Lista de Exercícios 1 de Introdução a Programação

Professor: Marcos Costa
Monitores: Felipe e Marcio

Orientações a serem seguidas:

a) a lista é individual e deverá ser entregue e apresentada em data a ser informada posteriormente;
b) a entrada dos dados e a saída dos programas será feita utilizando arquivos – a questão exemplo mostra como fazer;
c) tire suas as duvidas da lista preferencialmente com os monitores;
d) faça a lista progressivamente, dia após dia (não deixe para a "última hora");
e) as entradas e as saídas são padronizadas – observe atentamente a padronização nas questões;
f) não valide as entradas, exceto quando for solicitado na questão;
g) os arquivos de entrada mostrados nas questões são apenas exemplos – de maneira que, no momento de realizar testes, poderão ser utilizados outros arquivos, seguindo sempre a formatação indicada;
h) a entrega de questões fora dos padrões implica na perda parcial ou total da mesma;
i) deverão ser entregues (em CD ou pen-drive) apenas os fontes (arquivos com extensão .c) utilizando o seguinte padrão:
– Será criada, dentro da pasta raiz L1C, uma pasta para cada questão (q1, q2, q3, etc.);
– Dentro de cada pasta devera estar o arquivo correspondente a resposta da questão (q1.c, q2.c, q3.c, etc.).
j) Os arquivos de entrada e saída de cada questão, bem como os executáveis não devem ser enviados;
k) na primeira linha de cada código fonte dos programas fonte devemos ter um comentário de linha contendo o nome do autor da questão;
l) A utilização de entrega fora dos padrões estabelecidos implicarão em perda de pontos ou ainda da questão inteira, dependendo do caso;
m) os códigos fonte não devem ter system("PAUSE").
n) As questões de números 1 a 5 devem ser respondidas apenas com a função *main()*.

Questão exemplo: Escreva um programa que lê um numero inteiro e imprime o numero lido multiplicado por dois.

2

Exemplo de arquivo texto de entrada da questão (de nome e.txt), contendo apenas o número a ser lido – deve ser colocado, no projeto, no mesmo diretório do arquivo fonte .c

```
#include <stdio.h>
int main(void) {
int a;
/**
* A função freopen redireciona a entrada padrão (por default, o teclado), para o arquivo "e.txt".
* O "r" do segundo parâmetro significa que o arquivo "e.txt" será utilizado para leitura.
*/
freopen("e.txt", "r", stdin);
/**
* A função freopen redireciona a saída padrão (por default, o monitor), para o arquivo "s.txt".
* O "w" do segundo parâmetro significa que o arquivo "s.txt" será utilizados para a escrita.
*/
freopen("s.txt", "w", stdout);
//observe que, o scanf e o printf são utilizados normalmente.
scanf("%d", &a);
printf("%d", 2*a);
/**
* A função fclose é utilizada para fechar os arquivos abertos, que não
* vão mais ser utilizados.
*/
fclose(stdin);
fclose(stdout);
return 0;
}
```

4

Exemplo de arquivo de saída "s.txt". Procure observar atentamente como a saída é descrita e os exemplos são
mostrados.
Erros comuns de saída:
- adicionar mensagens não solicitadas (ex. Dobro = 10);
- colocar linha em branco no começo do arquivo quando não solicitado (a solicitação foi de apenas escrever um número);
- colocar tabulação quando não solicitado, etc.

1. Viajar de automóvel é relativamente rápido, mas envolve custos. Um deles é o do combustível, o outro é o pedágio. Sua tarefa é criar um programa que, dados o comprimento da pista, o custo do combustível por quilômetro rodado e os pedágios existentes, calcule o valor gasto pelo motorista ao viajar por toda a sua extensão.

