Instituto Federal do Espírito Santo — IFES (Campus Serra)
Coordenadoria de Informática — Cin

Caderno de exercícios

Linguagem C e Estrutura de dados

`v23.2.3`

Prof. Thiago M. Paixão
`thiago.paixao@ifes.edu.br`

1 Estruturas sequenciais

1. Faça um programa que leia a temperatura em graus Farenheit (F), converta-a para graus Celsius e mostre a temperatura convertida na tela. A fórmula para conversão é

$$C = 5 \times (F - 32)/9. \tag{1}$$

2. O mesmo que o exercício anterior, porém convertendo de Celsius para Farenheit.
3. Faça um programa que leia três notas e imprima na tela a média aritmética delas.
4. A prefeitura da Serra abriu uma linha de crédito para os funcionários estatutários. O valor máximo da prestação não poderá ultrapassar 30% do salário bruto (salário mais benefícios sem desconto de impostos). Faça um programa que leia o salário bruto de uma pessoa e imprima na tela o valor máximo possível de prestação para este funcionário.
5. Faça um programa para calcular as raízes reais de uma equação do 2º grau (ax^2+bx+c) dado que seus coeficientes são informados pelo usuário. Utilize uma variável real `delta` para armazenar o resultado de $b^2 - 4ac$.
6. Faça um programa que peça o tamanho de um arquivo para download (em MB) e a velocidade de um link de Internet (em Mbps), calcule e informe o tempo aproximado de download do arquivo usando este link EM MINUTOS.
7. Faça um programa que leia dois valores inteiros armazenando-os em variáveis `x` e `y`. Ao final da execução, os valores destas variáveis devem estar trocados. Restrição: você não deve utilizar variável temporária, ou seja, apenas duas variáveis podem ser utilizadas.
8. Escreva um programa em C que permita ao usuário extrair um dígito específico de uma sequência de 4 dígitos. O programa deve solicitar ao usuário a sequência de dígitos e a posição x do dígito que deseja extrair (0 a 4), e então exibir o dígito correspondente.

   Dica: Lembre-se de que ao dividir um número inteiro por 10, você "remove" o último dígito. E ao calcular o resto da divisão desse número por 10, você obtém o último dígito isoladamente.

2 Condição e seleção

9. Faça um programa que leia um número inteiro maior que zero e informe se é par ou ímpar.
10. A prefeitura da Serra abriu uma linha de crédito para os funcionários estatutários. O valor máximo da prestação não poderá ultrapassar 30% do salário bruto. Faça um programa que leia o salário bruto e o valor da prestação, e informe se o empréstimo pode ou não ser concedido.

11. Uma grande companhia química paga seus vendedores por comissão. Os vendedores recebem R$ 200,00 por semana mais 9% de suas vendas brutas naquela semana. Por exemplo, um vendedor que vender o equivalente a R$ 500,00 em produtos em uma semana recebe R$ 200,00 mais 9% de R$ 500,00, ou seja, um total de R$ 245,00. Se as vendas ultrapassarem R$ 1000,00, o vendedor recebe também um prêmio de R$ 800,00. Faça um programa que receba as vendas brutas de um vendedor na última semana e escreva o valor a ser recebido pelo funcionário.

12. Calcule a média aritmética das três notas de um aluno e mostre, além do valor da média, uma mensagem de acordo com as condições explicitadas a seguir:

Condição	Mensagem
Média $\geq$ 7,0	Aprovado
5,0 $\leq$ Média $<$ 7,0	Recuperação
Média $<$ 5,0	Reprovado

13. Faça um programa que verifique se três valores a, b e c podem ser os comprimentos dos lados de um triângulo. Caso positivo, seu programa deve informar se o triângulo é equilátero, isósceles ou escaleno. Caso contrário, seu programa deve escrever a mensagem "Não formam triângulo".
Obs. 1: Um triângulo equilátero possui os comprimentos dos três lados iguais.
Obs. 2: Um triângulo isósceles possui pelo menos dois lados de mesma medida.
Obs. 3: Um triângulo escaleno possui todos os seus lados com medidas diferentes.
Obs. 4: Supor que os valores lidos são inteiros e positivos.
Obs. 5: Em todo triângulo, qualquer lado tem medida menor que a soma das medidas dos outros dois.

14. Uma imagem digital possui largura w e altura h. Um pixel dessa imagem está associado a uma posição (x, y) no plano da imagem. Faça um programa que receba w, h, x e y, e informe se corresponde a um pixel da imagem especificada.

15. Faça um programa que calcule e imprima o valor da conta de água, a partir da leitura do consumo de água do mês anterior e do mês atual marcado no hidrômetro. Sabe-se que a conta de água é formada pela tarifa de água somada à tarifa de esgoto (2,5% da conta de água) e à tarifa de conservação do hidrômetro (R$ 5,00). O consumo de água é de acordo com a tabela mostra na sequência:

Consumo (m^3)	Tarifa (R\$/m^3)
0 até 10 (inclusive)	0,69
11 até 15 (inclusive)	1,17
16 até 25 (inclusive)	1,48
Acima de 25	1,60

16. Um vendedor necessita de um programa que calcule o preço total devido por um cliente. O algoritmo deve receber o código de um produto e a quantidade comprada e calcular o preço total usando a seguinte tabela:

Código	Preço unitário (R$)
1001	5,32
1324	6,45
6548	2,37
987	5,32
7623	6,45

Obs.: Mostrar a mensagem "Código inválido" caso o valor digitado não esteja de acordo com a tabela.

17. Escreva um programa que leia o número correspondente ao mês atual e os dígitos (somente os

quatro números) de uma placa de veículo, e através do número finalizador da placa (algarismo da casa das unidades) determine se o IPVA do veículo vence no mês corrente. Confira a tabela na sequência que relaciona o final da placa como mês para pagamento de IPVA.

Final	Mês	Final (cont.)	Mês (cont.)
1	Janeiro	6	Junho
2	Fevereiro	7	Julho
3	Março	8	Agosto
4	Abril	9	Setembro
5	Maio	0	Outubro

Obs.: O número da placa deve ser lido como tipo inteiro.

```c
#include <stdio.h>

int main() {
   int last_digit;

   printf("Digite o último dígito da placa do veículo (0-9): ");
   scanf("%d", &last_digit);

   switch (last_digit) {
      case 1:
         printf("O IPVA vence em Janeiro.\n");
         break;
      // Complete os casos para os outros dígitos e meses aqui...

      default:
         printf("Dígito inválido.\n");
         break;
   }

   return 0;
}
```

18. Escreva um programa que leia um peso na Terra e um número relativo a um planeta e imprima o valor do seu peso neste planeta. A relação de planetas é dada a seguir com o valor das gravidades relativas ã Terra:

n	Gravidade relativa	Planeta
1	0,37	Mercurio
2	0,88	Vênus
3	0,38	Marte
4	2,64	Júpiter
5	1,15	Saturno
6	1,17	Urano

19. Crie um programa que receba um valor x e mostre o valor de $f(x)$ definido a seguir:

$$f(x) = \begin{cases} 1, \text{se } x \leq 1 \\ x, \text{se } 1 < x \leq 2 \\ x^2, \text{se } 2 < x \leq 3 \\ x^3, \text{se } x > 3 \end{cases}$$

20. Faça um programa para calcular as raízes reais de uma equação do 2º grau (ax^2+bx+c) dado que seus coeficientes são informados pelo usuário. Utilize uma variável real `delta` para armazenar o resultado de $b^2 - 4ac$. Informe o usuário se a equação tem 2, 1 ou nenhuma raiz real.

3 Funções e ponteiros

21. Escreva um programa em linguagem C que utilize uma função chamada `quadrado` para calcular o quadrado de um número inteiro. A função `quadrado` deve receber um argumento inteiro `x` e retornar o valor do quadrado de `x`. O programa principal deve ler um número inteiro fornecido via teclado, chamar a função `quadrado` para calcular o quadrado e imprimir o resultado.

```c
#include <stdio.h>

int quadrado(int x) {
...
}

int main() {
   ...
   return 0;
}
```

22. Implemente a função `maior3` que recebe três inteiros e retorna o maior valor dentre eles. A função deve se apoiar, estritamente, na função `maior2`, ou seja, não pode utilizar o comando `if` ou `while`.

```c
#include <stdio.h>

int maior2(int a, int b)
{
   if (a >= b)
     return a;
   return b;
}

int maior3(int a, int b, int c)
{
...
}

int main()
{
   int a, b, c, maior;
   // entrada
   scanf("%d %d %d", &a, &b, &c);
   // processamento
```

```
    maior = maior3(a, b, c);
    // saída
    printf("%d", maior);

    return 0;
}
```

Exemplo de entrada: 10 11 4
Saída esperada: 11

23. Uma empresa deseja calcular o bônus salarial de seus funcionários com base no número de anos de trabalho e no salário atual. A regra para calcular o bônus é a seguinte:

    - Se o funcionário tiver mais de 5 anos de trabalho na empresa e o salário atual for superior a R$5000,00, o bônus será de 10% do salário.
    - Caso contrário, o bônus será de 5% do salário.

    Implemente uma função em linguagem C com assinatura `float calcular_bonus(int anos_trabalho , float salario)` para realizar esse cálculo. Escreva um programa que capture o número de anos de trabalho e o salário atual fornecidos via teclado, chame a função `calcular_bonus` para calcular o bônus e imprima o valor do bônus calculado.

24. Uma universidade deseja converter notas de seus estudantes de uma escala alfabética para uma escala numérica de acordo com a seguinte correspondência:

Nota	Valor Numérico
A	10
B	8
C	6
D	4
F	0

    Implemente uma função em linguagem C com assinatura `int converter_nota(char nota)` que recebe uma nota alfabética como argumento e retorna a nota equivalente na escala numérica. Caso a nota alfabética seja inválida, a função deve retornar -1. Escreva um programa completo que leia uma nota alfabética fornecida via teclado, chame a função `converter_nota` para realizar a conversão e imprima a nota equivalente na escala numérica ou uma mensagem de erro caso a nota seja inválida.

25. Implemente um programa em linguagem C que calcule o máximo divisor comum (MDC) de dois números inteiros positivos lidos do teclado utilizando a fórmula recursiva a seguir:

$$MDC(x, y) = \begin{cases} y, & \text{se } x \bmod y = 0, \\ MDC(y, x \bmod y), & \text{caso contrário.} \end{cases} \tag{2}$$

    O MDC deve ser calculado por uma função recursiva.

26. O máximo divisor comum de três números inteiros positivos, $MDC(x, y, z)$, pode ser calculado como $MDC(MDC(x, y), z)$. Escreva um programa que leia três números inteiros fornecidos via teclado e imprima o MDC deles, usando a função MDC apresentada no texto.

27. Implemente uma função em linguagem C chamada `coeficiente_binomial` que calcula o coeficiente binomial $\binom{n}{k}$ usando uma abordagem recursiva. O coeficiente binomial é dado por:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}, \tag{3}$$

com $\binom{n}{0} = \binom{n}{n} = 1$. O programa deve receber dois números inteiros, `n` e `k`, fornecidos pelo usuário e imprimir o coeficiente binomial correspondente.

28. Analise os códigos abaixo e determine o valor das variáveis ao final da execução:

(a)
```
int i=34, j;
int *p;
p = &i;
*p++;
j = *p + 33;
```

(b)
```
int i=7, j=5;
int *p;
int **q;
p = &i;
q = &p;
c = **q + j;
```

(c)
```
int x = 10;
int y = 20;
int *p, *q;

p = &x;
q = &y;

*p = *p + *q;
*q = *p - *q;
```

(d)
```
int a = 5;
int b = 7;
int *p, *q;

p = &a;
q = &b;

a = a + b;
b = a - b;
*p = *q + b;
```

29. Escreva um programa em linguagem C que realize o seguinte:

    (a) Leia um número inteiro do usuário.

    (b) Declare um ponteiro que aponte para a variável inteira lida.

    (c) Multiplique o valor da variável por 2, usando o ponteiro.

    (d) Imprima o valor original e o valor resultante da multiplicação.

Exemplo de Execução

```
Digite um número inteiro: 7
Valor original: 7
Valor após multiplicação por 2: 14
```

30. Escreva um programa em linguagem C que realize o seguinte:
    (a) Declare uma variável inteira.
    (b) Declare um ponteiro que aponte para a variável inteira.
    (c) Utilize o operador `sizeof` para imprimir o tamanho do ponteiro.
    (d) Utilize o operador `sizeof` para imprimir o tamanho da variável apontada pelo ponteiro.

    Exemplo de Execução

    ```
    Tamanho do ponteiro: 8 bytes
    Tamanho da variável apontada: 4 bytes
    ```

31. Implemente uma função com assinatura `int soma_com_ponteiros(int *a, int *b)`. A função `soma_com_ponteiros` deve receber dois ponteiros para inteiros `a` e `b` como argumentos, calcular a soma dos valores apontados pelos ponteiros e retornar o resultado da soma. Para fins de teste, implemente o programa principal para que lê =dois números inteiros fornecidos via teclado, chama a função `soma_com_ponteiros` para calcular a soma e imprime o resultado.

    Exemplo de Execução

    ```
    Digite o valor de a: 5
    Digite o valor de b: 10
    A soma de 5 e 10 é: 15
    ```

32. Implemente uma função com assinatura `void trocar_valores(int *a, int *b)` que troque os valores de duas variáveis inteiras apontadadas pelos ponteiros `a` e `b`. A seguir, implemente o programa principal que leia dois números inteiros fornecidos via teclado, chame a função `trocar_valores` para trocar os valores e imprima os novos valores das variáveis.

    Exemplo de Execução

    ```
    Digite o valor de a: 5
    Digite o valor de b: 10
    Valores antes da troca: a = 5, b = 10
    Valores depois da troca: a = 10, b = 5
    ```

4 Repetição

As questões que versam sobre implementação de função requerem a implementação do código completo para teste.

33. Desenvolva um programa que leia dois valores a e b ($a \leq b$) e mostre os seguintes resultados: (i) todos os valores em $[a, b]$; todos os valores ímpares em $[a, b]$; (iii) todos os valores ímpares em $[a, b]$ e múltiplos de 3.

34. Elabore um programa que leia um número de entrada (n) que indicará a quantidade de números a serem lidos. Em seguida, leia n números (conforme o valor informado anteriormente) e imprima o triplo de cada um.

35. Faça um programa que leia um valor n indicando a quantidade de valores a ler em seguida. Um número deve ser lido por vez e seu programa deve classificá-lo como positivo ou negativo.

36. Escreva um programa que leia n valores, um de cada vez, e conte quantos destes valores são negativos, escrevendo esta informação na tela.

37. Faça um programa que calcule a média aritmética de vários valores inteiros positivos, inseridos pelo usuário. O final da leitura acontecerá quando for lido um valor negativo.

38. Faca um programa que apresente na tela a tabela de conversão de graus Celsius para Fahrenheit no intervalo de -100°C a 100°C com valores igualmente espaçados (5 em 5).

39. Escreva um programa que calcule a média dos números digitados pelo usuário se eles forem pares. Termine a leitura se o usuário digitar 0.

40. Escreva um programa que leia n valores e encontre o maior e o menor deles. Mostre o resultado.

41. Faça um programa que imprima a média de n números (n é um valor positivo lido do teclado) excluindo o menor e o maior deles. Seu programa deve tratar casos em que $n < 3$ exibindo uma mensagem de erro.

42. Crie um programa que leia uma sequência composta por dígitos 0 e 1. Ao concluir a entrada, exiba a contagem de zeros seguida pela contagem de uns presentes na sequência. A sequência de entrada será encerrada quando um número negativo for inserido.

    Exemplo de Execução

    ```
    Digite uma sequência (0 ou 1, -1 para encerrar):
    1
    0
    1
    0
    0
    1
    1
    -1

    Quantidade de 0's: 3
    Quantidade de 1's: 4
    ```

43. Faça um programa que leia um valor n e logo após um número x. Em seguida, leia n valores e, ao final imprima em qual posição x aparece. Caso x não esteja na sequência, imprimir a mensagem *Não.*

    Exemplo de Execução

    ```
    Digite o valor de n: 6
    Digite o valor de x: 9
    Digite 6 valores para a sequência:
    2 5 9 4 8 1
    O valor 9 aparece na posição 2.

    Digite o valor de n: 4
    Digite o valor de x: 7
    Digite 4 valores para a sequência:
    3 1 4 6
    Não.
    ```

No primeiro exemplo, o programa encontra o valor 9 na posição 2 da sequência e imprime a mensagem "O valor 9 aparece na posição 2."No segundo exemplo, o valor 7 não está na sequência e o programa imprime "Não."

44. Faça um programa que leia um valor x inteiro de 1 a 100 e em seguida leia um valor inteiro positivo n. Seu algoritmo deve realizar n sorteios[1] de números inteiros no intervalo de 1 a 100 e indicar quantas vezes x foi sorteado.

45. Escreva um programa em que é declarada uma variável contendo o valor de π (com 10 casas decimais) e leia o raio R de um círculo. O algoritmo deve calcular e exibir a área do círculo. Isso é repetido varias vezes até que o usuário responda 'N' (não) para a pergunta: "Deseja calcular mais áreas: Sim (S) ou Não (N)?".

46. Faça um programa que leia um número real x e um inteiro n e imprima $x * n$. Utilize apenas o comando `for` e a operação aritmética de adição.

47. Faça um programa que leia um número real x e um inteiro n e imprima x^n. Utilize apenas o comando `for` e a operação aritmética de adição (ver questão anterior).

48. Escreva um programa que calcula $n!$. Utilize como estrutura de repetição o comando `while`.

49. A prefeitura de uma cidade fez uma pesquisa entre seus habitantes, coletando dados sobre o salário e número de filhos. A prefeitura deseja saber:

    (a) Média do salário da população.

    (b) Média do número de filhos.

    (c) Maior salário.

    (d) Percentual de pessoas com salário até R$ 100,00.

    Faça um programa que coleta essas informações até que o usuário responda 'N' (não) para a pergunta: "Deseja continuar coletando informações: Sim (S) ou Não (N)?".

50. Escreva um programa que leia o tamanho (n), o primeiro termo (a_1) e a razão (q) de uma Progressão Geométrica (PG). Seu programa deve calcular e imprimir o n-ésimo termo desta PG através da fórmula $a_n = a_1 q^{n-1}$.

51. Um número perfeito é aquele que é igual à soma dos seus divisores (p. ex., $6 = 1 + 2 + 3$, $28 = 1 + 2 + 4 + 7 + 14$). Faça um programa que gera e escreve os cinco primeiros números perfeitos.

52. Considere o conjunto $X = \{-5.0,\ -4.5,\ -4.0,\ \ldots,\ 4.5,\ 5.0\} \subset \mathbb{R}$. Faça um algoritmo leia dois valores a e b e imprima uma tabela com os valores de $(x, y) \in X^2$ e $f(x, y)$ (Equação 4). Mostre o par (x, y) que minimiza $f(x, y)$. Imprima utilizando '*' a parábola $f(0, y)$ (rotacionada em 90°), $y \in X$.

$$f(x, y) = \frac{x^2}{a^2} + \frac{y^2}{b^2}. \tag{4}$$

Exemplo ($a = 1$, $b = 1$):

```
x       y      f(x,y)
-5.0    -5.0   50.0
-5.0    -4.5   45.25
-5.0    -4.0   41.0
-5.0    -3.5   37.25
```

[1] Pesquise sobre geração de números aleatórios.

```
...
5.0     3.5   37.25
5.0     4.0   41.0
5.0     4.5   45.25
5.0     5.0   50.0

(0,0) é o par que minimiza f(x,y)
```

"*Pringles. An example of a hyperbolic paraboloid.*" Fonte: `https://en.wikipedia.org/wiki/Paraboloid`

53. O comprimento de um polígono regular P de n lados inscrito numa circunferência C de raio R_C e comprimento $L_C = 2\pi R$ pode ser calculado por:

$$L_P = 2 \cdot R_C \cdot n \cdot sen\left(\frac{\pi}{n}\right). \tag{5}$$

À medida que se aumenta o número de lados do polígono (n), seu comprimento se aproxima do comprimento da circunferência circunscrita. Faça um algoritmo que leia R, e mostre uma tabela relacionando o resíduo $L_c - L_p$ para $n = 3, 4, 5, \ldots, N_{max}$, sendo N_{max} um valor também lido pelo algoritmo. Seu algoritmo também deve informar para qual valor de n o resíduo se torna inferior a 2% de L_C.

