Material de Apoio - Semana 3

Estruturas Condicionais

Introdução a Técnicas de Programação (2025.2)

Objetivos de Aprendizagem

Ao final desta semana, você será capaz de:

- Compreender o funcionamento das estruturas condicionais em C
- Implementar decisões simples usando `if`
- Utilizar estruturas de decisão com alternativas usando `if/else`
- Construir estruturas condicionais aninhadas para casos complexos
- Aplicar estruturas condicionais na resolução de problemas reais

---

1. Introdução às Estruturas Condicionais

1.1 Conceito e Importância

As **estruturas condicionais** permitem que um programa execute diferentes blocos de código dependendo de condições específicas. Isso introduz a capacidade de tomada de decisão nos programas, tornando-os mais inteligentes e adaptáveis.

Em linguagem C, as principais estruturas condicionais são:

- `if` (se) - para execução condicional simples
- `if/else` (se/senão) - para alternativas
- `if/else if/else` - para múltiplas condições

1.2 Fluxo de Execução Condicional

Até agora, nossos programas tinham um fluxo linear de execução. Com as estruturas condicionais, podemos criar caminhos alternativos:

```c
#include <stdio.h>

int main() {
    int numero;

    printf("Digite um número: ");
    scanf("%d", &numero);

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

```
    printf("Fim do programa.\n");
    return 0;
}
```

Neste exemplo, a mensagem "O número é positivo" só será exibida se a condição `numero > 0` for verdadeira.

---

2. Estrutura Condicional `if`

2.1 Sintaxe Básica

A estrutura `if` tem a seguinte sintaxe:

```
if (condição) {
    // bloco de código a ser executado
    // se a condição for verdadeira
}
```

Componentes:

- **Condição**: Uma expressão que resulta em verdadeiro (diferente de 0) ou falso (0)
- **Bloco de código**: Conjunto de instruções entre chaves `{}`

2.2 Exemplos Práticos

```
// Exemplo 1: Verificar se um número é par
int numero;
printf("Digite um número: ");
scanf("%d", &numero);

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

// Exemplo 2: Verificar maioridade
int idade;
printf("Digite sua idade: ");
scanf("%d", &idade);

if (idade >= 18) {
    printf("Você é maior de idade.\n");
    printf("Pode votar e dirigir.\n");
}
```

2.3 Operadores Relacionais em Condições

Os operadores relacionais são essenciais para construir condições:

Operador	Descrição	Exemplo
`==`	Igual a	`idade == 18`
`!=`	Diferente de	`numero != 0`
`>`	Maior que	`salario > 1000`
`<`	Menor que	`temperatura < 0`
`>=`	Maior ou igual	`nota >= 7`
`<=`	Menor ou igual	`altura <= 1.90`

Cuidado importante: Não confunda `=` (atribuição) com `==` (comparação):

```
int x = 5;
if (x = 3) {    // ✗ ERRO: Atribui 3 a x e sempre executa o bloco
    printf("Sempre executado!\n");
}

if (x == 3) {   // ✓ CORRETO: Compara x com 3
    printf("Executado apenas se x for 3\n");
}
```

---

3. Estrutura Condicional `if/else`

3.1 Sintaxe e Funcionamento

A estrutura `if/else` permite definir um bloco alternativo para quando a condição é falsa:

```
if (condição) {
    // bloco executado se condição for verdadeira
} else {
    // bloco executado se condição for falsa
}
```

3.2 Exemplos Práticos

```
// Exemplo 1: Verificar par ou ímpar
int numero;
printf("Digite um número: ");
scanf("%d", &numero);

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

```
}

// Exemplo 2: Aprovação simples
float nota;
printf("Digite sua nota: ");
scanf("%f", &nota);

if (nota >= 7.0) {
    printf("Aprovado!\n");
} else {
    printf("Reprovado.\n");
}
```

3.3 Operadores Lógicos

Para condições mais complexas, usamos operadores lógicos:

