	UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA-UESB DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS-DCET DISCIPLINA: ALGORITMOS E PROGRAMAÇÃO I PROFA. CÁTIA KHOURI	LISTA DE EXERCÍCIOS 2	FUNÇÕES

PARTE 1 Introdução

(1) Escreva um programa em C++ que leia do teclado um valor inteiro e armazene este valor em uma variável. Este valor deverá ser passado como argumento para uma função denominada quadrado. Esta função imprime na tela o quadrado do valor passado como argumento.

(2) O mesmo exercício anterior, mas agora a função deverá retornar para uma segunda variável do programa principal o quadrado do valor passado como argumento. Imprimir o valor calculado (estando no programa principal).

(3) Escreva uma função que receba um valor inteiro como parâmetro e devolva `1`, caso o valor passado seja par e `0`, caso contrário. Escreva um programa que receba do teclado um número inteiro positivo e informe se o número é par. Seu programa deve usar a função.

(4) Modifique a função anterior para que ela devolva `true` se o número recebido for múltiplo de 5 e `false`, caso contrário. Modifique o programa para usar adequadamente a função.

(5) Escreva uma função que conte de até 1.000.000.000, isto é, a função deve conter um laço for com um contador que vai de 1 até 1.000.000.000. Escreva um programa que receba do teclado um valor, chame a função e então mostre o valor digitado.

```
Exemplo de execução:
Digite um número: 653
Você digitou...  653
```

(6) Implementar a função **`raizquadrada`**. Esta função deve:

(a) Receber um número do tipo float como parâmetro.

(b) Retornar a raiz quadrada do número recebido de tal maneira que esta raiz, quando elevada ao quadrado, apresente um erro máximo de 0.01% em relação ao valor do parâmetro.

(7) Implementar a função **`inverte`** que recebe um número unsigned int como parâmetro e retorna este número escrito ao contrário. Ex: 431 <-> 134.

(8) Implementar a função `int` **`power`** `(int base, int expoente)`, que retorna o valor de base elevado a expoente.

(9) Faça uma função que recebe, por parâmetro, a hora de inicio e a hora de término de um jogo, ambas subdivididas em 2 valores distintos: horas e minutos. A função deve retornar a duração do jogo em minutos, considerando que o tempo máximo de duração de um jogo é de 24 horas e que o jogo pode começar em um dia e terminar no outro.

(10) Um número a é dito permutação de um número b se os dígitos de a formam uma permutação dos dígitos de b.

Exemplo: 5412434 é uma permutação de 4321445, mas não é uma permutação de 4312455.

Obs.: Considere que o dígito 0 (zero) não aparece nos números.

(a) Faça uma função **contadigitos** que dados um inteiro n e um inteiro d, 0 < d < 9, devolve quantas vezes o dígito d aparece em n.

(b) Usando a função do item anterior, faça um programa que lê dois inteiros positivos *a* e *b* e responda se *a* é permutação de *b*.

(11) Faça uma função que recebe um valor inteiro e verifica se o valor é positivo, negativo ou zero. A função deve retornar 1 para valores positivos, -1 para negativos e 0 para o valor 0.

Escreva um programa que receba um valor inteiro do teclado, chame a função e imprima uma mensagem a respeito do sinal do número lido (<0, >0 ou =0).

(12) Escreva um programa que receba 3 números e uma letra. Se a letra for A (ou a), deve ser chamada uma função que retorna a média aritmética dos números dados; se for P (ou p), deve ser chamada uma função que retorna a média ponderada (pesos: 5, 3 e 2); e se for H (ou h), chamas uma função que retorna a média harmônica. Cuide para uma letra válida seja processada (A, a, P, p, H, h).

Veja a fórmula de cada cálculo a seguir:

$$M_a = \frac{x_1+x_2+x_3+\cdots+x_n}{n} \qquad M_p = \frac{x_1p_1+x_2p_2+x_3p_3+\cdots+x_np_n}{p_1+p_2+p_3+\cdots+p_n} \qquad M_h = \frac{n}{\frac{1}{x_1}+\frac{1}{x_2}+\frac{1}{x_3}+\cdots+\frac{1}{x_n}}$$

Casos de teste:

X1	X2	X3	Tipo	Saída
5.5	6.5	7.5	X	"Tipo inválido"
5.5	6.5	7.5	A	6.5
5.5	6.5	7.5	P	6.2
5.5	6.5	7.5	H	6,39662

(13) Escreva uma função que receba dois números inteiros positivos m e n e retorne `true` se n é divisor de m e `false`, caso contrário.