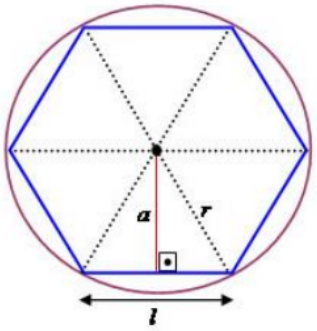

Figura 1

54. Complete a função `calcula_resto` que recebe dois inteiros positivos a e b, e retorna o resto da divisão (inteira) de a por b. Você não poderá utilizar o operador `%`. *Dica: utilize subtrações sucessivas.*

```c
#include <stdio.h>

// Calcula o resto da divisão de a por b
int calcula_resto(int a, int b)
{
```

```
...
}

int main()
{
   int a, b, resto;

   printf("Digite dois números inteiros positivos: ");
   scanf("%d %d", &a, &b);

   resto = calcula_resto(a, b);

   printf("O resto da divisão de %d por %d é: %d\n", a, b, resto);

   return 0;
}
```

Exemplo de Execução

```
Digite dois números inteiros positivos: 10 4
O resto da divisão de 10 por 4 é: 2

Digite dois números inteiros positivos: 17 5
O resto da divisão de 17 por 5 é: 2
```

55. Implemente uma função com a assinatura `int fatorial_div(int n)` que retorna o valor do fatorial de m, onde m é o maior divisor positivo de n, $m < n$.

Exemplo de Execução

```
Digite um número inteiro positivo: 8
O maior divisor positivo de 8 é 4.
O fatorial de 4 é 24.

Digite um número inteiro positivo: 12
O maior divisor positivo de 12 é 6.
O fatorial de 6 é 720.
```

56. Implemente uma função com assinatura `int soma_dos_digitos(int n)` que calcula e retorna a soma dos dígitos de um número inteiro não negativo `n`.

Exemplo de Execução

```
Digite um número inteiro não negativo: 12345
A soma dos dígitos de 12345 é: 15

Digite um número inteiro não negativo: 9876
A soma dos dígitos de 9876 é: 30
```

57. Um número primo é um número inteiro positivo maior que 1 que não possui divisores além de 1 e ele mesmo. Dado um número inteiro positivo, verifique se ele é um número primo. Implemente uma função com assinatura `int eh_primo(int numero)` que recebe um número inteiro positivo como parâmetro e retornar 1 se o número for primo, ou 0 caso contrário.

Exemplo de Execução

```
Digite um número inteiro positivo: 17
O número 17 é primo.

Digite um número inteiro positivo: 25
O número 25 não é primo.
```

5 Vetores

58. Implemente um programa que leia os elementos de um vetor de valores reais e mostre a soma e a média dos seus valores na tela.

    Exemplo de Execução

    ```
    Digite o tamanho do vetor: 5
    Preencha o vetor com 5 valores reais:
    Valor 1: 10.5
    Valor 2: 15.3
    Valor 3: 7.2
    Valor 4: 8.8
    Valor 5: 12.1

    Soma dos valores: 54.9
    Média dos valores: 10.98
    ```

59. Crie um programa que solicita ao usuário o tamanho n de um vetor de números inteiros (v) e realiza a leitura dos seus elementos. Após o preenchimento, o programa calculará um segundo vetor (s), onde cada elemento é calculado como a soma acumulada dos elementos correspondentes do vetor original, usando a seguinte fórmula:

$$s[i] = \sum_{k=0}^{i} v[k] \text{ para } i = 0, 1, \ldots, n-1.$$

    O programa, em seguida, exibirá ambos os vetores, permitindo ao usuário visualizar como os valores se acumulam ao longo do vetor original.

60. Leia um conjunto de valores reais armazenando-os em um vetor. O final da leitura deve ocorrer quando um valor negativo for dado como entrada. Em seguida, calcule a média do vetor e mostre-a na tela. Na linha seguinte, imprima todos elementos maiores que a média.

    Exemplo de execução

    ```
    Digite os valores reais (insira um valor negativo para encerrar):
    5.5
    7.2
    4.8
    -1

    Média dos valores: 5.83
    Valores maiores que a média:
    7.2
    ```

61. Faça um programa que leia um caractere c, e, em seguida, n posições de um vetor de caracteres. Seu programa deverá imprimir todos os índices em que forem encontrados c no vetor.

Assinatura da função:

```c
void encontre_indices(char vetor[], int tamanho, char c);
```

Fragmento de código a ser completado:

```c
#include <stdio.h>

void encontre_indices(char vetor[], int tamanho, char c) {
   // Seu código aqui
}

int main() {
   char c;
   int n;

   printf("Digite o caractere a ser procurado (c): ");
   scanf(" %c", &c);

   printf("Digite o tamanho do vetor (n): ");
   scanf("%d", &n);

   char vetor[n];

   printf("Digite os caracteres do vetor (separados por espaço ou enter):\n");
   for (int i = 0; i < n; i++) {
      scanf(" %c", &vetor[i]);
   }

   encontre_indices(vetor, n, c);

   return 0;
}
```

Exemplo de execução:

```
Digite o caractere a ser procurado (c): a
Digite o tamanho do vetor (n): 9
Digite os caracteres do vetor (separados por espaço ou enter):
a
b
c
a
d
e
f
a
g

O caractere 'a' foi encontrado nos seguintes índices:
```

```
0 3 7
```

62. Faça um programa que leia um caractere c, e, em seguida, n posições de um vetor de caracteres. Seu programa deverá imprimir todos os índices em que forem encontrados c no vetor.

Exemplo de execução

```
Digite o caractere a ser procurado (c): a
Digite o tamanho do vetor (n): 9
Digite os caracteres do vetor (separados por espaço ou enter):
a
b
c
a
d
e
f
a
g

O caractere 'a' foi encontrado nos seguintes índices:
0 3 7
```

63. Numa certa eleição, existem 5 candidatos identificados pelos valores $1, 2, \ldots, 5$. Uma votação pode ser representada como um vetor em que cada posição indica um eleitor e o conteúdo dessa posição seu voto. Por exemplo, a sequência $(1, 2, 1, 2, 3, 5)$ indica a existência de seis eleitores, sendo que os candidatos 1 e 2 obtiveram dois votos e os candidatos 3 e 5 obtiveram um voto cada. Construa um programa que leia os votos de n eleitores, indique quanto cada candidato recebeu em votos e qual candidato obteve o maior número de votos.

