Lista de Exercícios I

PE-L1 – v1.0

Prof. Paulo Joia Filho

Ferramentas necessárias

- Para resolver os exercícios você irá precisar de um compilador C instalado, preferencialmente:
  - *GNU Compiler Collection* (GCC) para plataformas Linux; ou
  - *Minimalist GNU for Windows* (MinGW) para plataformas Windows.

- Lembre-se: a lista de exercícios é uma atividade individual.

  ☞ *Neste tipo de atividade o capricho e a organização são importantes.*

Apresentação dos resultados e entrega

Passos a serem seguidos:

1. Crie um documento no **LibreOffice Writer** (ou Microsoft Word se preferir).

2. Neste documento, faça uma capa simples, intitulada:

   Lista de Exercícios I

   A capa deve conter:
   - Disciplina, turma e turno;
   - RA e seu nome completo.

3. Apresente as soluções dos exercícios em ordem crescente, conforme proposto na lista.

   ☞ *Apresentar o enunciado do exercício no documento é opcional.*

Apresentação dos resultados e entrega

4. Salve o documento com o nome na forma:

   BCC_MeuPrimeiroNome_MeuRA *(sem espaços!)*

   Exemplo:

   BCC_Paulo_11201810999

   Após resolver a lista, exporte o documento para o formato pdf, usando o mesmo nome e extensão **.pdf**.

5. Crie uma pasta com o mesmo nome. Exemplo: BCC_Paulo_11201810999 e salve cada programa C dentro dela, com a seguinte nomenclatura:

   ex1.c, ex2.c, ...

   Ao finalizar, compacte a pasta de modo a produzir um arquivo com o mesmo nome e extensão **.zip**.

Apresentação dos resultados e entrega

6. Envie os dois documento produzidos, o **.pdf** e o **.zip** para o email:

paulo.joia@ufabc.edu.br

No assunto (subject) especifique: **PE-Lista1**

❖ **Observações Importantes: preste muita atenção!**

- Se os arquivos .pdf e/ou .zip ficarem muito grandes (acima de 25MB) você terá que reduzi-los para enviar por email.
- Se um dos arquivos não for enviado a lista não será considerada.
- Questões que não apresentarem o programa C correspondente não serão consideradas.
- Questões com arquivos de código fora do padrão de nomes serão penalizadas.

Solução esperada

Apresente toda informação empregada na solução de forma organizada!

- **Programas C**
  1. Toda questão deve apresentar um arquivo contendo o código-fonte em C.
  2. Salve cada arquivo de acordo com o padrão de nomes explicado anteriormente.
  3. Faça comentários no código para aumentar a clareza do documento quando necessário.
- **Documento PDF**
  1. Faça um *print screen* da tela de código do **gedit** ou outro editor que estiver usando e apresente no documento pdf.
  2. Logo abaixo mostre o resultado da execução do programa para algum valor de sua escolha.
  3. Se necessário, explique a sequência de passos de forma clara e objetiva.

Expressões Matemáticas

Exercício 1

Escreva um programa para calcular a distância entre dois pontos (x_1, y_1) e (x_2, y_2) no plano cartesiano. Os pontos serão informados pelo usuário[1]. A distância entre dois pontos é dada por:

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

Nota:

– Formate a saída com "%g".

Exemplo de Funcionamento

```
Informe o primeiro ponto: 4 4
Informe o segundo ponto: 7 8
A distância entre os pontos é: 5
```

[1] Sempre que os dados forem informados pelo usuário, utilize o comando **scanf**.

Expressões Matemáticas

Exercício 2

Escreva um programa em C para calcular a resistência equivalente entre dois resistores R_1 e R_2, em paralelo. Lembre-se que a resistência equivalente entre dois resistores em paralelo é dado por:

$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2}$$

Nota:

– Formate a saída com 3 casas decimais.