   Observações:
   a) Os pedágios são igualmente espaçados entre si, e no início da pista não há pedágios (embora possa haver um exatamente no seu final, caso em que o motorista também deve pagar).
   b) A primeira linha do arquivo de entrada contém um único inteiro n (n > 0), correspondente ao número de pistas analisado. As n linhas seguintes contêm quatro números, que representam respectivamente: o comprimento da pista; a distância entre pedágios (ambos em km); o custo do combustível em valor por quilômetro; e o valor cobrado em cada pedágio.
   c) A saída deve conter n inteiros em n linhas, correspondentes ao valor gasto pelo motorista ao percorrer cada uma das n estradas.
   d) Utilizar variáveis do tipo *unsigned int*.

3 111 37 1 10 100 30 3 14 20 70 9 17 e1.txt
141 342 180 s1.txt

2. Léo é um corredor profissional que viaja bastante para realizar seus treinamentos. Como as pistas de corrida de cada lugar por onde ele passa nem sempre têm o mesmo tamanho, ele decidiu fixar o número de metros percorridos em vez do número de voltas na pista. Após cada treinamento, Léo precisa beber meio litro de água antes de realizar qualquer outro esforço. Por isso, ele deseja deixar sua garrafa de água exatamente no ponto onde terminará sua corrida. Escreva um programa que informe em que ponto de cada pista Léo deve posicionar sua garrafa de água.

   Observações:
   a) A primeira linha da entrada contém o número n de pistas em que Léo treinará, enquanto os n pares de números nas linhas seguintes correspondem, respectivamente, ao número de metros que Léo pretende percorrer e o comprimento da pista.

   b) Utilize variáveis do tipo unsigned int. Cada linha do arquivo de saída corresponde à posição da garrafa de água de cada pista, em metros após o ponto de partida.

4 7000 100 918 76 1260 92 5040 63 e2.txt
0 6 64 0 s2.txt

3. Sabe-se que o DNA é composto de uma dupla-hélice de cadeias de nucleotídeos, dispostos de forma a combinar suas respectivas bases nitrogenadas da seguinte forma:

A base...	...combina-se com...
T (Timina)	A (Adenina)
A (Adenina)	T (Timina)
C (Citosina)	G (Guanina)
G (Guanina)	C (Citosina)

Sua tarefa é escrever um programa que, dada uma sequência de bases nitrogenadas correspondentes a uma das cadeias de um trecho de DNA, forneça uma segunda sequência que combine com a primeira. A leitura de dados da entrada termina quando for lido o caractere ".".

TTGGGATCTACACGTCTTACACGA. e3.txt
AACCCTAGATGTGCAGAATGTGCT s3.txt

4. Os padrões de resolução de telas de computadores pessoais são geralmente 4:3 ou 16:9. Escreva um programa que lê um numero inteiro positivo e um caractere, no qual o caractere informa se o numero lido é a largura (W) ou a altura (H) da resolução de vídeo. Na saída deve ser exibida a resolução completa nos dois formatos, 4:3 e 16:9.

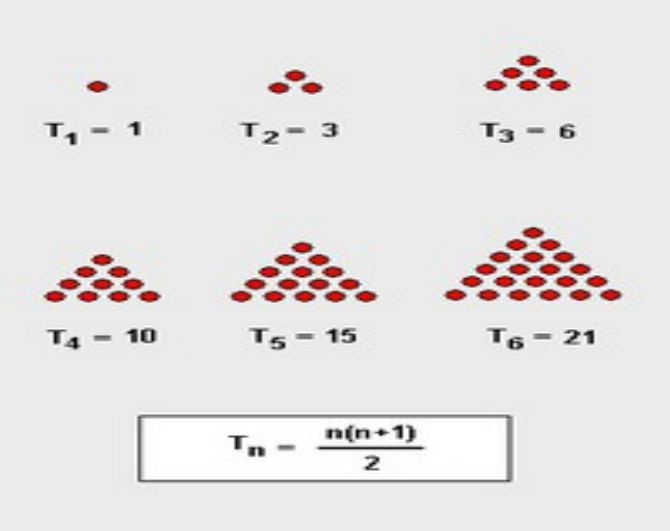
720 H e4.txt
4:3 = 960x720 16:9 = 1280x720 s4.txt

5. Escreva um programa que lê uma série de números inteiros positivos, e imprime todos os seus múltiplos ate *n*, onde *n* é o primeiro número do arquivo de entrada. O programa deve parar quando ler o número 0. Observação: o número *n* é o maior que todos os outros e os múltiplos calculados deverão ser menores que *n*.