Exemplo de execução

```
Digite o número de eleitores (n): 6
Digite os votos dos eleitores (valores de 1 a 5, separados por espaço ou enter):
1
2
1
2
3
5

Resultado da Eleição:
Candidato 1: 2 votos
Candidato 2: 2 votos
Candidato 3: 1 voto
Candidato 4: 0 votos
Candidato 5: 1 voto
Candidato Vencedor: Candidato 1 e Candidato 2 (Empate)
```

64. Elabore uma função com assinatura `int sao_iguais(int v1[], int v2[], int n1, int n2 ↪ )` que retorna 1 se o vetor `v1`, com `n1` elementos, for igual a `v2`, este com tamanho `n2`. Caso contrário, seu programa deve retornar 0.

65. Implemente uma função com assinatura `int count_consonants(char str[], int n)` que re-

torna o número de consoantes num vetor de caracteres `str` composto apenas por vogais, consoantes e dígitos.

Exemplo de execução

```
Digite o tamanho do vetor (n): 6
Digite os caracteres do vetor (separados por espaço ou enter):
a
b
c
3
4
e

Número de consoantes no vetor: 2
```

66. Elabore uma função com assinatura `void soma_acumulada(int src[], int tgt[], int n)` que realiza a soma acumulada dos n valores de `src` e armazena em `tgt`.

Exemplo de execução

```
Digite o tamanho do vetor (n): 5
Digite os elementos do vetor (separados por espaço ou enter):
1
0
4
3
2

Resultado da soma acumulada:
1 1 5 8 10
```

67. Elabore uma função com assinatura `int min(int v[], int n, int *idxmin)` que retorna o menor valor de `v`. Adicionalmente, seu programa de retornar, por meio do ponteiro `idxmin`, o índice correspondente ao menor elemento.

Exemplo de execução

```
Digite o tamanho do vetor (n): 5
Digite os elementos do vetor (separados por espaço ou enter):
0
-2
1
5
8

Menor valor do vetor: -2
Índice correspondente: 1
```

68. Elabore uma função com assinatura `void uniao(int src1[], int src2[], int tgt[], int n)` que realiza a união dos vetores de n elementos `src1` e `src2` e armazena em `tgt`.

Exemplo de execução

```
Digite o tamanho dos vetores (n): 4
Digite os elementos do vetor src1 (separados por espaço ou enter):
-1
0
3
2
Digite os elementos do vetor src2 (separados por espaço ou enter):
3
4
0
-1

Resultado da união dos vetores src1 e src2:
-1 0 3 2 4
```

69. Elabore uma função com assinatura `void largest_sequence_ones(int v[], int n, int * pos, int *size)` que retorna, por meio dos ponteiros `pos` e `size`, respectivamente, o índice inicial da maior sequência de 1´s do vetor `v` (formado apenas por 0´s e 1´s) e o número de 1´s dessa sequência.

Exemplo de execução

```
Digite o tamanho do vetor (n): 10
Digite os elementos do vetor (0 ou 1, separados por espaço ou enter):
0 0 1 0 1 1 1 0 0 0

Número de 1's consecutivos mais longos: 4 (inicia na posição 2)
Tamanho do segmento: 3
```

70. Escreva uma função com assinatura `void unique(int v1[], int v2[], int n)` que copia – preservando a ordem – os elementos de `v1` para `v2` sem repetição. As posições não preenchidas de `v2` devem assinaladas com 0.

Exemplo de execução

```
Digite o tamanho do vetor (n): 10
Digite os elementos do vetor (separados por espaço ou enter):
0 1 1 2 1 1 1 3 5 2

Resultado após copiar elementos únicos para v2 e preencher com 0:
0 1 2 3 5 0 0 0 0 0
```

71. Elabore uma função com assinatura `void shift_zeros(int v[], int n, int *idxmin)` que transfere para o final os 0´s do vetor `v`. Essa operação deve ser executada "inplace", ou seja, o resultado é armazenado no próprio vetor. Além disso, não deve ser criado um vetor auxiliar.

Exemplo de execução

```
Digite o tamanho do vetor (n): 7
Digite os elementos do vetor (separados por espaço):
0 1 3 -1 0 0 5

Resultado após transferir os 0's para o final do vetor:
1 3 -1 5 0 0 0
```

72. Elabore uma função com assinatura `int match(int v1[], int v2[], int n)`. Assuma que `v1` e `v2` são vetores formados apenas por 0's e 1's, sendo o tamanho de `v2` estritamente menor do que o de `v1`. Tal função deve verificar se `v2` é uma subsequência de `v1` e retornar o índice do vetor `v1` onde foi verificado o casamento dos padrões ou -1 se não houver casamento.

Exemplo de execução

```
Vetor v1: 1 0 1 0 0 1 1
Vetor v2: 0 1 0

Resultado: Padrão encontrado em v1 a partir do índice 2.
```

6 Matrizes

73. Faça um programa que leia uma matriz $m \times m$ (m é inteiro lido do teclado), preencha com 1's a diagonal principal e com 0's os demais elementos (matriz identidade). A matriz deve ser impressa na tela ao final da execução.

74. Faça uma função que receba duas matrizes $m \times m$ (m é inteiro lido do teclado) e informe se elas são iguais.

75. Escreva uma função que recebe uma matriz do tipo `float` e retorne (via parâmetro) a matriz normalizada resultante da divisão da matriz original pelo seu maior valor.

76. Elabore uma função com assinatura `int busca(float M[MAX][MAX], int n_lin, int n_col , float v, int *lin, int *col)` que retorna nas variáveis *lin* e *col*, respectivamente, a linha e coluna em que se encontra do valor `v` ma matriz (caso aja algum). Em não havendo, a função deve retornar -1 ao final, caso contrário, retornar qualquer valor diferente de -1. No caso de valores repetidos, sua função deve retornar o primeiro encontrado considerando o percorrimento de linhas e colunas no sentido *raster*.

77. Faça uma função em C que receba uma matriz 3×3, calcule seu determinante e ao final mostre na tela o valor resultante.

```c
int calcular_determinante(int matriz[3][3]) {
    // Sua implementação aqui
}

int main() {
    int matriz[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int determinante = calcular_determinante(matriz);
    printf("O determinante da matriz é: %d\n", determinante);
    return 0;
}
```

Exemplo de Execução

```
Entrada:
1 2 3
4 5 6
7 8 9

Saída:
O determinante da matriz é: 0
```

78. Elabore um programa em C com uma função com assinatura `int busca(float M[MAX][MAX], int n_lin, int n_col, float v, int *lin, int *col)` que retorna nas variáveis `lin` e `col`, respectivamente, a linha e coluna em que se encontra o valor `v` na matriz (caso haja algum). Em não havendo, a função deve retornar -1 ao final. No caso de valores repetidos, sua função deve retornar o primeiro encontrado considerando o percorrimento de linhas e colunas no sentido *raster*.