Escreva um programa que solicite dois números inteiros positivos, chame a função e imprima uma mensagem informando se o primeiro número dado é divisor do segundo.

(14) Escreva as seguintes funções:

funcaoA_B – recebe dois valores inteiros $a < b$, e imprime os valores no intervalo $[a, b]$ em ordem crescente.

funcaoB_A – recebe dois valores inteiros $b < a$, e imprime os valores no intervalo $[b, a]$ em ordem decrescente.

funcaoIguais – mostra a mensagem "valores iguais".

Faça um programa que lê dois valores e, usando as funções acima, imprime:

- se o primeiro valor for menor que o segundo, a lista de valores do primeiro até o segundo;
- se o primeiro valor for maior que o segundo, a lista de valores do segundo até o primeiro em ordem decrescente;

- se ambos forem iguais, a mensagem "valores iguais".

Casos de teste:

a	b	Saída
4	12	4, 5, 6, 7, 8, 9, 10, 11, 12
20	15	20, 19, 18, 17, 16, 15
-3	-8	-3, -4, -5, -6, -7, -8
13	13	Valores iguais

(15) Escreva um programa que leia 50 valores inteiros e imprima seus respectivos valores absolutos. Para obter o valor absoluto do número utilize uma função absoluto que recebe um número inteiro e retorna seu valor absoluto.

Obs.: Não precisa ler todos os números primeiro; para cada número lido, você já imprime seu valor absoluto.

(16) Escreva um programa que simule uma calculadora com o uso de quatro funções: *soma*, *subtracao*, *multiplicacao*, *divisao*. O usuário informa a operação no formato:

<operando1> <operacao> <operando2> ,

e o programa chama a função correspondente à operação informada. A função chamada realiza a operação e retorna o resultado, o qual será mostrado pelo programa principal.

(17) Escreva uma função que calcule e retorne a distância entre dois pontos (x_1, y_1) e (x_2, y_2). Todos os números e valores de retorno devem ser do tipo float. Escreva um programa que receba as coordenadas de dois pontos, chame a função e imprima a distância entre eles.

Obs.: A distância entre dois pontos A (x_1, y_1) e B (x_2, y_2) é calculada da seguinte maneira:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

😉 **DICA**: Use as funções

```
double sqrt (double) e
double pow (double, double)
```

da biblioteca **cmath**

```
Caso de teste:
A(8, 3)
B(3, 6)
dAB = 5.83095
```

(18) Fazer um programa que pergunta uma medida em metros e imprime o valor correspondente em decímetros, centímetros e milímetros. Para cada unidade, o programa deve chamar uma função distinta para fazer a conversão, isto é, uma função chamada *decimetro*, outra chamada *centimetro* e outra chamada *milimetro*.

(19) Construa uma função **encaixa** que dados dois inteiros positivos a e b verifica se b corresponde aos últimos dígitos de a.

Ex.:

a	*b*	
567890	890	=> encaixa
1243	1243	=> encaixa
2457	245	=> não encaixa
457	2457	=> não encaixa

(a) Usando a função da **encaixa**, faça um programa que lê dois inteiros positivos a e b e verifica se o menor deles é segmento do outro.

Exemplo:

a	*b*	
567890	678	=> b é segmento de a
1243	2212435	=> a é segmento de b
235	236	=> um não é segmento do outro

(20) (MAT-94) Uma sequência de n números inteiros não nulos é dita piramidal m-alternante se é constituída por m segmentos: o primeiro com um elemento, o segundo com dois elementos e assim por diante até o m-ésimo, com m elementos. Além disso, os elementos de um mesmo segmento devem ser todos pares ou todos ímpares e para cada segmento, se seus elementos forem todos pares (ímpares), os elementos do segmento seguinte devem ser todos ímpares (pares).