100 5 4 10 32 21 0 e5.txt
5: 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95 4: 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 64, 68, 72, 76, 80, 84, 88, 92, 96 10: 20, 30, 40, 50, 60, 70, 80, 90 32: 64, 96 21: 42, 63, 84 s5.txt

6. Um número triangular é um número natural que pode ser representado na forma de triângulo equilátero. Foi desenvolvido por Gauss em 1788 quando ele tinha somente 10 anos. Para encontrar o *n-ésimo* número triangular a partir do anterior basta somar-lhe *n* unidades. Os primeiros números triangulares são: 1, 3, 6, 10, 15, 21, 28, 36, 45, 55...

   Em geral, o *n-ésimo* número triangular é dado por:



Escreva um programa que lê n números inteiros positivos, calcula T_n pra cada número lido e informa se T_n é um número perfeito. A leitura deve terminar quando for lido o número 0.

Na Matemática, um número perfeito é um número inteiro para o qual a soma de todos os seus divisores positivos próprios (excluindo ele mesmo) é igual ao próprio número. Por exemplo, 6 é um número perfeito, pois 6 = 1 + 2 + 3.

2 6 3 31 11 0 e6.txt
T2=3 imperfeito T6=21 imperfeito T3=6 perfeito T31=496 perfeito T11=66 imperfeito s6.txt

7. Escreva um programa que calcule o número PI utilizando a série de *David Bailey*. O programa deve ler um número inteiro positivo *n*, que representa os *n+1* termos da série a ser utilizada nos cálculos (*k* deverá ir, no somatório informado na fórmula abaixo, de 0 até n, intervalo fechado). Além disso, também deve ser calculado o erro relativo do valor PI calculado em função do valor da constante M_PI da biblioteca *math.h*. Para calcular o número PI, utilize uma função com o seguinte protótipo: *double calculaPI(int n)*. O número PI calculado e o erro relativo devem ser impressos com 15 casas decimais.

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right)$$

5 e7.txt
PI: 3.141592653228088 Erro: 0.000000000115134 s7.txt

8. Escreva um programa que lê uma série de pares de números inteiros positivos (base e expoente, respectivamente) e calcula base^{expoente}. Para calcular o exponencial, escreva duas funções de protótipos *int multiplicacao(int, int)* e *int exponenciacao(int base, int expoente)*. A função de exponenciação deve invocar a função multiplicacao em vez de utilizar o operador "*" (a exponenciação deve ser calculada através de multiplicações). A função de multiplicação deve utilizar o operador "+" (nunca o operador "*") para calcular o produto de dois números. A leitura termina quando for lida uma linha com 2 números iguais a 0.

2 5 3 4 7 1 9 3 0 0 e8.txt
32 81 7 729 s8.txt

9. Escreva um programa que, dado um número inteiro positivo *m* (1 < *m* < 21), imprima o triângulo de Pascal com *m* linhas.
   a) Cada termo do triângulo é definido pela seguinte fórmula:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

   Onde *n* é a posição da linha, e *k* a posição da coluna no triângulo, ambos contados a partir da posição zero (*n* = 0, *k* = 0).

   b) Crie a função *double fatorial (unsigned int)*, não recursiva, que deve retornar o fatorial do número a.

5 e9.txt
1 1 1 1 2 1 1 3 3 1 1 4 6 4 1 s9.txt

10. Pitágoras descobriu que é possível calcular o quadrado de um número inteiro positivo *n* somando os *n* primeiros números ímpares positivos. Por exemplo, $5^2 = 1+3+5+7+9$. Escreva um programa que, dado um número, descubra se ele é um quadrado perfeito e em caso afirmativo, imprima-o como uma soma de números ímpares. Caso contrário, o programa deve imprimir a mensagem "imperfeito". O programa deve parar de executar caso leia um número menor ou igual a zero.