Exemplo de Execução

```
Digite os valores da matriz:
1.0 2.0 3.0
4.0 5.0 6.0
7.0 8.0 9.0

Digite o valor a ser buscado: 5.0
O valor 5.0 foi encontrado na linha 1 e coluna 1.
```

79. Implemente uma função de assinatura `void lineariza(int M[MAX][MAX], int n_lin, int_col , int v[MAX*MAX])` que tem por objetivo copiar os valores de uma matriz `M` para um vetor `v`. Teste sua função com um programa que lê as dimensões e valores da matriz a ser analisada, imprimindo o vetor `v`.

Exemplo de execução

```
Digite o número de linhas da matriz: 3
Digite o número de colunas da matriz: 4
Digite os elementos da matriz:
5 8 12 7
2 4 9 1
11 6 3 10

Vetor resultante após linearização da matriz:
5 8 12 7 2 4 9 1 11 6 3 10
```

80. Implemente uma função de assinatura `void altera_forma(int M1[MAX][MAX], int n_lin1, int_col1, int M2[MAX][MAX], int int n_lin2, int n_col2)` que tem por objetivo copiar os valores de uma matriz `M1` para uma matriz `M2`. Contudo, `M2` terá n_lin2 linhas e n_col2 colunas. Teste sua função com um programa que lê as dimensões e valores da matriz a ser analisada, imprimindo `M1` e `M2` ao final.

81. Dada duas matrizes $A_{m \times n}$ e $B_{n \times o}$, faça um programa que mostre a matriz $P = A \cdot B$.

82. Na teoria dos sistemas, define-se como elemento minimax de uma matriz o menor elemento da linha onde se encontra o maior elemento da matriz. Sabendo disso, escreva um programa que lê uma matriz inteira $n_{lin} \times n_{col}$ e a escreve na tela, e que escreve na tela o elemento minimax e sua posição (linha e coluna) na matriz.

Exemplo de execução

```
Digite o número de linhas da matriz: 3
Digite o número de colunas da matriz: 4
Digite os elementos da matriz:
5 8 12 7
2 4 9 1
11 6 3 10
```

```
Matriz:
5 8 12 7
2 4 9 1
11 6 3 10

O elemento minimax é 5
Ele está localizado na linha 0 e coluna 0
```

83. Considere a Tabela 1 que relaciona distâncias entre 5 cidades capixabas. A tabela em questão pode ser modelada por meio de uma matriz 5×5. Considere o seguinte circuito (rota que passa por todas as cidades): Vitória (Posição 0) → Serra (Posição 3) → Cariacica (Posição 2) → Vila Velha (Posição 1) → Guarapari (Posição 4) → Vitória (Posição 5). Faça um programa que:

    - Declare uma matriz com os valores da referida tabela;
    - Leia um circuito como um vetor de índices;
    - Imprima a distância total percorrida.

Tabela 1: Distâncias entre Cidades Capixabas

Cidade	Distância (em km)				
	Vitória	Vila Velha	Cariacica	Serra	Guarapari
Vitória	-	10	7	15	50
Vila Velha	10	-	15	8	45
Cariacica	7	15	-	20	55
Serra	15	8	20	-	40
Guarapari	50	45	55	40	-

Exemplo de execução

```
Digite o circuito como um vetor de índices: 0 3 2 1 4 0
Circuito: Vitória -> Serra -> Cariacica -> Vila Velha -> Guarapari -> Vitória
Distância total percorrida: 160 km
```

7 Alocação dinâmica

84. Escreva uma função com assinatura `int* aloca_vetor(int n)` em linguagem C, que aloque dinamicamente um vetor de inteiros com `n` elementos e retorne um ponteiro para esse vetor. Certifique-se de que você aloque memória para o vetor corretamente e lide com possíveis erros de alocação.

Exemplo de execução

```
Digite o tamanho do vetor: 5
Digite os 5 elementos do vetor:
10 20 30 40 50

Vetor alocado dinamicamente:
10 20 30 40 50
```

85. Escreva uma função em linguagem C chamada `void libera_vetor(int *v, int n)` que liberar a memória alocada para um vetor `v` de tamanho `n`.

86. Escreva uma função em linguagem C chamada `float** aloca_matriz(int lin, int col)` que aloque dinamicamente uma matriz com elementos do tipo `float` com `lin` linhas e `col` colunas. A função deve retornar um ponteiro para a matriz alocada.

87. Suponha que você está desenvolvendo um programa em C que lida com processamento de dados em matrizes de floats. Você precisa alocar uma matriz de floats dinamicamente para armazenar esses dados e, posteriormente, liberar a memória alocada para evitar vazamentos de memória.

    (a) Explique a utilidade de alocar dinamicamente a matriz em vez de usar uma matriz estática de tamanho fixo.

    (b) Descreva como você usaria no programa principal a função `aloca_matriz` que você implementou para alocar memória para a matriz. Qual é o papel dos argumentos `lin` e `col` na função?

    (c) Ao trabalhar com matrizes alocadas dinamicamente, é importante garantir que a memória seja adequadamente liberada para evitar vazamentos de memória. Explique como você usaria a função `libera_matriz` que você implementou para desalocar a memória da matriz.

    (d) O que aconteceria se você esquecesse de chamar a função `libera_matriz` após o término do uso da matriz alocada dinamicamente? Quais são as possíveis consequências disso?

88. Implemente uma função que receba como parâmetros dois vetores de inteiros `x1` e `x2` e as suas respectivas quantidades de elementos `n1` e `n2`. A função deverá retornar um ponteiro para um terceiro vetor, `x3`, alocado dinamicamente, contendo a união de `x1` e `x2` e usar o ponteiro `n3` para retornar o tamanho de `x3`. Adote como assinatura da função `int* uniao(int *x1, int *x2, int n1, int n2, int* n3)`.

    Exemplo de execução

    ```
    Vetor x1: 1 3 5 6 7
    Vetor x2: 1 3 4 6 8

    União dos vetores:
    Vetor x3: 1 3 4 5 6 7 8
    Tamanho de x3: 7
    ```

89. Crie uma função que receba como parâmetros dois vetores de inteiros `x1` e `x2` e as suas respectivas quantidades de elementos `n1` e `n2`. A função deverá retornar um ponteiro para um terceiro vetor, `x3`, alocado dinamicamente, contendo a intersecção de `x1` e `x2` e usar o ponteiro `n3` para retornar o tamanho de `x3`. Adote como assinatura da função `int* interseccao(int *x1, int *x2, int n1, int n2, int *n3)`.

    Exemplo de execução

    ```
    Vetor x1: 1 3 5 6 7
    Vetor x2: 1 3 4 6 8

    Intersecção dos vetores:
    Vetor x3: 1 3 6
    Tamanho de x3: 3
    ```

8 Strings

Observação: Pesquise sobre as funções `strlen`, `strcat`, `strcpy`, `strstr` e `strcmp` para resolver os exercícios propostos.

90. Faça um programa que leia uma string e imprima seus caracteres na ordem reversa.

91. Faça um programa que conte quantas letras maiúsculas existem numa string lida do teclado e mostre o resultado da contagem.
Dica: O código ASC para letras maiúsculas está no intervalo $[65, 90]$ e para as letras minúsculas no intervalo de $[97, 122]$.

92. ⋆ Escreva uma programa que troque todas as letras maiúsculas por minúsculas e as minúsculas por maiúsculas de uma string recebida como entrada. Mostre na tela a string resultante.
Dica: A diferença entre as letras minúsculas e maiúsculas na tabela ASCII é sempre 32. Portanto, você pode usar essa diferença para realizar a conversão manualmente.

93. Faça um programa que leia 2 strings (`s1` e `s2`) e gere uma terceira string formada pelos caracteres de `s1` e `s2` intercalados.
Exemplo: Se `s1 = "Quarta"` e `s2 = "Segunda"`, a resposta deve ser `"QSueagrutnada"`.

94. Escreva uma função que receba uma string `s`, um caractere `c` e retorne o índice da primeira ocorrência de `c` em `s`. Caso não haja o caractere, retornar -1.
Sugestão de assinatura de função: `int buscar_caractere(char [], char c);`.

95. ⋆ Faça um programa que lê duas strings (`s1` e `s2`) e verifica se `s2` é uma substring de `s1`.

96. Desenvolva um programa em C capaz de substituir todas as ocorrências de uma substring `sub1` em uma string `s` por outra substring `sub2`, e exibir a nova string resultante.
Exemplo: Considere a string original: "A casa amarela é bonita." Queremos substituir todas as ocorrências da substring "amarela" por "vermelha". A nova string resultante seria: "A casa vermelha é bonita.".

97. Implemente sua própria versão das seguintes funções: `strcpy`, `strcmp`, `strlen` e `strcat`.

98. Você foi contratado para projetar um Mini-Google. Seu programa deve receber uma string `s` e um valor inteiro `n` como entrada. Em seguida, o programa deve ler `n` strings adicionais e imprimir apenas aquelas para as quais `s` é uma substring.
Dica: utilize a função `strstr`.

99. Um caractere pode ser convertido em um inteiro utilizando o *typecasting* (ex. `int x = (int)c`, onde `c` é uma variável do tipo `char`). Sabendo disso, implemente uma função de assinatura `int string_para_inteiro(char str[])` que recebe uma string contendo apenas dígitos e retorna um número inteiro correspondente à sequência de dígitos dessa string.

100. Leia uma cadeia de caracteres no formato "DD/MM/AAAA" (`char data[11]`) e implemente uma função que receba essa cadeia de caracteres como entrada e copie o dia, mês e ano para três variáveis inteiras separadas. Assuma que o usuário digitará a data corretamente. Assinatura sugerida da função: `void extrair_data(char data[], int* dia, int* mes, int* ano);`. A função `extrair_data` recebe a cadeia de caracteres `data` contendo a data no formato especificado e atualiza as variáveis `dia`, `mes` e `ano` com os valores correspondentes no formato inteiro.

101. Uma string é utilizada para representar uma das fitas de uma cadeia de DNA. Para tanto, as bases Adenina, Guanina, Citosina, Timina e Uracila são representadas pelas letras 'A', 'G', 'C', 'T' e 'U', respectivamente. Deseja-se construir um programa que, dada uma sequência de DNA, forneça a sequência de RNA-m equivalente de acordo com a transformação indicada na Tabela 2.

Tabela 2: Correspondência entre as bases nitrogenadas do DNA e do RNA-m

Base nitrogenada no DNA	Base correspondente no RNA-m
A	U
G	C
C	G
T	A

102. Um operador de crossover pode ser aplicado a duas strings `s1` e `s2` que representam sequências de DNA. Esse operador consiste em sortear aleatoriamente um ponto em `s1` e `s2` (mesmo índice) e, em seguida, realizar a troca de informações entre `s1` e `s2`, como ilustrado na Figura 2.

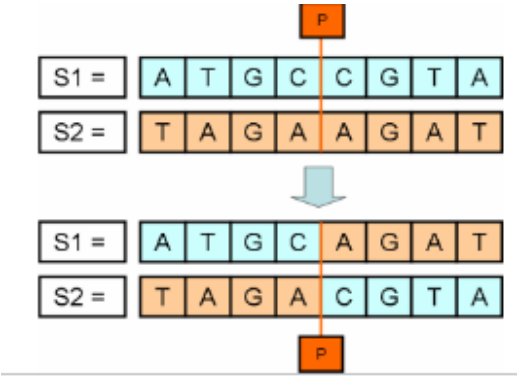

Figura 2: Esquema ilustrando o operador de crossover entre as strings `s1` e `s2`.

Construa um programa que, dado as strings `s1` e `s2` formadas apenas pelos caracteres 'C', 'T', 'A' e 'G', e o ponto `p` a partir do qual as trocas serão realizadas, empregue o operador de crossover para construir novas strings `s1_crossover` e `s2_crossover`.

As novas strings `s1_crossover` e `s2_crossover` devem ser formadas pela troca de informações entre `s1` e `s2` a partir do índice `p`. Em outras palavras, as informações de `s1` a partir do índice `p` serão substituídas pelas informações de `s2` a partir do mesmo índice, e vice-versa.

Após realizar as trocas, mostre as strings resultantes `s1_crossover` e `s2_crossover`, juntamente com o valor do índice `p`.

9 Arquivos

103. Faça um programa que crie um arquivo **texto** em disco com o nome `dados.txt`, e escreva neste arquivo em disco uma contagem que vá de 1 até 100, com um número em cada linha. Abra este arquivo em um editor de textos.

104. Faça um programa que crie um arquivo **binário** em disco com o nome `dados.bin`, e escreva neste arquivo em disco uma contagem que vá de 1 até 100, com um número em cada linha. Abra este arquivo em um editor de textos e observe como ficou o seu conteúdo (ilegível!).

105. Faça 2 programas, um que leia o arquivo texto criado no primeiro exercício e outro que leia o arquivo binário criado no segundo. Exibir na tela os dados lidos dos respectivos arquivos.

106. Neste exercício, você criará um programa em C para criptografar e descriptografar o conteúdo de um arquivo usando um algoritmo de cifra simples.

Parte 1: Criptografia Crie um programa em C chamado `criptografia.c` que abra um arquivo de entrada chamado `entrada.txt` para leitura. Leia cada caractere do arquivo de entrada e aplique a seguinte função de criptografia:

$$C(P, K) = (P + K) \mod 26,$$

onde P é o caractere de texto original (letra do alfabeto) e K é a chave de deslocamento (um número inteiro). Grave o conteúdo criptografado em um arquivo de saída chamado `criptografado.txt`.