Exemplo de Funcionamento

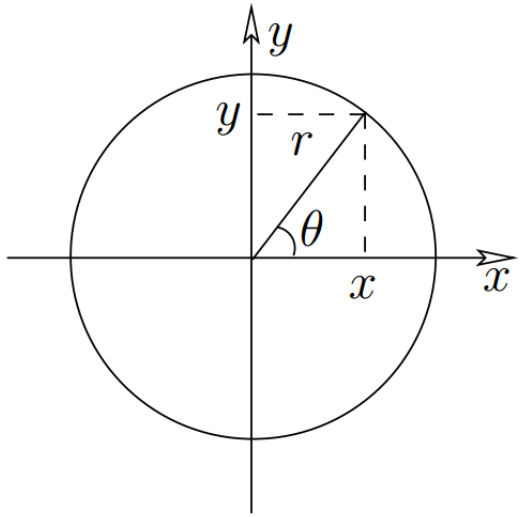
```
Informe o valor de R1: 7.8
Informe o valor de R2: 12.5
Req = 4.803
```

Expressões Matemáticas

Exercício 3

A localização de um ponto em um plano cartesiano pode ser expressa por coordenadas retangulares (x, y) ou coordenadas polares (r, θ). A relação entre estes dois sistemas de coordenadas pode ser expressa como:

$$
\begin{aligned}
x &= r\ \cos(\theta) \\
y &= r\ \sin(\theta) \\
r &= \sqrt{x^2 + y^2} \\
\theta &= \arctan\left(\frac{y}{x}\right)
\end{aligned}
$$



Escreva duas funções: **`rect2polar`** e **`polar2rect`** para conversão entre os dois sistemas de coordenadas, com o ângulo θ expresso em graus. Implemente também um menu de operações usando **`do...while`** para o usuário escolher a conversão que deseja.

Exercício 3

Alguém consegue reproduzir o comportamento abaixo?

Exemplo de Funcionamento

```
*=========================================*
* Conversão entre Sistemas de Coordenadas *
*=========================================*
* [1] Conversão retangular => polar       *
* [2] Conversão polar => retangular       *
* [3|<alfa>] Sair                         *
*=========================================*
Escolha uma opção: 1

Entre as coordenadas retangulares: x y = 0.866025 0.5
x = 0.866025, y = 0.5 ==> r = 1, theta = 30

Escolha uma opção: 2

Entre as coordenadas polares: r theta = 1 30
r = 1, theta = 0.523599 ==> x = 0.866025, y = 0.5

Escolha uma opção: 3
Programa encerrado.
```

Exemplo de Validação

```
*=========================================*
* Conversão entre Sistemas de Coordenadas *
*=========================================*
* [1] Conversão retangular => polar       *
* [2] Conversão polar => retangular       *
* [3|<alfa>] Sair                         *
*=========================================*
Escolha uma opção: —1
Resposta inválida, informe 1, 2 ou 3.
Escolha uma opção: 0
Resposta inválida, informe 1, 2 ou 3.
Escolha uma opção: 4
Resposta inválida, informe 1, 2 ou 3.
Escolha uma opção: 2.2
Resposta inválida, informe 1, 2 ou 3.
Escolha uma opção: 1.1
Resposta inválida, informe 1, 2 ou 3.
Escolha uma opção: s
Programa encerrado.
```

Estruturas de Decisão

Definição (Desigualdade Triangular)

Em todo triângulo, o comprimento de um dos lados é sempre inferior à soma dos comprimentos dos outros dois lados.

Exercício 4

Elaborar um programa para ler três medidas a, b e c. Em seguida, verificar se elas podem ser as medidas dos lados de um triângulo. Se forem, verificar se o triângulo é **equilátero**, **isósceles** ou **escaleno**.

Lembrando que...

- Triângulo equilátero tem três lados de comprimentos iguais.
- Triângulo isósceles tem dois lados de comprimentos iguais.
- Triângulo escaleno os lados não têm comprimentos iguais.

Exercício 4

Exemplos de Funcionamento

```
Informe as medidas dos lados: a b c = 1 2 3
não é triângulo
```

```
Informe as medidas dos lados: a b c = 2 3 3
triângulo isósceles
```

```
Informe as medidas dos lados: a b c = 3 4 5
triângulo escaleno
```

```
Informe as medidas dos lados: a b c = 6 6 12
não é triângulo
```

```
Informe as medidas dos lados: a b c = 6 6 11.9
triângulo isósceles
```

```
Informe as medidas dos lados: a b c = 7 7 7
triângulo equilátero
```

Estruturas de Decisão

Exercício 5

O custo c de enviar um pacote com peso p, por Sedex, é definido abaixo:

- R$ 10,00 para o primeiro quilo, i.e., se $p \leqslant 1$;
- R$ 3,75 para cada quilo adicional (e.g., se $1 < p \leqslant 2$, $c = 13{,}75$);
- Se o pacote pesar mais de 35 quilos, uma taxa fixa de R$ 10,00 é adicionada ao custo.
- