Exercícios de Repetição e Condicional

1. Menu Interativo de Opções

Um menu interativo permite que o usuário escolha uma opção e execute ações diferentes com base nessa escolha. Veja um exemplo simples de um menu interativo:

```c
#include <stdio.h>

int main() {
    int opcao;

    do {
        printf("Menu de Opções:\n");
        printf("1. Somar\n");
        printf("2. Subtrair\n");
        printf("3. Sair\n");
        printf("Escolha uma opção: ");
        scanf("%d", &opcao);

        switch (opcao) {
            case 1:
                printf("Você escolheu Somar!\n");
                break;
            case 2:
                printf("Você escolheu Subtrair!\n");
                break;
            case 3:
                printf("Saindo...\n");
                break;
            default:
                printf("Opção inválida! Tente novamente.\n");
                break;
        }
    } while (opcao != 3);

    return 0;
}
```

No exemplo acima, é lido um valor inteiro (opção) e caso seja:

- 1: Poderia ser realizado uma rotina de soma, mas no exemplo é apenas printado a opção.
- 2: Poderia ser realizado uma rotina de subtração.
- 3: O programa é encerrado.
- default: Caso seja uma opção diferente de 1, 2 ou 3 ele retorna opção inválida

O `break` no fim de cada case é para indicar a saída do bloco switch retornado ao início do loop onde é feita a leitura da opção novamente. Caso você esqueça dele, o programa continuará a executar os casos

subsequentes, mesmo que as condições não sejam verdadeira. Por exemplo, se `opcao` fosse 1, e você não tivesse um `break` após o `case 3` o programa seria encerrado.

Esse processo de leituras e prints da opção vai continuar até o momento em que o usuário digitar 3, momento esse onde o `do-while` é finaizado.

Exercícios Práticos

1. **Menu de Opções:** Crie um menu (assim como o visto acima) que permita ao usuário escolher entre somar, subtrair, multiplicar e dividir dois números (floats). Utilize de condicionais e estruturas de repetição para resolver esse exercício.
2. **Cálculo do IMC:** Peça ao usuário que insira seu peso e altura e calcule o IMC (Índice de Massa Corporal). Use condicionais para classificar o IMC em categorias (como abaixo do peso, normal, sobrepeso, etc.). Utilize de condicionais para resolver esse exercício.
3. **Sequência de Fibonacci:** Gere a sequência de Fibonacci até um número fornecido pelo usuário. Utilize de estruturas de repetição como o for para resolver esse exercício.
4. **Números Primos:** Crie um programa que imprima todos os números primos até um número fornecido pelo usuário. Utilize de condicionais e estruturas de repetição para resolver esse exercício.
5. **Números Perfeitos:** Peça ao usuário um número máximo e imprima todos os números perfeitos até esse número.

O que é Sequência de Fibonacci?

A **sequência de Fibonacci** é uma sequência numérica na qual cada número é a soma dos dois anteriores. A sequência geralmente começa com 0 e 1. Os primeiros números da sequência de Fibonacci são:

```
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...
```

Fórmula Recursiva

A sequência de Fibonacci pode ser definida recursivamente da seguinte forma:

- F(0) = 0
- F(1) = 1
- F(n) = F(n-1) + F(n-2) para n ≥ 2

Exemplo de Execução

Vamos supor que o usuário deseja calcular os primeiros 5 termos da sequência de Fibonacci. Ao inserir o valor 5, a saída do programa seria:

```
Digite quantos termos da sequencia de Fibonacci voce deseja: 5
Sequencia de Fibonacci: 0, 1, 1, 2, 3,
```

Dica

Complete o código abaixo para resolver o exercício proposto

```c
#include <stdio.h>

int main() {
    int n, primeiro = 0, segundo = 1, proximo;

    printf("Digite quantos termos da sequencia de Fibonacci voce deseja: ");
    scanf("%d", &n);

    printf("Sequencia de Fibonacci: ");

    for (int i = 1; i <= n; i++) {
        if (i == 1) {
            printf("%d, ", primeiro);
            continue;
        }
        if (i == 2) {
            printf("%d, ", segundo);
            continue;
        }
        ...
        ...
        ...

        printf("%d, ", proximo);
    }

    printf("\n");
    return 0;
}
```

O que são números primos?

Um **número primo** é um número natural maior que 1 que não pode ser formado pela multiplicação de dois números naturais menores que ele. Em outras palavras, um número primo tem exatamente dois divisores: 1 e ele mesmo. Os primeiros números primos são:

```
2, 3, 5, 7, 11, 13, 17, 19, 23, ...
```

Exemplo de Execução

Vamos supor que o usuário deseja encontrar os números primos até o número 20. Ao inserir o valor 20, a saída do programa seria:

```
Digite um numero inteiro positivo: 20
Números primos até 20: 2, 3, 5, 7, 11, 13, 17, 19,
```

Dica

Complete o código abaixo para resolver o exercício proposto

```c
#include <stdio.h>

int main() {
    int n, i, j, is_primo;

    printf("Digite um numero inteiro positivo: ");
    scanf("%d", &n);

    printf("Números primos até %d: ", n);

    for (i = 2; i <= n; i++) {
        is_primo = 1; // Assume que o número é primo

        for (j = 2; j * j <= i; j++) { // Verifica se i é divisível por j
            if (i % j == 0) {
                ...
                break; // Sai do loop se encontrar um divisor
            }
        }

        if (is_primo) {
            printf("%d, ", i); // Imprime o número primo
        }
    }

    printf("\n");
    return 0;
}
```

O exercício acima é muito interessante pois envolve variáveis booleanas para decidir se um número pertence a um conjunto ou não.

O que são números perfeitos?

Um **número perfeito** é um número inteiro positivo que é igual à soma de seus divisores próprios positivos (excluindo ele mesmo). Em outras palavras, a soma dos divisores de um número perfeito, excluindo o próprio número, deve resultar exatamente no número.

Exemplo de números perfeitos:

Os primeiros números perfeitos são:

- 6: Os divisores próprios são 1, 2 e 3. A soma é `1 + 2 + 3 = 6`.

- 28: Os divisores próprios são 1, 2, 4, 7 e 14. A soma é `1 + 2 + 4 + 7 + 14 = 28`.
- 496: Os divisores próprios são 1, 2, 4, 8, 16, 31, 62, 124, 248. A soma é `1 + 2 + 4 + 8 + 16 + 31 + 62 + 124 + 248 = 496`.

Exemplo de Execução

Vamos supor que o usuário deseja verificar se o número 28 é perfeito. Ao inserir o valor 28, a saída do programa seria:

```
Digite um numero inteiro positivo: 28
28 eh um número perfeito.
```

Se o usuário inserir 12, que não é um número perfeito, a saída seria:

```
Digite um numero inteiro positivo: 12
12 não eh um número perfeito.
```

Dica

Complete o código abaixo para resolver o exercício proposto

```c
#include <stdio.h>

int main() {
    int n, soma = 0;

    printf("Digite um numero inteiro positivo: ");
    scanf("%d", &n);

    // Calcula a soma dos divisores próprios
    for (int i = 1; i <= n / 2; i++) {
        if (...) {
            soma += i; // Adiciona o divisor à soma
        }
    }

    // Verifica se a soma dos divisores é igual ao número
    if (...) {
        printf("%d é um número perfeito.\n", n);
    } else {
        printf("%d não é um número perfeito.\n", n);
    }

    return 0;
}
```

Conclusão

Ao longo desta semana, experimente implementar os programas mencionados e tente criar suas próprias variações!