EXERCÍCIOS DE FIXAÇÃO
ESTRUTURAS DE REPETIÇÃO

Para esta série de exercícios de fixação, não será fornecido o gabarito. Resolva os exercícios e realize pequenas refatorações. Em caso de dúvidas, consulte um dos professores em sala de aula ou envie um e-mail.

1. Escreva um programa em ANSI C que exiba os números de -50 a 50. Utilize as estruturas de repetição for, while e do-while.

2. Elabore um programa que leia um número inteiro positivo n e calcule a soma dos números de 1 a n usando as estruturas de repetição for, while e do-while. Exiba o resultado da soma.

3. Implemente um programa que leia um número inteiro positivo n e exiba todos os números ímpares de 1 até n usando as estruturas de repetição for, while e do-while. Exiba um número por linha.

4. Desenvolva um programa em ANSI C que leia um número inteiro positivo n e calcule o fatorial de n utilizando as estruturas de repetição while e do-while.

5. Escreva um programa que leia um número inteiro positivo n e exiba a sequência dos n primeiros números da sequência de Fibonacci (https://brasilescola.uol.com.br/matematica/sequencia-fibonacci.htm). Utilize as estruturas de repetição for, while e do-while para calcular a sequência e exiba cada termo em uma linha.

6. Crie um programa em ANSI C que leia números inteiros fornecidos pelo usuário e exiba o maior número digitado. A entrada de números termina quando o usuário digitar zero. Use uma estrutura de repetição do-while.

7. Implemente um programa que leia um número inteiro n e calcule o somatório dos números pares de 1 até n usando uma estrutura de repetição for. Exiba o somatório ao final.

8. Desenvolva um programa que leia um número inteiro positivo n e exiba a tabuada de multiplicação de n (de 1 a 10) usando uma estrutura de repetição while e do-while. Exiba cada linha da tabuada no formato: n x i = res.

9. Escreva um programa em ANSI C que leia um número inteiro positivo n e verifique se n é um número primo. Utilize uma estrutura de repetição for para realizar a verificação e exiba uma mensagem indicando se n é primo ou não. Um número primo é um número natural maior que 1 que só pode ser dividido por 1 e por ele mesmo sem deixar resto. Em outras palavras, um número primo tem exatamente dois divisores: 1 e o próprio número.

10. Elabore um programa que leia um número inteiro positivo n e exiba todos os divisores de n, cada um em uma linha. Utilize uma estrutura de repetição for para encontrar e exibir os divisores.

11. Escreva um programa que leia um número inteiro positivo n e exiba a soma dos dígitos de n. Utilize uma estrutura de repetição while para realizar o cálculo.

12. Implemente um programa que leia um número inteiro positivo n e exiba uma pirâmide de asteriscos com n linhas. Cada linha i deve conter i asteriscos. Utilize uma estrutura de repetição aninhada.

13. Desenvolva um programa em ANSI C que leia números inteiros positivos até que o usuário digite um número negativo. Ao final, exiba quantos números pares foram digitados. Utilize uma estrutura de repetição do-while.

14. Crie um programa que exiba a sequência de números de 1 a 100, mas, para múltiplos de 3, exiba "IFSP" em vez do número, e para múltiplos de 5, exiba "CAR". Para múltiplos de ambos, exiba "IFSP-CAR". Utilize uma estrutura de repetição for.

15. Implemente um programa em ANSI C que calcule o número reverso de um inteiro positivo n fornecido pelo usuário. Utilize uma estrutura de repetição while para inverter os dígitos e exibir o número resultante. A entrada de dados deve ser através de um único scanf().

16. Elabore um programa que leia uma sequência de números inteiros (positivos e negativos) e terminada pelo número zero e exiba a média dos números pares digitados. Utilize uma estrutura de repetição do-while.

17. Desenvolva um programa que leia um número inteiro positivo n e exiba uma tabela com a potência de 2 para cada expoente de 1 até n (ou seja, 2^1, 2^2, ... 2^n). Utilize uma estrutura de repetição for para realizar os cálculos.