Por exemplo, a sequência com *n* = 10 elementos:

12 3 7 2 10 4 5 13 5 11 é piramidal 4-alternante.

A sequência com *n* = 3 elementos:

7 10 2 é piramidal 2-alternante.

A sequência com *n* = 8 elementos:

1 12 4 3 13 5 12 6 não é piramidal alternante pois o último segmento não tem tamanho 4.

(a) Escreva uma função **bloco** que recebe como parâmetro um inteiro n e lê n inteiros do teclado, devolvendo um dos seguintes valores:

0, se os *n* números lidos forem pares;
1, se os *n* números lidos forem ímpares;
-1, se entre os *n* números lidos há números com paridades diferentes.

(b) usando a função do item anterior, escreva um programa que, dados um inteiro $n \geq 1$ e uma sequência de *n* números inteiros, verifica se ela é piramidal *m*-alternante. O programa deve imprimir o valor de *m* ou dar a resposta *não*.

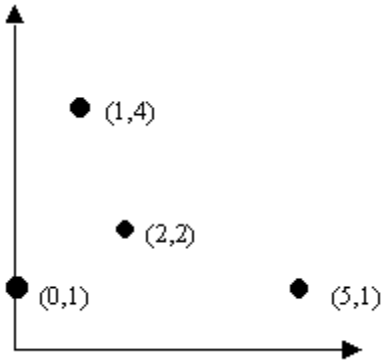
(21) Faça uma função **arctan** que recebe o número real $x \in [0,1]$ e devolve uma aproximação do arco tangente de x (em radianos) através da série incluindo todos os termos da série

$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$, incluindo todos os termos da série até $\left|\frac{x^k}{k}\right| < 0{,}0001$.

(a) Faça uma função **angulo** que recebe um ponto de coordenadas cartesianas reais (*x*,*y*), com $x \geq 0$ e $y \geq 0$ e devolve o ângulo formado pelo vetor (*x*,*y*) e o eixo horizontal.

Exemplos: Observe a figura abaixo e verifique que os ângulos correspondentes aos pontos marcados são, aproximadamente:

(0,1) 90 graus
(2,2) 45 graus
(1,4) 75 graus
(5,1) 11 graus



Use a função do item anterior mesmo que você não a tenha feito. Note que a função só calcula o arco tangente de números entre 0 e 1, e o valor devolvido é o ângulo em radianos (use o valor π = 3.14 radianos = 180 graus).

Para calcular o valor do ângulo α pedido, use a seguinte fórmula:

$$\alpha = \begin{cases} arctan\left(\frac{y}{x}\right), caso\ y < x \\ \frac{\pi}{2} - arctan\left(\frac{x}{y}\right), caso\ contrário \end{cases}$$

(b) Faça um programa que, dados *n* pontos do primeiro quadrante (x $\geq$ 0 e *y* $\geq$ *0*) através de suas coordenadas cartesianas, determina o ponto que forma o menor ângulo com o eixo horizontal. Use a função do item anterior, mesmo que você não a tenha feito.

(22) Escreva uma função que recebe um inteiro positivo m e devolve 1 se m é primo, 0 em caso contrário.

(23) Escreva um programa que leia um inteiro não-negativo *n* e imprima a soma dos *n* primeiros números primos. Use a função anterior.

(24) Escreva uma função que recebe como parâmetros dois números *a* e *b* e devolve o *mdc* (máximo divisor comum) de *a* e *b*, calculado por meio do algoritmo de Euclides.

(25) Escreva um programa que leia um inteiro positivo *n* e uma sequência de *n* inteiros não-negativos e imprime o *mdc* de todos os números da sequência. Use a função anterior.

PARTE 2 Geração de números aleatórios

(26) Crie um jogo onde o computador sorteia um número de 0 até 50, e você tenta adivinhar qual é. Inicialmente, o programa deve chamar a função gerador para gerar o número a ser adivinhado. Faça com que o programa diga quantas tentativas você levou para acertar. A cada vez que você chutar, ele deve dizer se você chutou abaixo do valor real, acima ou se acertou. Ao final, diz o número de tentativas que você usou e se bateu o recorde ou não. Ah, ao final de cada rodada, o programa pergunta se você quer jogar novamente ou não, exibindo o recorde atual.

(a) Altere a função **`gerador`** de modo que o número gerado esteja no intervalo [1, 50].

(b) Altere a função **`gerador`** de modo que o número gerado esteja no intervalo [10, 50].

(27) Faça um programa para lançar uma moeda. O programa deve chamar a função **`cara_coroa`** que retorna *cara* ou *coroa*. Crie outra função - **`lancamentos`**, que receba com argumento um inteiro

n e faça n lançamentos de moedas, onde n é o valor que o usuário quiser, e mostre a porcentagem de vezes que deu *cara* e *coroa*.

(28) Faça um programa semelhante ao anterior mas agora a função ***lancamentos*** vai chamar a função ***cara_coroa n*** vezes e imprimir o que é retornado por ela. A função ***cara-coroa*** vai contar o número de lançamentos feitos e o número de vezes em que o resultado é cara. Cada vez que ela for chamada, ela deve retornar o percentual de ocorrências de cara até o instante atual.

RESTRIÇÕES:

1. A função ***cara-coroa*** não tem qualquer parâmetro de entrada.
2. Não use qualquer variável global.

DICA: Use variáveis estáticas para fazer as contagens.

(29) Escreva um programa para jogar par ou ímpar com o computador. O seu programa deve pedir ao jogador para escolher entre par ou ímpar e em seguida escolher um número entre 0 e 10. Em seguida, um número entre 0 e 10 deve ser gerado para representar a jogada do computador. O programa então deve dizer quem ganhou e perguntar se o jogador quer outra rodada. Caso ele informe que sim, o processo se repete. Um placar também deve ser mantido e apresentado pelo programa.

RESTRIÇÕES:

1. O programa deve chamar as seguintes funções:
   -> ***jogo***
      - solicita ao jogador que escolha par ou ímpar (0 para par e 1 para ímpar);
      - solicita ao jogador que escolha um inteiro;
      - chama a função numero_computador.
      - verifica quem ganhou a rodada e retorna 'j' para jogador e 'c' para computador.
   -> ***numero_computador***
      - gera um número aleatório entre 0 e 10 e retorna esse número.
   -> ***placar***
      - recebe o ganhador da rodada (j ou c), atualiza e mostra o placar.
2. Não use qualquer variável global.

DICA: use variáveis estáticas em placar.

Veja um exemplo de execução do programa:

```
--------------------------------
Par ou ímpar?
0 para par
1 para ímpar
0
Digite a sua jogada: 5
--------------------------------
Sua escolha: PAR
Sua jogada: 5
Jogada do computador: 7
```

```
Vencedor da rodada: você

============ PLACAR ============
  Você: 1        Computador: 0
================================

Mais uma rodada (S/N)? S
--------------------------------

Par ou ímpar?

0 para par

1 para ímpar
1
Digite a sua jogada: 8
--------------------------------
Sua escolha: ÍMPAR
Sua jogada: 8
Jogada do computador: 2
Vencedor da rodada: computador

============ PLACAR ============
  Você: 1        Computador: 1
================================

Mais uma rodada (S/N)? S
--------------------------------

Par ou ímpar?

0 para par

1 para ímpar
0
Digite a sua jogada: 0
--------------------------------
Sua escolha: PAR
Sua jogada: 4
Jogada do computador: 0
Vencedor da rodada: você

============ PLACAR ============
  Você: 2        Computador: 0
================================

Mais uma rodada (S/N)? N
```

(30) Escreva um programa para jogar pedra-papel-tesoura com o computador. O seu programa deve pedir ao jogador para escolher entre pedra, papel e tesoura (um número entre 1 e 3, inclusive). Em seguida, um número aleatório entre 1 e 3 (inclusive) deve ser gerado para representar a jogada do computador. O programa então deve dizer quem ganhou e perguntar se o jogador quer outra rodada. Caso ele informe que sim, o processo se repete. Um placar também deve ser mantido e apresentado pelo programa.

REGRAS DO JOGO:

PAPEL enrola PEDRA TESOURA corta PAPEL PEDRA quebra TESOURA

isto é:

PAPEL vence PEDRA; TESOURA vence PAPEL; PEDRA vence TESOURA.

RESTRIÇÕES:

1. O programa deve chamar as seguintes funções:

-> ***jogo***

- solicita ao jogador que escolha pedra-papel-tesoura (1 para pedra, 2 para papel e 3 para tesoura);
- chama a função jogada_computador.
- verifica quem ganhou a rodada e retorna 'j' para jogador e 'c' para computador.
- mostra a jogada de cada um (jogador e computador);

-> ***jogada_computador***

- gera um número aleatório entre 0 e 10 e retorna esse número.

-> ***placar***

- recebe o ganhador da rodada (j ou c);
- mostra o vencedor da rodada (jogador ou computador);
- atualiza e mostra o placar.

2. Não use qualquer variável global.

DICA: use variáveis estáticas em placar.

Veja um exemplo de execução do programa:

```
Pedra-Papel-Tesoura?
1 para pedra
2 para papel
3 para tesoura
--------------------------------
3
Sua jogada:1
Jogada do computador:1
Vencedor da rodada: computador

========== PLACAR ==============
  Você: 0        Computador: 1
================================

Mais uma rodada (S/N)? S
--------------------------------
Pedra-Papel-Tesoura?
1 para pedra
```

```
2 para papel

3 para tesoura

-------------------------------
2
Sua jogada:2
Jogada do computador:1
Vencedor da rodada: você

========== PLACAR ==============
  Você: 1        Computador: 1
================================

Mais uma rodada (S/N)? S
-------------------------------
Pedra-Papel-Tesoura?

1 para pedra

2 para papel

3 para tesoura

-------------------------------
1
Sua jogada:1
Jogada do computador:1
Vencedor da rodada: empate

========== PLACAR ==============
  Você: 1        Computador: 1
================================

Mais uma rodada (S/N)? N
-------------------------------
```

(31) Crie um jogo 21 (BlackJack) em C++. Existe uma infinidade de formas de se jogar 21; o objetivo sempre é formar 21 pontos com cartas (ou chegar o mais próximo possível). O nosso jogo será uma versão simplificada, para um jogador solitário. Ele deve ter a chance de permanecer jogando uma rodada após a outra até decidir parar. Veja as regras:

(a) Valor das cartas: todas as cartas com figura (valete, dama e rei) valem 10 pontos; o ás, vale 1; e as demais cartas valem o seu próprio número.

(b) Incialmente, o jogador recebe 3 cartas (e calcula a soma de pontos).

(c) Se a soma de seus pontos der 21, ele já vence no início.

(d) Se ele ultrapassar os 21 pontos, já perde no início.

(e) Se passar do início, para formar os 21 pontos, o jogador pode "comprar" uma carta do monte, quantas vezes quiser, e somar os pontos desta carta aos seus pontos atuais. Mas ele deve fazer isso com cautela. Se em algum momento seus pontos ultrapassarem 21, ele perde o jogo.

(f) O jogador deve informar quando quiser para de comprar cartas e encerrar a rodada (caso não tenha atingido (vence) ou ultrapassado (perde) os 21 pontos. Se ele parar sem atingir os 21,

ele deve ver uma mensagem informando os seus pontos e o recorde atual – o maior número de pontos já atingido considerando todas as rodadas.

RESTRIÇÕES:

O seu programa deve atender aos seguintes requisitos:

- Use uma função ***compra_carta*** que sorteia um número de 1 a 13 representando uma carta (ás, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, valete, dama, rei) e retorna o número de pontos correspondente à carta;
- Use uma função ***recorde*** que
  - registra a pontuação atingida em cada rodada;
  - atualiza o recorde, se necessário; e
  - mostra essa a pontuação da rodada comparando com o recorde.

  DICA: use uma variável estática aqui 😉.
- Use uma função ***jogo*** que, com a ajuda das funções acima implementa o jogo.
- Chame a função jogo a partir do programa principal provendo nele uma estrutura de repetição que permita ao jogador repetir o jogo (iniciar uma nova rodada) sempre que desejar.