25 10 9 36 64 0 e10.txt
25=1+3+5+7+9 imperfeito 9=1+3+5 36=1+3+5+7+9+11 64=1+3+5+7+9+11+13+15 s10.txt

11. Jack e Daniel são dois dos melhores jogadores de Wyrt do Velho Oeste. Wyrt é um jogo de cartas e dados que consiste em cada jogador puxar uma carta de um baralho comum de 52 cartas e jogar um dado de seis faces. Eles sempre se reúnem ao pôr do sol na taverna para jogar enquanto bebem uísque, mas sempre têm problemas para decidir quem venceu – no fim da noite, eles não estão sóbrios o suficiente para isso. Então, os dois jogadores decidiram pedir sua ajuda para anunciar o vencedor de cada noite e, como o maior inventor dessas bandas, você não tem tempo para isso e decidiu escrever um programa que, dadas as jogadas de cada um durante as partidas, anuncie automaticamente o vencedor.

   a) Wyrt é jogado usando um baralho comum de 52 cartas e um dado de seis faces. Cada carta possui um valor que vai de 2 a 11, sendo que as cartas "figuradas" (A, K, Q, J) valem 11 pontos. Os naipes não influenciam na pontuação. O jogo consiste em cada jogador puxar três cartas do baralho e, alternadamente, cada um jogar uma carta e o dado. Se o resultado do dado for par, a soma dos pontos do dado e da carta se soma aos seus próprios. Caso o resultado seja ímpar, a soma dos dois é subtraída. Cada jogador começa com zero ponto e a pontuação pode também resultar em números negativos.

   b) A primeira linha do programa é um inteiro n, o número de partidas jogadas pelos dois numa

determinada noite. Cada um dos n conjuntos de três linhas contêm dois pares de inteiros a1, b1, a2, b2 (2 <= a1, a2 <= 11, 1 <= b1, b2 <= 6), representando as jogadas de cada um, e são espaçados por uma linha em branco. A primeira jogada é sempre de Jack.

c) Seu programa deve imprimir o nome do vencedor de cada partida. Caso os pontos de ambos sejam iguais, o programa deve imprimir "empate".

2 10 2 5 1 3 3 6 2 10 1 5 4 11 6 10 2 3 4 10 5 6 2 7 4 e11.txt
Daniel Jack s11.txt

12. Escreva um programa que informa qual dia da semana corresponde a uma determinada data. Observações:
a) O primeiro número n lido na entrada, que varia de 1 a 7, corresponde ao primeiro dia do ano. Considere que n representa um domingo se for igual a 1, segunda-feira se igual a 2, e assim por diante até o sábado, cujo número correspondente é 7.

b) Os próximos pares de números correspondem às datas, sendo o primeiro número o dia e o segundo o mês. Para cada data, o programa deve imprimir o dia da semana correspondente. O programa deve parar de executar ao ler um par de zeros.

c) Crie ao menos duas funções com os seguintes protótipos: unsigned int calculaDia (unsigned int dia, unsigned int mes) e void imprimeDia (unsigned int diaDaSemana). Use a primeira para "calcular" o dia da semana correspondente à data e a segunda para imprimi-lo.

d) Desconsidere anos bissextos.

4 29 3 1 1 31 12 30 3 0 0 e12.txt
sabado quarta feira quarta feira domingo s12.txt

13. Escreva um programa que imprima a soma todos os elementos de um vetor inteiro de 10 posições – o vetor deve conter números randômicos entre 1 e 100. O programa lê, como entrada, o número x de vezes que devem ser gerados e somados os 10 números randômicos que são armazenados no vetor. Na saída, o programa imprime a soma dos números, seguido pelos elementos do vetor. Observação: a cada execução, os números randômicos gerados devem ser diferentes.