18. Escreva um programa que leia um número inteiro positivo n e exiba a sequência de n números de uma série que começa em 1 e dobra o valor a cada passo (1, 2, 4, 8, ...). Utilize uma estrutura de repetição while.

19. Implemente um programa que leia um número inteiro positivo n e verifique se ele é um número de Armstrong (ou seja, se a soma dos seus dígitos elevados ao número de dígitos é igual ao próprio número). Utilize uma estrutura de repetição while. Um número de Armstrong é um número que é igual à soma de seus próprios dígitos, cada um elevado à potência do número de dígitos. Por exemplo: 153 é um número de Armstrong porque (1^3 + 5^3 + 3^3 = 153) e 370 também é um número de Armstrong porque (3^3 + 7^3 + 0^3 = 370). O código poderá receber até 4 digitos, elaborar a função para calcular o quadrado, cubo e elevado a quarta potência.

20. Escreva um programa que leia um número inteiro n e exiba os primeiros n números palíndromos. Um número palíndromo é aquele que é igual ao seu reverso. Utilize uma estrutura de repetição while.

21. Crie um programa que leia um número inteiro positivo n e exiba uma tabela de conversão de metros para pés para valores de 1 até n metros. Utilize uma estrutura de repetição for para calcular e exibir a conversão de cada valor. Sendo: Pés = Metros × 3,28084.

22. Desenvolva um programa que leia n pares de números, onde o primeiro número é um inteiro e o segundo é um número de ponto flutuante. Usando uma estrutura de repetição do-while, multiplique os pares entre si, realizando o casting necessário para que o resultado seja um número de ponto flutuante (float). Exiba cada produto com precisão de três casas decimais.

23. Escreva um programa que converta uma série de temperaturas de Fahrenheit para Celsius, repetindo o cálculo para cada valor inserido pelo usuário até que seja digitado o valor zero. Utilize uma estrutura de repetição for e converta o valor de entrada de int para float antes de realizar o cálculo, aplicando a fórmula de conversão: C = 5/9 x (F - 32)

24. Escreva um programa que peça ao usuário para digitar dois números inteiros positivos a e b, onde a deve ser menor que b. Caso o usuário digite valores fora desse critério, exiba uma mensagem de erro e solicite novamente ambos os números até que eles sejam válidos.

25. Desenvolva um programa que solicite a idade e o peso de uma pessoa. A idade deve ser um número entre 1 e 100, e o peso deve estar entre 30 e 150 kg. Caso uma ou ambas as entradas estejam fora do intervalo, exiba uma mensagem de erro e peça os valores novamente até que ambos sejam válidos.

26. Implemente um programa que leia dois números inteiros x e y. O programa deve garantir que x seja par e que y seja ímpar. Se qualquer um dos números não atender aos critérios, exiba uma mensagem de erro e solicite novamente ambos os números até que sejam válidos.

27. Elabore um programa que peça ao usuário duas notas de um aluno (valores entre 0 e 10) e um número inteiro representando a quantidade de faltas do aluno. Caso alguma das notas esteja fora do intervalo ou o número de faltas seja negativo, exiba uma mensagem de erro e peça novamente todas as entradas até que estejam corretas.

28. Crie um programa que solicite dois números inteiros a e b. O programa deve garantir que ambos os números sejam múltiplos de 5 e que a seja menor que b. Caso alguma das condições não seja atendida, exiba uma mensagem de erro e peça os valores novamente até que ambos estejam corretos.

29. Desenvolva um programa que contenha uma função chamada somaDigitos, que receba um número inteiro positivo e retorne a soma dos seus dígitos. No main, leia uma sequência de números até que o usuário insira o valor 0. Para cada número inserido, chame a função somaDigitos e exiba o resultado. Use uma estrutura de repetição para solicitar continuamente os números do usuário até que ele digite 0.

30. Escreva um programa em ANSI C que solicite ao usuário uma string utilizando a função gets(). Em seguida, utilizando uma estrutura de repetição, percorra cada caractere da string para contar o número total de vogais presentes (a, e, i, o, u), considerando tanto letras maiúsculas quanto minúsculas. Ao final, exiba a quantidade total de vogais encontradas na string fornecida.