PARTE 3 Referências

(32) Escreva uma função que retorna a razão entre dois números. A função deve retornar, pelo comando return, o valor 1, se a operação foi possível; e o valor 0, se a operação não foi possível (divisão por zero, por exemplo). O resultado da divisão deve ser retornado por um parâmetro por referência.

Escreva um programa que chame a função.

(33) Escreva uma função que receba dois inteiros como argumentos e retorne o valor da divisão de cada um deles por 10.

Note que além dos inteiros que serão operados, a função deverá receber dois argumentos por referência a fim de armazenar os resultados das divisões por 10.

Escreva um programa que chame a função.

(34) Faça uma função chamada **`ePrimo`** que recebe como parâmetro (pelo menos) um inteiro positivo n e retorna true se n for primo, e false, caso contrário.

(a) Faça uma função chamada **`doisPrimos`** que recebe como parâmetro (pelo menos) um inteiro positivo m e retorna o maior número primo que é menor do que m e o menor número primo que é maior do que m. Esta função deve chamar **`ePrimo`**.

(b) Faça um programa que peça ao usuário um número inteiro positivo n e imprima os números primos mais próximos dele (um maior e outro menor que ele).

OBSERVAÇÃO: Fique à vontade para incluir outros parâmetros para as funções.

RESTRIÇÕES:

- Não utilize variáveis globais.

Exemplos de execução:

Digite um número inteiro positivo:
45
43 é o maior número primo menor que 45.

47 é o menor número primo maior que 45.

Digite um número inteiro positivo:
2
Não existe número primo menor que 2.
3 é o menor número primo maior que 2.

(35) Escreva um programa que receba 3 números e chame uma função que calcule:

(a) a média aritmética dos números dados;

(b) a média ponderada (pesos: 1, 3 e 6); e

(c) a média harmônica.

O programa principal deve mostrar os resultados. Veja a fórmula de cada média a seguir:

$M_a = \frac{x_1+x_2+x_3+\cdots+x_n}{n}$, onde n é o número de valores x_i, cuja média aritmética se deseja calcular.

$M_p = \frac{x_1p_1+x_2p_2+x_3p_3+\cdots+x_np_n}{p_1+p_{2+}p_3+\cdots+p_n}$, onde p_i é o peso de x_i .

$M_h = \frac{n}{\frac{1}{x_1}+\frac{1}{x_2}+\frac{1}{x_3}+\cdots+\frac{1}{x_n}}$, onde n é o número de valores x_i, cuja média harmônica se deseja calcular.

RESTRIÇÕES:

- Não utilize variáveis globais.
- O programa deve permanecer perguntando ao usuário se quer calcular as médias de três valores até que ele informe que não.

==========
Exemplo de execução:

Digite três números: 5.5 6.5 7.5

MA = 6.5
MP = 7.0
MH = 6.39662

Quer calcular médias de novo? Digite S/N: s

Digite três números: -4.25 2.02 1.3

MA = -0.31
MP = 0.961
MH = 2.91549

Quer calcular médias de novo? Digite S/N: n

Process returned 0 (0x0) execution time : 24.830 s
Press any key to continue.

(36) Escreva um programa que receba um número inteiro positivo e informe se ele é perfeito. Um número é dito perfeito se ele for igual à soma de seus divisores excetuando-se ele próprio. Caso o número informado seja negativo, uma mensagem de erro deve ser emitida, seguida de nova solicitação para que o usuário digite um número inteiro positivo.

Além do programa principal, você deve escrever as seguintes funções:

(a) **`ePositivo`** – função que recebe um inteiro e retorna true, se o número é positivo, e false, caso contrário. O programa deve usar esta função para verificar a validade do número fornecido pelo usuário.

(b) **`somaDivisores`** – função que recebe um inteiro positivo n e retorna a soma de seus divisores.

(c) **`ePerfeito`** – função que recebe um inteiro n, chama **`somaDivisores`** e retorna true, se n é perfeito, e false, caso contrário. No caso de n não ser perfeito, a função também deve retornar o valor da soma dos divisores de n.

