📚 Lista Completa de Exercícios em C

Pratique e consolide seu conhecimento em programação C

🟢 NÍVEL FÁCIL (Fundamentos)

Estruturas de Controle

1. Par ou Ímpar

Objetivo: Ler um número e verificar se é par ou ímpar

Conceitos: if-else, operador módulo %

Bibliotecas: stdio.h

2. Calculadora Simples

Objetivo: Criar uma calculadora com as 4 operações básicas usando menu

Conceitos: switch-case, operadores aritméticos

Bibliotecas: stdio.h

3. Tabuada

Objetivo: Exibir a tabuada de um número de 1 a 10

Conceitos: for, multiplicação

Bibliotecas: stdio.h

4. Contagem Regressiva

Objetivo: Fazer contagem regressiva de 10 até 0

Conceitos: while, decremento

Bibliotecas: stdio.h

5. Maior de Três Números

Objetivo: Ler três números e exibir o maior

Conceitos: if-else aninhado, comparações

Bibliotecas: stdio.h

Strings e ctype.h

6. Contador de Vogais

Objetivo: Contar vogais em uma string

Conceitos: for, tolower(), comparação de caracteres

Bibliotecas: stdio.h, string.h, ctype.h

7. Conversor Maiúsculas/Minúsculas

Objetivo: Converter string para maiúsculas ou minúsculas

Conceitos: toupper(), tolower(), loops

Bibliotecas: stdio.h, string.h, ctype.h

8. Classificador de Caracteres

Objetivo: Classificar cada caractere (letra, dígito, espaço, pontuação)

Conceitos: isalpha(), isdigit(), isspace(), ispunct()

Bibliotecas: stdio.h, string.h, ctype.h

Vetores (Arrays)

9. Soma de Elementos

Objetivo: Ler 5 números, armazenar em vetor e calcular a soma

Conceitos: arrays, for, acumulador

Bibliotecas: stdio.h

10. Maior e Menor Elemento

Objetivo: Encontrar o maior e menor valor em um vetor

Conceitos: arrays, comparações, algoritmo de busca

Bibliotecas: stdio.h

🟠 NÍVEL MÉDIO (Intermediário)

Funções

11. Função Fatorial

Objetivo: Criar função que calcula o fatorial de um número

Conceitos: funções, recursão ou loop, passagem de parâmetros

Bibliotecas: stdio.h

12. Função Palíndromo

Objetivo: Criar função que verifica se uma string é palíndromo

Conceitos: funções, string.h, comparação de strings

Bibliotecas: stdio.h, string.h, ctype.h

13. Calculadora com Funções

Objetivo: Refatorar calculadora criando funções separadas para cada operação

Conceitos: funções, modularização, retorno de valores

Bibliotecas: stdio.h

14. Validador de CPF (simplificado)

Objetivo: Criar função que valida formato básico de CPF (XXX.XXX.XXX-XX)

Conceitos: funções, isdigit(), validação de formato

Bibliotecas: stdio.h, string.h, ctype.h

string.h - Manipulação de Strings

15. Contador de Palavras

Objetivo: Contar quantas palavras há em uma frase

Conceitos: strtok(), isspace(), parsing de strings

Bibliotecas: stdio.h, string.h, ctype.h

16. Invertedor de String

Objetivo: Inverter uma string sem usar função pronta

Conceitos: strlen(), manipulação de índices, troca de posições

Bibliotecas: stdio.h, string.h

17. Comparador de Strings (ignorando case)

Objetivo: Comparar duas strings ignorando maiúsculas/minúsculas

Conceitos: strcasecmp() ou tolower(), comparação

Bibliotecas: stdio.h, string.h, ctype.h

18. Substituidor de Caracteres

Objetivo: Substituir todas as ocorrências de um caractere por outro

Conceitos: strchr(), loops, manipulação de strings

Bibliotecas: stdio.h, string.h

Matrizes (Arrays Bidimensionais)

19. Soma de Matrizes

Objetivo: Somar duas matrizes 3x3

Conceitos: matrizes, loops aninhados, operações matriciais

Bibliotecas: stdio.h

20. Diagonal Principal

Objetivo: Exibir e somar elementos da diagonal principal de uma matriz

Conceitos: matrizes, índices [i][i]

Bibliotecas: stdio.h

21. Transposta de Matriz

Objetivo: Calcular a matriz transposta de uma matriz 3x3

Conceitos: matrizes, troca de índices [i][j] → [j][i]

Bibliotecas: stdio.h

stdlib.h - Conversões e Utilidades

22. Conversor de String para Número

Objetivo: Converter strings numéricas para inteiros e float

Conceitos: atoi(), atof(), validação de entrada

Bibliotecas: stdio.h, stdlib.h, string.h

23. Gerador de Números Aleatórios

Objetivo: Gerar 10 números aleatórios entre 1 e 100

Conceitos: rand(), srand(), time()

Bibliotecas: stdio.h, stdlib.h, time.h

24. Jogo de Adivinhação

Objetivo: Criar jogo onde usuário adivinha número aleatório

Conceitos: rand(), loops, condicionais

Bibliotecas: stdio.h, stdlib.h, time.h

🔴 NÍVEL DIFÍCIL (Avançado)

