

Apêndice B. Exercícios de programação

Capítulo 4: Funções de Entrada e Saída

- 4.1 Faça um programa que leia 2 números reais e imprima a média aritmética entre eles.
- 4.2 Faça um programa escreva na tela o caracter ASCII e o respectivo códigos hexadecimal de um valor decimal digitado pelo usuário. [Sugestão: Use a função `putchar ()` para escrever os caracteres].
- 4.3 Faça um programa que leia um angulo (em graus) e imprima o valor do *seno*, *coseno* e *tangente* deste angulo.
- 4.4 Altere o programas 4.2 e 4.3 para que utilizem efeitos de impressão colorida.
- 4.5 O volume de um esfera de raio R é $V = \frac{4}{3}\pi R^3$. Faça um programa que leia um numero R e imprima o volume da esfera correspondente.
- 4.6 Faça um programa que leia uma frase e rescreva esta frase centralizada no topo da tela.

Capítulo 5: Estruturas de Controle

- 5.1 Faça um programa que leia 3 números e imprima o número de maior valor absoluto.
- 5.2 Faça um programa escreva na tela todos os caracteres ASCII, os respectivos códigos decimais e hexadecimais.
- 5.3 Faça um programa que imprima os 10 primeiros números primos.
- 5.4 Faça um programa que imprima os números ímpares no intervalo fechado $[a, b]$ (a e b escolhidos pelo usuário).
- 5.5 Faça um programa que leia uma frase digitada e imprima um relatório contendo: o número de palavras, o número de vogais e o número de letras digitadas.
- 5.6 Altere o programa 4.1 para que o usuário determine a quantidade de números manipulados.
- 5.7 Faça um programa que imprima os N primeiros números da série de Fibonacci: $1, 1, 2, 3, 5, 8, 13, \dots$. A fórmula de recorrência para esta série é $n_i = n_{i-1} + n_{i-2}$ para $i \geq 2$ pois $n_0 = n_1 = 1$.
- 5.8 Altere o programa 5.1 para que o usuário determine a quantidade de números manipulados.
- 5.9 Altere o programa 5.3 para que o usuário determine a quantidade de números manipulados.

- 5.10** Faça um programa que leia os três parâmetros *a*, *b*, *c* de uma equação de segundo grau e escreva suas raízes (reais ou complexas).
- 5.11** Faça um programa que execute um proteção de tela do seguinte modo: Ao iniciar o programa, um caracter (ou outro qualquer) percorra a tela em direções aleatórias apagando os caracteres de fundo. Ao se pressionar qualquer tecla o texto de fundo reaparece e o programa termina. Use as funções `gettext()` e `puttext()`.
- 5.12** Faça um programa que peça para o usuário adivinhar um número escolhido aleatoriamente entre 1 e 100. Se o usuário digitar um número errado, o programa responde o novo intervalo do número procurado. Se o usuário acertou o número procurado, o programa diz quantos palpites foram dados. Por exemplo:
 O número procurado está entre 1 e 100:
 Palpite: 45
 O número procurado está entre 1 e 44:
 Palpite: 27
 O número procurado está entre 28 e 44:
 Palpite: 36
 Parabéns! Você acertou o número em 3 tentativas.
- 5.13** Faça um programa que leia um valor inteiro de 0 a 1000 escreva o seu valor por extenso. Por exemplo:
 Digite valor: 279
 Extenso: duzentos e setenta e nove.
- 5.14** Faça um programa que coloque um caracter no centro da tela e permita o movimentação deste com o uso das setas.
- 5.15** Implemente um ‘rastros’ para o caracter do programa **5.14**. Sugestão: use os caracteres de preenchimento: 176d, 177d, 178d e 219d.
- 5.16** Faça um programa que desenhe um janela com bordas (simples ou duplas) em uma posição centralizada da tela. Pinte o interior da janela com alguma cor diferente do fundo da tela. [Sugestão: Use o laço `for...` para escrever as bordas e a função `clrscr()` para pintar o interior da janela].