RESTRIÇÕES:

- O programa principal deve chamar apenas as funções **`ePositivo`** e **`ePerfeito`**.
- Não utilize variáveis globais.

`Exemplos de execução:`

Digite um número inteiro positivo:

-567

Dado inválido. Digite um número inteiro positivo:

463

O número dado não é perfeito. A soma de seus divisores é igual a 1

==========

Digite um número inteiro positivo:

496

O número dado é perfeito.

==========

(37) Crie um jogo para um jogador que funcione como segue.

(a) O objetivo é obter o maior número de pontos a partir da soma dos números das faces superiores de 3 dados.

(b) O jogador deve ter até 8 chances de lançar os dados, mas ele pode parar quando desejar. A pontuação atual do jogador é a que foi obtida no lançamento mais recente, isto é, ele pode aumentar ou diminuir sua pontuação a cada jogada.

(c) Cada vez que ele lançar os dados, a pontuação correspondente deve ser calculada e uma mensagem deve informar se ele ganhou ou perdeu pontos com relação à sua melhor jogada até então.

Escreva um programa que execute o jogo. Para isso, ele deve chamar a função **`lanca3dados`**, que simula o lançamento de três dados e retorna o valor obtido na face de cada um deles – **`d1`**, **`d2`** e **`d3`** –bem como a soma das faces.

A função **`lanca3dados`**, por sua vez, deve chamar a função **`soma3Dados`**, que recebe 3 inteiros e retorna o valor da soma.

O programa deve perguntar ao jogador se deseja lançar os dados (até, no máximo, 8 vezes), mostrar o resultado de cada dado e a soma dos 3 em cada jogada, bem como a mensagem de perda ou ganho com relação à jogada anterior.

Exemplo de Execução 1:

```
Deseja lançar os dados (S/N)? s

Sua melhor pontuação até agora = 12
d1 = 4  d2 = 1  d3 = 6  ==> 11 pontos.
Não bateu seu recorde...

Deseja lançar os dados (S/N)? s

Sua melhor pontuação até agora = 12
d1 = 5  d2 = 2  d3 = 4  ==> 11 pontos.
Não bateu seu recorde...

Deseja lançar os dados (S/N)? s

Sua melhor pontuação até agora = 12
d1 = 1  d2 = 5  d3 = 1  ==> 7 pontos.
Não bateu seu recorde...

Deseja lançar os dados (S/N)? s

Sua melhor pontuação até agora = 12
d1 = 4  d2 = 5  d3 = 2  ==> 11 pontos.
Não bateu seu recorde...

Deseja lançar os dados (S/N)? s

Sua melhor pontuação até agora = 12
d1 = 5  d2 = 3  d3 = 3  ==> 11 pontos.
Não bateu seu recorde...

Deseja lançar os dados (S/N)? s

Sua melhor pontuação até agora = 12
d1 = 1  d2 = 6  d3 = 2  ==> 9 pontos.
Não bateu seu recorde...

Deseja lançar os dados (S/N)? s

Jogou 8 vezes.
Process returned 0 (0x0)   execution time : 11.954 s
Press any key to continue.
```

Exemplo de Execução 2:

```
Deseja lançar os dados (S/N)? s

Sua melhor pontuação até agora = 0
d1 = 6  d2 = 3  d3 = 6  ==> 15 pontos.
Você melhorou sua pontuação!

Deseja lançar os dados (S/N)? s
```

```
Sua melhor pontuação até agora = 15
d1 = 4  d2 = 1  d3 = 4  ==> 9 pontos.
Não bateu seu recorde...

Deseja lançar os dados (S/N)? s

Sua melhor pontuação até agora = 15
d1 = 6  d2 = 6  d3 = 1  ==> 13 pontos.
Não bateu seu recorde...

Deseja lançar os dados (S/N)? n

Jogou 4 vezes.
Process returned 0 (0x0)   execution time : 11.970 s
Press any key to continue.
```

PARTE 4 Sobrecarga, Gabarito, Argumentos default

(38) Escreva três funções soma sobrecarregadas que retornem, respectivamente:

(a) a soma de dois valores inteiros;

(b) a soma de um inteiro e um float;

(c) a soma de três inteiros;

Escreva um programa que use as funções.