Alocação Dinâmica de Memória

25. Vetor Dinâmico

Objetivo: Criar vetor com tamanho definido pelo usuário usando malloc()

Conceitos: malloc(), free(), ponteiros

Bibliotecas: stdio.h, stdlib.h

```
// Exemplo de estrutura:
int *vetor;
int n;
printf("Tamanho do vetor: ");
scanf("%d", &n);
vetor = (int*) malloc(n * sizeof(int));
```

```
// ... usar o vetor ...
free(vetor);
```

26. Matriz Dinâmica

Objetivo: Criar matriz dinâmica com dimensões definidas pelo usuário

Conceitos: malloc(), ponteiros duplos, free()

Bibliotecas: stdio.h, stdlib.h

27. Cadastro Dinâmico de Alunos

Objetivo: Sistema que armazena nome e nota de N alunos dinamicamente

Conceitos: struct, malloc(), realloc(), arrays dinâmicos

Bibliotecas: stdio.h, stdlib.h, string.h

28. Lista de Compras Dinâmica

Objetivo: Permitir adicionar/remover itens dinamicamente

Conceitos: malloc(), realloc(), free(), gerenciamento de memória

Bibliotecas: stdio.h, stdlib.h, string.h

29. Comparação malloc vs calloc

Objetivo: Demonstrar diferença entre malloc() e calloc()

Conceitos: malloc(), calloc(), inicialização de memória

Bibliotecas: stdio.h, stdlib.h

```
// malloc - não inicializa
int *v1 = (int*) malloc(5 * sizeof(int));

// calloc - inicializa com zero
int *v2 = (int*) calloc(5, sizeof(int));
```

Projetos Integradores

30. Sistema de Cadastro Completo

Objetivo: Cadastro com menu (inserir, listar, buscar, editar, remover)

Conceitos: struct, funções, switch-case, arrays, strings

Bibliotecas: stdio.h, string.h, stdlib.h

31. Agenda Telefônica

Objetivo: Armazenar nome, telefone e email com busca

Conceitos: struct, arrays, funções, strcmp(), ordenação

Bibliotecas: stdio.h, string.h, stdlib.h

32. Analisador de Texto Completo

Objetivo: Contar palavras, caracteres, vogais, consoantes, frases

Conceitos: todas as funções de ctype.h e string.h

Bibliotecas: stdio.h, string.h, ctype.h

33. Jogo da Forca

Objetivo: Implementar jogo da forca completo

Conceitos: strings, arrays, funções, strchr(), loops, validação

Bibliotecas: stdio.h, string.h, ctype.h, stdlib.h

34. Calculadora com Histórico

Objetivo: Calculadora que guarda histórico de operações

Conceitos: struct, arrays dinâmicos, funções, malloc()

Bibliotecas: stdio.h, stdlib.h, string.h

35. Sistema de Notas com Estatísticas

Objetivo: Calcular média, mediana, moda, desvio padrão

Conceitos: funções matemáticas, ordenação, estatística

Bibliotecas: stdio.h, stdlib.h, math.h

locale.h - Internacionalização

36. Formatação de Moeda

Objetivo: Exibir valores monetários no formato brasileiro (R$ 1.234,56)

Conceitos: setlocale(), LC_MONETARY, formatação

Bibliotecas: stdio.h, locale.h

```
setlocale(LC_ALL, "pt_BR.UTF-8");
printf("Valor: R$ %.2f\n", 1234.56);
```

37. Conversor de Data

Objetivo: Exibir data em diferentes formatos e idiomas

Conceitos: setlocale(), LC_TIME, strftime()

Bibliotecas: stdio.h, locale.h, time.h

Algoritmos Clássicos

38. Ordenação - Bubble Sort

Objetivo: Implementar algoritmo de ordenação

Conceitos: algoritmos, loops aninhados, troca de valores

Bibliotecas: stdio.h

39. Busca Binária

Objetivo: Implementar busca binária em vetor ordenado

Conceitos: algoritmos de busca, recursão/iteração

Bibliotecas: stdio.h

40. Fibonacci com Memoization

Objetivo: Calcular Fibonacci usando alocação dinâmica para cache

Conceitos: recursão, malloc(), otimização

Bibliotecas: stdio.h, stdlib.h

📊 Tabela Resumo por Biblioteca

Biblioteca	Funções Principais	Exercícios
`stdio.h`	printf, scanf, fgets, fprintf	Todos
`string.h`	strlen, strcpy, strcmp, strcat, strtok, strchr	6, 15-18, 30-34
`ctype.h`	isalpha, isdigit, toupper, tolower, isspace	6-8, 12, 14, 32

`stdlib.h`	malloc, calloc, free, realloc, atoi, rand	22-29, 31, 33-35, 40
`locale.h`	setlocale	36-37
`math.h`	sqrt, pow, funções matemáticas	35

🎯 Dicas de Estudo

1. **Comece pelo fácil** - domine os fundamentos antes de avançar
2. **Pratique diariamente** - faça pelo menos 1-2 exercícios por dia
3. **Não pule etapas** - cada exercício prepara para o próximo
4. **Teste com diferentes entradas** - valide edge cases
5. **Refatore seu código** - melhore após fazê-lo funcionar
6. **Use debugger** - aprenda a depurar com gdb ou IDE

Boa sorte nos estudos! 🚀

Documento gerado em 20/01/2026