8 e13.txt
569 = 100 + 57 + 11 + 35 + 85 + 91 + 8 + 12 + 91 + 79 532 = 69 + 42 + 43 + 15 + 40 + 71 + 72 + 51 + 64 + 65 322 = 47 + 48 + 6 + 26 + 89 + 26 + 24 + 9 + 22 + 25 456 = 24 + 61 + 5 + 63 + 80 + 78 + 3 + 17 + 75 + 50 512 = 64 + 71 + 78 + 42 + 40 + 69 + 21 + 53 + 58 + 16 440 = 63 + 23 + 71 + 64 + 17 + 1 + 85 + 14 + 13 + 89 493 = 18 + 65 + 34 + 22 + 95 + 37 + 16 + 93 + 80 + 33 492 = 60 + 41 + 85 + 61 + 26 + 31 + 85 + 86 + 8 + 9 s13.txt

14. Escreva um programa que, dado um vetor de 20 posições lido, ordena esse vetor utilizando o algoritmo da bolha (http://pt.wikipedia.org/wiki/Bubble_sort). Crie uma função, de protótipo *void ordena(int v[], int n)*, onde v é o vetor de 20 posições a ser ordenado e *n* é o tamanho do vetor (no caso, 20). Essa função deve ordenar o vetor v, mas não deve imprimir o vetor ordenado, essa tarefa caberá à função main. Observação: a função *ordena* deverá estar em um arquivo de nome *"bolha.h"*, armazenado mesmo diretório do arquivo *"q14.c"*.

1 20 -1 0 3 8 9 -9 0 4 -4 5 12 10 -10 11 -6 10 19 2 e14.txt
-10 -9 -6 -4 -1 0 0 1 2 3 4 5 8 9 10 10 11 12 19 20 s14.txt

15. O máximo divisor comum dos inteiros x e y é o maior inteiro que divide precisamente x e y. Escreva um programa que calcula o mdc de vários pares de números inteiros positivos. Para calcular o mdc, escreva uma função recursiva mdc que retorne o maior divisor comum de x e y. O maior divisor comum de x e y é definido recursivamente como se segue: Se y for igual a 0, então o *mdc(x, y)* é *x*; de outra forma *mdc(x, y)* é *mdc(y, x % y)* onde *%* é o operador resto. A 1ª linha do arquivo informa o número de pares. A seguir, temos os pares, sempre com o primeiro nº maior do que o segundo.

2 30 25 24 20 e15.txt
5 4 s15.txt

16. Escreva um programa que lê um número inteiro maior ou igual e zero, calcula e exibe a série e a soma da série de fibonacci até o número lido. A série de Fibonacci é definida como 0, 1, 1, 2, 3, 5, 8, 13, 21, …. Defina uma função recursiva de protótipo *int fibonnacci(int n)* para calcular o fibonnacci de um número.

20 e16.txt
0+1+1+2+3+5+8+13+21+34+55+89+144+233+377+610+987+1597+2584+4181+6765=17710 s16.txt

17. Escreva uma função recursiva que inverta um texto lido – a função deve imprimir os caracteres em ordem inversa. A primeira linha da entrada contém o nº de textos a serem invertidos. Cada linha a ser invertida deve ser armazenada em um vetor de 30 posições (cada linha conterá no máximo 29 caracteres).

2 escreva uma funcao recursiva para inverter um texto e17.txt
avisrucer oacnuf amu avercse otxet mu retrevni arap s17.txt

18. Considere que as funções *sin* (seno) e *cos* (cosseno) são definidas da seguinte forma para um ângulo x em radianos:

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} \text{ para todo } x$$

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} \quad \text{para todo } x$$

Considere ainda que a função *tan* (tangente) é o resultado da divisão do seno pelo cosseno.

Escreva um programa que lê vários números reais, que representam ângulos entre 0o e 360o, calcula e imprime o seno, cosseno e tangente do ângulo lido. Importante observar que, para cada ângulo em graus lido, o programa lê ainda um número inteiro positivo n que representa o ponto de parada dos somatórios que devem ser realizados para se calcular o seno e o cosseno (se fossemos somar até infinito, os cálculos não terminariam).