Parte 2: Decriptografia Faça um programa que leia o arquivo `criptografado.txt` e decodifique usando a função

$$D(C, K) = (C - K) \mod 26.$$

Utilize a mesma chave empregada na criptografia. Mostre o conteúdo decriptografado na tela e compare-o com o arquivo `entrada.txt`.

107. Considere um arquivo texto com as notas dos alunos de uma disciplina. Cada linha do arquivo contém a matrícula de um aluno (cadeia de nove caracteres), seguida pelos valores de suas três notas (P_1, P_2 e P_3). Considere ainda que podem existir linhas em branco no arquivo. Um exemplo desse formato é:

```
9010087-2       2.0     4.3     6.5
8820324-3       7.0     8.2     8.6
9210478-5       6.0     7.5     7.8
9020256-8       3.0     0.5     4.2
```

Escreva uma função que receba como parâmetros o número de matrícula de um aluno e o nome de um arquivo com as notas de uma disciplina no formato descrito, e retorne a média do aluno na disciplina. A média de um aluno é calculada pela fórmula $(P_1 + P_2 + P_3)/3$.

Caso o número de matrícula passado como parâmetro não seja encontrado no arquivo, a função deve retornar -1. Se não for possível abrir o arquivo de entrada, a função deve imprimir a mensagem "Erro" e terminar a execução do programa.

108. Considere a existência de um arquivo texto, denominado `turma.txt`, com um cadastro de uma turma. Para cada aluno da turma, existem duas linhas no arquivo: na primeira linha, encontra-se o nome do aluno; na linha seguinte, encontram-se as duas notas obtidas pelo aluno. Considere que podem existir linhas em branco no arquivo. Um exemplo deste arquivo é mostrado a seguir:

```
Fulano Pereira
9.0 8.0
Beltrano Silva
4.0 5.0

Sicrano Santos
3.0 7.0
Fulana Souza
4.0 4.0

Maria Paula
6.0 7.0
```

Escreva um programa completo que leia o conteúdo do arquivo `turma.txt` com o formato anterior e crie outro arquivo, com o nome `aprovados.txt`, com a lista dos alunos que obtiveram médias

maiores ou iguais a 5.0, seguindo a mesma ordem do arquivo de entrada. Cada linha do arquivo de saída deve conter o nome e a média obtida pelo aluno. Se esse arquivo anterior fosse usado num exemplo, a saída obtida seria:

```
Fulano Pereira 8.5
Sicrano Santos 5.0
Maria Paula 6.5
```

Se o arquivo de entrada não puder ser aberto, deve-se imprimir a mensagem "Erro" na tela e abortar o programa. Pode-se considerar que sempre será possível abrir o arquivo de saída.

109. Considere um tipo que representa um funcionário de uma empresa, definido pela estrutura a seguir:

```c
// Define o tipo Funcionario
typedef struct funcionario Funcionario;
struct funcionario {
   char nome[81]; // nome do funcionário
   float valor_hora; // valor da hora de trabalho em Reais
   int horas_mes; // horas trabalhadas em um mês
};
```

Escreva uma função em C que preencha um vetor de ponteiros para Funcionario com os dados lidos de um arquivo texto. Essa função deve receber como parâmetros o vetor de ponteiros para Funcionario e o nome do arquivo de entrada. Nesse arquivo de entrada, os dados de cada funcionário são armazenados em duas linhas: uma com o seu nome (cadeia com até 80 caracteres), e outra com o valor de sua hora de trabalho e com o número de horas trabalhadas em um mês (nessa ordem). Um exemplo desse formato é mostrado a seguir.

```
João da Silva
15.0 160
Manuel Santos
15.0 80
Fulana de Tal
23.5 40
```

10 Tipo abstrato de dados

110. Implemente um TAD em C que representa um círculo contendo as informações de seu raio e centro no plano cartesiano. Para isso, implemente a estrutura `circle` e mais duas funções (Código 1): uma para inicializar a estrutura e outra para verificar se há interseção entre dois círculos.

Observação: Dois círculos se intersectam se a distância entre os seus centros é menor ou igual à soma dos seus raios.

Código 1: circle.h.

```c
// Definição do tipo
typedef struct circle Circle;

// Inicializa círculo alocando espaço e inicializando seus campos internos.
Circle* init_circle(float x, float y, float radius);

// Verifica interseção entre dois círculos (1 - sim, 0 - não)
int intersect(Circle* c1, Circle* c2);
```

Código 2: circle.c.

```c
#include <circle.h>
#include <math.h>

// Implementação do tipo
struct circle {
/**
 * Seu código aqui.
 */
};

// Inicializa círculo alocando espaço e inicializando seus campos internos
Circle* init_circle(float x, float y, float radius) {
/**
 * Seu código aqui.
 */
}

// Verifica interseção entre dois círculos (1 - sim, 0 - não)
int intersect(Circle* c1, Circle* c2) {
/**
 * Seu código aqui.
 */
}
```

11 Vetores dinâmicos

111. Diferencie tamanho e capacidade de um vetor dinâmico.

112. Implemente a função `void dv_write_to_file(DynVec *dv, const char *filename)` que grava o conteúdo do vetor `dv` em um arquivo texto de nome `filename`.

113. Aponte a falha e no código a seguir justificando sua resposta. O objetivo da função em questão é liberar o espaço de memória ocupado pelo vetor dinâmico `dv`.

```c
// Function to free the memory associated with the dynamic vector
void dv_free(DynVec *dv)
{
  free(dv); // Free the ADT
  free(dv->v); // Free the data vector
}
```

114. Implemente as seguintes funções:

   (a) `DynVec *dv_concatenate(DynVec *dv1, DynVec *dv2)`: Concatena `dv1` e `dv2` em um novo vetor dinâmico e retorna um ponteiro para esse novo vetor.

   (b) `DynVec *dv_union(DynVec *dv1, DynVec *dv2)`: Retorna um vetor com a união dos elementos de `dv1` e `dv2`.

   (c) `DynVec *dv_intersection(DynVec *dv1, DynVec *dv2)`: Retorna um vetor com a interseção dos elementos de `v1` e `v2`.

12 Listas simplesmente encadeadas

115. Implemente as seguintes funções:

    (a) `LinkedList *ll_reversed(LinkedList *l))`: Retorna uma cópia da lista `l` com os elementos na ordem inversa.

    (b) `ll_append(LinkedList *l, int v)`: Insere um elemento com valor `v` ao final da lista `l`.

    (c) `void ll_insert_sorted(LinkedList *l, int v)`: Insere um elemento `v` na lista `l` preservando a ordenação (ordem crescente).

    (d) `int ll_is_sorted(LinkedList *l)`: Verifica se a lista `l` está ordenada (ordem crescente).

    (e) `LinkedList *ll_concatenate(LinkedList *l1, LinkedList *l2)`: Retorna uma nova lista resultante da concatenação da lista `l1` com a lista `l2`.