Capítulo 6: Funções

- 6.1** Crie um função `float round(float r)` que faça o arredondamento de números reais: Por exemplo: $5 = \text{round}(5.4)$, $7 = \text{round}(6.5)$.
- 6.2** Crie uma função `int sim_nao(void)` que espera o usuário pressionar as teclas [*s*] ou [*n*] retornando 1 ou 0 respectivamente. Se o usuário pressionar qualquer outra tecla um som (de advertência) de 50 Hz por 250 ms é emitido.
- 6.3** Transforme o programa do exercício **5.16** em uma função com a seguinte declaração: `int borda(esq, sup, dir, inf, corf, corb)` onde *esq*, *sup*, *dir*, *inf* são as posições das bordas, *corf*, *corb* as cores do fundo e da borda da janela respectivamente. A função retorna 1 se houve algum erro na passagem dos parâmetros (*esq* > *dir*, por exemplo) e 0 caso contrário.
- 6.4** Faça uma função que determine se três números *a*, *b*, *c* formam um triângulo ou não. A função deve ter a seguinte declaração `int triângulo(float a, float b, float c)` onde o valor de retorno tem o seguinte significado:
 0: não forma triângulo,

- 1: triângulo qualquer,
- 2: triângulo isósceles,
- 3: triângulo equilátero.

6.5 Faça uma função que determine se um determinado número é primo ou não. A função deve ter a seguinte declaração `int primo(int N)` onde N é o valor a ser testado. A função deve retornar 1 se N é primo e 0 caso contrário. [Sugestão: Altere o programa do exercício **5.3**].

6.6 Transforme o programa do exercício **5.7** em uma função `int fib(int n)` que retorna o n -ésimo número de Fibonacci.

6.7 A *média elíptica*¹ (ou aritmético-geométrica) de dois números positivos a e b [$a < b$], é calculada do seguinte modo: Chamando $a_{n+1} = \sqrt{a_n b_n}$ e $b_{n+1} = (a_n + b_n) / 2$ respectivamente as médias geométrica e aritmética desses números obtemos uma seqüência de números a_0, a_1, a_2, \dots e b_0, b_1, b_2, \dots tal que $a_0 < a_1 < a_2 < \dots < b_2 < b_1 < b_0$. O limite desta seqüência é $m = a_\infty = b_\infty$. Por exemplo: a média elíptica de 4 e 7 é 5.5932... Faça uma função `double elip(double a, double b)` que calcule a média elíptica de a e b . [Sugestão: Use um laço `while(a < b)`...].

6.8 O maior divisor comum dos inteiros positivos a e b , que abreviamos como $mdc(a, b)$, é o maior número m tal que m é divisor tanto de a quanto de b . Por exemplo: $4 = mdc(20, 16)$, $7 = mdc(21, 7)$. O valor de m pode ser calculado com o seguinte algoritmo recursivo², de *Euclides*:

```
se a > b então mdc(a, b) é igual a
    b se resto(a, b) é 0
    mdc(b, resto(a, b)) caso contrário.
```

Faça uma função *recursiva* para o cálculo do máximo divisor comum de dois números.

6.9 Caso já não tenha feito assim, transforme a função *iterativa* do exercício **6.6** em uma função *recursiva*.

Capítulo 7: Vetores

7.1 Escreva um programa que leia um vetor de N números inteiros, ($N \leq 100$), inverta a ordem dos elementos do vetor e imprima o vetor invertido. Por exemplo o vetor: {1, 3, 5, 7} terá seus elementos invertidos: {7, 5, 3, 1}. **Observação:** É necessário inverter os elementos do vetor. Não basta imprimi-los em ordem inversa!

7.2 Escreva um programa que leia um vetor a de N números reais, ($N \leq 100$), e um outro real k e construa e imprima um outro vetor b cujos elementos são os respectivos elementos de a multiplicados por k . **Por exemplo:** $a = \{1, 2, 3\}$, $k = 5$, $b = \{5, 10, 15\}$.

7.3 Escreva duas funções: uma que **leia** um vetor v de n números inteiros, ($n \leq 100$), e outra que **escreva** este vetor. A declaração destas funções devem ser, respectivamente: `void le_vet(int v, int n)` e `void escreve_vet(int v, int n)`.

7.4 Escreva um programa que leia um vetor `gabarito` de 10 elementos. Cada elemento de `gabarito` contém um número inteiro 1, 2, 3, 4 ou 5 correspondente as opções corretas de uma prova objetiva. Em seguida o programa deve ler um vetor `resposta`, também de 10 elementos inteiros, contendo as respostas de um aluno. O programa deve comparar os dois vetores e escrever o número de acertos do aluno.