Operador	Descrição	Exemplo
`&&`	E lógico	`(idade >= 18) && (idade <= 65)`
`||`	OU lógico	`(nota1 >= 7) || (nota2 >= 7)`
`!`	NÃO lógico	`!(numero == 0)`

Exemplos:

```
// Verificar se um número está entre 0 e 100
int numero;
printf("Digite um número: ");
scanf("%d", &numero);

if (numero >= 0 && numero <= 100) {
    printf("O número está entre 0 e 100.\n");
} else {
    printf("O número está fora do intervalo 0-100.\n");
}

// Verificar se um ano é bissexto
int ano;
printf("Digite um ano: ");
scanf("%d", &ano);

if ((ano % 4 == 0 && ano % 100 != 0) || (ano % 400 == 0)) {
    printf("O ano é bissexto.\n");
} else {
    printf("O ano não é bissexto.\n");
}
```

---

4. Estruturas Condicionais Aninhadas

4.1 Conceito e Aplicação

Podemos colocar estruturas condicionais dentro de outras, criando decisões mais complexas:

```
if (condição1) {
    // Executado se condição1 for verdadeira
    if (condição2) {
        // Executado se ambas condições forem verdadeiras
    }
} else {
    // Executado se condição1 for falsa
}
```

4.2 Exemplo: Classificação de Notas

```
float nota;
printf("Digite sua nota: ");
scanf("%f", &nota);

if (nota >= 9.0) {
    printf("Conceito A\n");
} else {
    if (nota >= 7.0) {
        printf("Conceito B\n");
    } else {
        if (nota >= 5.0) {
            printf("Conceito C\n");
        } else {
            printf("Conceito D - Reprovado\n");
        }
    }
}
```

4.3 Estrutura `else if` para Múltiplas Condições

Para evitar aninhamento excessivo, podemos usar `else if`:

```
float nota;
printf("Digite sua nota: ");
scanf("%f", &nota);

if (nota >= 9.0) {
    printf("Conceito A\n");
} else if (nota >= 7.0) {
    printf("Conceito B\n");
} else if (nota >= 5.0) {
    printf("Conceito C\n");
} else {
```

```
    printf("Conceito D - Reprovado\n");
}
```

Esta estrutura é equivalente à anterior, mas mais legível.

---

5. Aplicação em Problemas Reais

5.1 Resolução de Equação do 2º Grau

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

int main() {
    float a, b, c, delta, x1, x2;

    printf("Digite os coeficientes a, b e c: ");
    scanf("%f %f %f", &a, &b, &c);

    if (a == 0) {
        printf("Não é uma equação do segundo grau.\n");
    } else {
        delta = b * b - 4 * a * c;

        if (delta > 0) {
            x1 = (-b + sqrt(delta)) / (2 * a);
            x2 = (-b - sqrt(delta)) / (2 * a);
            printf("Duas raízes reais: %.2f e %.2f\n", x1, x2);
        } else if (delta == 0) {
            x1 = -b / (2 * a);
            printf("Uma raiz real: %.2f\n", x1);
        } else {
            printf("Não existem raízes reais.\n");
        }
    }

    return 0;
}
```

5.2 Classificação de Triângulos

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

int main() {
    float a, b, c;

    printf("Digite os três lados do triângulo: ");
    scanf("%f %f %f", &a, &b, &c);
```

```
    // Verificar se forma um triângulo
    if (a + b > c && a + c > b && b + c > a) {
        printf("Forma um triângulo ");

        // Classificar quanto aos lados
        if (a == b && b == c) {
            printf("equilátero");
        } else if (a == b || a == c || b == c) {
            printf("isósceles");
        } else {
            printf("escaleno");
        }

        // Classificar quanto aos ângulos
        // Encontrar o maior lado
        float maior = a;
        if (b > maior) maior = b;
        if (c > maior) maior = c;

        if (maior * maior == a*a + b*b + c*c - maior*maior) {
            printf(" retângulo\n");
        } else if (maior * maior < a*a + b*b + c*c - maior*maior) {
            printf(" acutângulo\n");
        } else {
            printf(" obtusângulo\n");
        }
    } else {
        printf("Os valores não formam um triângulo.\n");
    }

    return 0;
}
```