(39) Escreva uma função gabarito **`multiplos`**, com tipo de retorno `void` que recebe três parâmetros: `soma`, `x`, e `n`. Os dois primeiros parâmetros devem ter o mesmo tipo, o qual será definido na chamada da função (um coringa). O terceiro parâmetro será sempre inteiro. soma é passado por referência e os demais argumentos, por valor. Essa função irá computar:

```
soma = 1 + x + 2x + 3x + … + nx
```

Escreva um programa que chame a função passando para ela os seguintes argumentos. O programa deve imprimir o valor de soma para cada caso.

Caso 1.	$x_1 = 2, n_1 = 10$	`(soma = 111)`
Caso 2.	$x_2 = 3.5, n_2 = 15$	`(soma = 421)`

(40) Escreva funções **`media`** sobrecarregadas que retornem, respectivamente:

(a) a média de quatro inteiros;

(b) a média de três valores float;

(c) a média de dois valores double;

Escreva um programa que use as funções.

(41) Escreva uma função gabarito que receba um argumento x que pode ser inteiro ou char. Se o argumento for inteiro, a função retorna o valor consecutivo. Por ex.: para x = 47, o valor retornado é 48. Se o argumento passado é char, a função retorna o caractere consecutivo na tabela ASCII. Por ex.: para x = 'g', retorna 'h'.

(42) Escreva funções **`area`** sobrecarregadas que retornem, respectivamente:

(a) a área de um quadrado, dado seu lado;

(b) a área de um retângulo, dados base e altura;

(c) a área de um triângulo retângulo, dados sua base e altura;

Escreva um programa que use as funções.

(43) Escreva uma função gabarito que receba dois argumentos x e y que podem ser ambos inteiros ou reais, e retorne o produto entre eles.

Escreva um programa que solicite 2 números inteiros e 2 números reais e, com ajuda da função, imprima os produtos dos valores de mesmo tipo.

(44) Escreva funções volume sobrecarregadas que retornem, respectivamente:

(a) O volume de um cubo, dada sua aresta;

(b) O volume de um paralelepípedo, dadas suas três dimensões;

(c) O volume de um cilindro, dados seu raio e altura.

Escreva um programa que use as funções.

(45) Escreva uma função gabarito que receba dois argumentos x e y que podem ser ambos inteiros, char ou reais, e troque o valor dos argumentos passados.

Escreva um programa que solicite 2 números inteiros, 2 reais e 2 caracteres e armazene-os nas variáveis i1, i2, r1, r2, c1 e c2, respectivamente. Com ajuda da função, troque os valores das variáveis aos pares, isto é, troque i2 com i2; r1 com r2 e c1 com c2. O programa deve imprimir os valores das variáveis antes e depois da chamada correspondente da função.

(46) Em uma determinada empresa, o salário de um funcionário depende de seu cargo, conforme o quadro abaixo, e é igual ao salário-base mais o valor correspondente às horas extras trabalhadas no mês, quando houverem.

CARGO	SALÁRIO-BASE	VALOR DA HORA EXTRA
1 (Vendedor)	2.000,00	17,05
2 (Chefe de setor)	2.500,00	21,31
3 (Gerente)	4.000,00	34,10

Escreva funções **`salario`** sobrecarregadas que recebam os dados dos funcionários e imprimam seu salário no mês. Uma das funções deve receber o código do cargo e o número de horas extras trabalhadas. A outra recebe apenas o código do cargo.

Escreva um programa que receba os dados de todos os trabalhadores da empresa e com a ajuda das funções acima, imprima o salário de cada um. Caso o funcionário não tenha trabalhado horas extras no mês, o programa deve passar apenas o código do cargo para a função salario.

(47) Escreva um template de função que receba três argumentos: x, y e z, que podem ser todos do tipo inteiro ou float. A função deve retornar a média aritmética dos valores recebidos.

Escreva um programa que solicite 3 números inteiros e 3 números reais e, com ajuda da função, imprima as médias dos valores de mesmo tipo.