¹ Descoberta pelo matemático alemão *Carl F. Gauss*. Ver LIMA, E. L., *Meu Professor de Matemática*?, p.123

² Ver SMITH, J. D. *Design and Analysis of Algorithms*, Boston: PWS-Kent Pub. Co. 1989 p.272

- 7.5 Escreva uma função `int min_vet(float v[], int n)` receba um vetor e retorne o índice do **menor** elemento deste vetor.
- 7.6 Escreva uma função `int remove_dup(float v[], int n)` receba um vetor e verifique a existência de elementos duplicados. Caso não existam elementos duplicados retorne 0. Caso existam, **remova** estes elementos (deixando apenas um) e retorne o número de elementos removidos.
- 7.7 Escreva uma função `void insert(float v[], int n, float valor, int pos)` que faça a **inserção** de valor na posição `pos` do vetor `v`, deslocando os demais elementos.
- 7.8 Transforme o *programa* do exemplo `e0705.cpp` em uma *função* `void ordem(int v, int n)` que **ordene** os elementos de um vetor `v` de `n` elementos inteiros.
- 7.9 Escreva uma função `int merge(float r[], float s[], float v[], int n, int m)` receba um vetor `r` de `n` elementos e outro vetor `s` de `m` elementos e construa um vetor `v` com os elementos de `r` e `s`, **ordenado e não duplicado**. A função deve retornar o tamanho do vetor `v` construído. Sugestão: Utilize as funções dos exercícios 7.6, 7.7 e 7.8.
- 7.10 A função do exercício 7.9 pode ser entendida como uma função que retorna a união entre dois conjuntos. Escreva uma função `int intersec(float r[], float s[], float v[], int n, int m)` que construa um vetor `v` com a **interseção** entre `r` e `s`, **ordenados**. A função deve retornar o tamanho do vetor `v` construído.

- 7.11 Escreva uma função `void desordem(int v, int n)` que **desordene** os elementos de um vetor `v` (não necessariamente ordenado) de `n` elementos inteiros. Sugestão: use o seguinte algoritmo:

```
para i de n-1 até 0 faça
    j ← valor aleatório entre 0 e i
    v[i] ↔ v[j]
fim faça
```

Observação: Esta rotina pode ser usada para simular o processo de embaralhar as cartas de um baralho.

- 7.12 Escreva uma função `int find(char v[], char t[], int m, int n)` que receba um vetor `v` de `m` elementos e um vetor `t` de `n` elementos ($n < m$). Esta função deve verificar a ocorrência do padrão `t` em `v` ou não. Se houver, deve retornar a posição inicial da primeira ocorrência. **Por exemplo:** se `v={As bananas do Panamá são bacanas}` e `p={anas}` deve retornar 6. Caso não haja ocorrência, retorne -1. **Observação:** Algoritmos como esses são usados em editores de texto³.

- 7.13 O **produto escalar** entre dois vetores pode ser definido⁴ como: $e = \vec{u} \cdot \vec{v} = \sum_{i=0}^n u_i v_i$, onde u_i e v_i são os elementos do vetor. Escreva uma função `float prod_esc(float u, float v, int n)` que receba dois vetores `u` e `v` de `n` elementos reais e retorne o valor do produto escalar entre eles.

³ Ver o algoritmo de Knuth-Morris-Pratt em SMITH (op. Cit.), p 294

⁴ Ver SPIEGEL, M. R., *Análise Vetorial*, São Paulo: McGraw-Hill. 1976, p. 23.

7.14 O *código Morse* foi muito usado no tempo do telégrafo para transmitir mensagens. Neste sistema cada símbolo (letra, número ou sinal de pontuação) é enviado por fio em uma série de pulsos elétricos curtos ou longos (*pontos* ou *traços*) conforme a tabela abaixo:

a	.-	b	-...	c	-.-. .	d	-..	e	.	f	..-.
g	--.	h	i	..	j	..---	k	-.-	l	..-..
m	--	n	-. .	o	---	p	..--.	q	--.-	r	.-.
s	...	t	-	u	..-	v	...-	w	..--	x	-.-. .
y	-.--	z	--..	1	..----	2	..----	3	...--	4-
5	6	-.....	7	--....	8	----..	9	-----	0	-----
.	.-.-.-	?	..-.-.-	,	---.-.-	:	---.-.-				

Escreva um programa que leia uma frase digitada pelo usuário e emita pelo alto-falante do PC o som correspondente a uma transmissão completa em código Morse. **Sugestão:** Crie uma tabela `código[40][5]` em que cada linha represente um sinal contendo os números 1 (ponto), 2 (traço) ou 0 (fim do código).