---

6. Boas Práticas com Condicionais

6.1 Indentação e Legibilidade

Boa indentação:

```
// ✓ CORRETO: Indentação clara
if (condição) {
    // bloco de código
    if (outra_condição) {
        // bloco aninhado
    }
} else {
    // bloco alternativo
}
```

Má indentação:

```
// ✗ EVITAR: Dificulta a leitura
if (condição) {
// bloco de código
if (outra_condição) {
// bloco aninhado
}
} else {
// bloco alternativo
}
```

6.2 Uso de Chaves

Sempre use chaves, mesmo para blocos de uma linha:

```
// ✓ CORRETO: Chaves sempre usadas
if (idade >= 18) {
    printf("Maior de idade\n");
}

// ✗ EVITAR: Pode causar erros ao modificar o código
if (idade >= 18)
    printf("Maior de idade\n");
```

6.3 Condições Complexas

Divida condições complexas para melhor legibilidade:

```
// Condição complexa (difícil de ler)
if ((idade >= 18 && idade <= 65) && (salario > 1000 || tem_contrato)) {
    // código
}

// Condição dividida (mais legível)
int idade_valida = (idade >= 18 && idade <= 65);
int condicao_financeira = (salario > 1000 || tem_contrato);

if (idade_valida && condicao_financeira) {
    // código
}
```

6.4 Validação de Entrada

Sempre valide entradas do usuário:

```c
float nota;
printf("Digite uma nota (0-10): ");
scanf("%f", &nota);

if (nota < 0 || nota > 10) {
    printf("Erro: Nota deve estar entre 0 e 10.\n");
    return 1; // Encerra o programa com código de erro
}

// Continua com o processamento normal...
```

---

7. Exercícios Resolvidos

7.1 Verificador de Número Positivo/Negativo/Zero

```c
#include <stdio.h>

int main() {
    float numero;

    printf("Digite um número: ");
    scanf("%f", &numero);

    if (numero > 0) {
        printf("O número é positivo.\n");
    } else if (numero < 0) {
        printf("O número é negativo.\n");
    } else {
        printf("O número é zero.\n");
    }

    return 0;
}
```

7.2 Calculadora Simples

```c
#include <stdio.h>

int main() {
    float num1, num2, resultado;
    char operador;

    printf("Digite a expressão (ex: 2 + 3): ");
    scanf("%f %c %f", &num1, &operador, &num2);

    if (operador == '+') {
        resultado = num1 + num2;
```

```
        printf("Resultado: %.2f\n", resultado);
    } else if (operador == '-') {
        resultado = num1 - num2;
        printf("Resultado: %.2f\n", resultado);
    } else if (operador == '*') {
        resultado = num1 * num2;
        printf("Resultado: %.2f\n", resultado);
    } else if (operador == '/') {
        if (num2 != 0) {
            resultado = num1 / num2;
            printf("Resultado: %.2f\n", resultado);
        } else {
            printf("Erro: Divisão por zero!\n");
        }
    } else {
        printf("Operador inválido!\n");
    }

    return 0;
}
```

---

8. Dicas para Resolver a Lista de Exercícios

1. **Leia atentamente** cada problema antes de começar a codificar
2. **Identifique as condições** necessárias para cada decisão
3. **Desenhe um fluxograma** mental ou no papel antes de codificar
4. **Teste com valores extremos** (valores mínimos, máximos e casos especiais)
5. **Use nomes descritivos** para variáveis relacionadas a condições
6. **Comente seu código** para explicar condições complexas

Exemplo de nomenclatura descritiva:

```
// ✓ BOM: Nomes descritivos
int idade_usuario;
float salario_bruto;
int eh_maior_de_idade;

// ✗ RUIM: Nomes genéricos
int i;
float x;
int flag;
```

---

9. Próximos Passos

Na **Semana 4**, estudaremos:

- Estruturas de repetição (for, while, do-while)

- Controle de fluxo com break e continue
- Laços aninhados
- Aplicações práticas de repetição