Observações:

- para converter graus em radianos, utilize a seguinte fórmula: radianos = graus * M_PI / 180.0, onde M_PI é uma constante definida na biblioteca *math.h*;
- o programa não deve imprimir o valor da tangente quando o ângulo for 90º ou 270º;
- para calcular o seno, escreva o código da função de protótipo *double seno(double x, int precisao)*;
- para calcular o cosseno, escreva o código da função de protótipo *double cosseno(double x, int precisao)*;
- os resultados dos cálculos devem ser impressos com 4 casas decimais;
- os ângulos em graus devem ser impressos com 2 casas decimais;
- o programa para de executar quando for lido um ângulo negativo.

30.0 6 45.0 3 45.0 8 90.0 2 90.0 7 60.0 8 60.5 8 -1.0 0 e18.txt
sen(30.00): 0.5000, cos(30.00): 0.8660, tan(30.00): 0.5774 sen(45.00): 0.7071, cos(45.00): 0.7074, tan(45.00): 0.9996 sen(45.00): 0.7071, cos(45.00): 0.7071, tan(45.00): 1.0000 sen(90.00): 0.9248, cos(90.00): -0.2337 sen(90.00): 1.0000, cos(90.00): 0.0000 sen(60.00): 0.8660, cos(60.00): 0.5000, tan(60.00): 1.7321 sen(60.50): 0.8704, cos(60.50): 0.4924, tan(60.50): 1.7675 s18.txt

19. Em uma eleição sindical concorrem ao cargo de presidente três candidatos (X, Y e Z). Na apuração dos votos devem ser computados votos nulos (N) e votos em branco (B), além dos votos válidos para cada candidato. Deve ser escrito um programa que faça a leitura dos votos, armazenando-os em um vetor de caracteres, considerando que existem 200 eleitores. Deve ser impresso (conforme exemplo *s19.txt*):

- na primeira linha do arquivo, o vencedor da eleição;na segunda, se haverá segundo turno ("Segundo turno: S"), ou não ("Segundo turno: N");
- nas próximas três linhas devem ser impressos o total de votos de cada candidato, seguidos pelo percentual total e pelo percentual de votos válidos, nessa ordem de candidatos: X, Y e Z;
- nas próximas duas linhas devem ser impressos os totais e percentuais de votos brancos e nulo nessa ordem: B e N.

Observações:

- experimente ler os caracteres do arquivo de entrada utilizando a função getchar da biblioteca *stdio.h*;
- cada linha de entrada do arquivo contem 20 votos;
- para evitar problemas com leituras de espaços em branco e caractere de nova linha, experimente a função isspace da biblioteca *ctype.h*.
- Para não haver segundo turno, um dos candidatos deve possuir 50% + 1 dos votos válidos.

```
Z X X N X B Y X N Y B B N B Z Z X Z B Z
Y Y Y Y N N Z B N N B Z Y Z Y Z Z N X X
N Z B Z Y Y B B Y X X Y B N N B N N N B
N Y B B X Y Z N N B B B B Y X X Z N N Z
Y B Y N N Y B X X Y Z N Y B X N Z Y X Y
Y Z B X B B N X N Y N Z B Y B B N Y Z B
X Z X X N Z N X Z Z B X N B Z Y N N Z N
Y Y Y B Z N X Z Z Y N N N B B N Y Z B N
N Y Y Y B Z B X Z Y Y X X Y X Y N N X Y
N Y X Z X Y N N Y Y B Y Z Y Z Z B Z Z N
```

e19.txt

```
Vencedor: Y
Segundo turno: S
X: 31 15.50% 26.96%
Y: 46 23.00% 40.00%
Z: 38 19.00% 33.04%
B: 39 19.50%
N: 46 23.00%
```

s19.txt

20. Escreva um programa que leia um vetor de 25 posições de caracteres, verifique se existem valores iguais e os imprima, considerando a ordem alfabética, juntamente com a quantidade de repetições.

B A b A b b T a a a E J X m m s S t T R z T w T Z e20.txt
A: 2 T: 4 a: 3 b: 3 m: 2 s20.txt