7.15 Existe um problema famoso no xadrez chamado *Problema das 8 damas*: consiste em dispor sobre o tabuleiro (de 8 x 8 casas) do jogo um conjunto de **8 damas** de tal forma que nenhuma dama fique na mesma *linha*, *coluna* ou *diagonal* que outra. Escreva um programa que calcule pelo menos uma solução deste problema. **Sugestão:** crie um vetor `tab[8][8]` contendo 0 para uma casa vazia e 1 para uma casa ocupada. Escreva uma função que **crie** as configurações e outra rotina que **teste** a solução.

7.16 As populares calculadoras *HP (Hewlett-Packard)* usam a notação RPN (*Reverse Polish Notation*) para o cálculo de expressões numéricas. Este tipo de notação torna mais fácil o cálculo de expressões complexas. Cada valor digitado é guardado em uma **pilha de dados** e cada **tecla de operação** efetua uma operação entre os últimos dois valores da pilha. Por exemplo, para efetuar a expressão $2+5*3$ digitamos:

- [5] (*colocamos o primeiro valor na pilha*),
- [e r e r] [3] (*deslocamos 5 para a 2ª posição da pilha e colocamos 3 para o 1º valor na pilha*),
- [*] (*efetuamos a multiplicação dos dois valores, o valor 15 aparece na 1ª posição da pilha*),
- [2] (*deslocamos 15 para a 2ª posição da pilha e colocamos 3 para o 1º valor na pilha*)
- [+] (*adicionamos 2 ao resultado, 17 aparece na 1ª posição da pilha*).

Escreva um programa que simule uma **calculadora RPN** de 4 operações (+, -, *, /) utilizando vetores para representar a pilha de dados.

7.17 *Reverse* é o nome de um antigo jogo de tabuleiro, que pode ser facilmente implementado em um computador: consiste de um tabuleiro de 3x3 casas, com um disco branco ou preto dispostos, inicialmente, de modo aleatório em cada casa.

```
[1][2][3]
[4][5][6]
[7][8][9]
```

Ao selecionar uma das casas o jogador **reverte** a cor do disco daquela casa e de algumas casas vizinhas conforme o esquema acima. O objetivo do jogo é reverter todas as casas para uma mesma cor. Escreva um programa que simule o tabuleiro do jogo de *Reverse*.

Ao pressionar	Reverte:
[1]	[1],[2] e [4]
[2]	[2],[1] e [3]
[3]	[3],[2] e [6]
[4]	[4],[1] e [7]
[5]	[5],[2],[4],[6] e [8]
[6]	[6],[3] e [9]
[7]	[7],[4] e [8]
[8]	[8],[7] e [9]

7.18 Caso já não tenha feito assim, Rescreva o programa do exercício 7.1 tornando-a recursiva.

7.19 Escreva uma função que receba duas matrizes de ordem m e construa o produto matricial entre eles.

Capítulo 8: Ponteiros

8.1 Rescreva a função `round()` do exercício 6.1 para que receba o endereço da variável a ser arredondada. A nova função deve ter a seguinte declaração `void round(float *r)` e deve arredondar o próprio número passado.

8.2 Escreva uma função `void troca(int *a, int *b)` que permuta entre si os valores de a e b .

8.3 Escreva uma função `void stat(float vet, int N, float *med, float *dsvpd)` que receba um vetor de números reais vet , seu tamanho N e calcule a média aritmética med e o desvio padrão⁵ $dsvpd$ destes valores.

8.4 Escreva uma função `void extremos(float vet, int N, float *max, float *min)` que receba um vetor de números reais vet , seu tamanho N e determine o maior e o menor (max e min) destes valores.

8.5 Rescreva o programa do exemplo 7.1 para que, usando alocação dinâmica de memória, construa um vetor de N elementos, sendo N digitado pelo usuário.

8.6 Escreva um programa que, usando alocação dinâmica de memória, construa um vetor de N elementos gerados aleatoriamente no intervalo $[0, 10]$, sendo N digitado pelo usuário. Em seguida este programa deve chamar a função `remove_dup()` do exercício 7.6. É necessária alguma modificação no código desta função?

8.7 Escreva uma rotina `void graf(float a float b, float (*pf)(float))` que receba um endereço de uma função no ponteiro pf , os valores de extremos a e b e desenhe o gráfico da função apontado por pf .

⁵ Ver GIOVANNI, J. R., BONJORNO, J. R., *Matemática*, vol. 3, São Paulo: FTD, s.d. p. 320.