# REFERÊNCIAS

# Referências

- MIZRAHI, V. V., *Treinamento em Linguagem C – Curso Completo*, 2a Edição, Pearson Makron Books, 2005.
- MIYAZAWA, F. K., DE SOUZA, C. C. & KOWALTOWSKI, T.. *Notas de Aula da disciplina Algoritmos e Programação de Computadores*. Instituto de Computação, Unicamp.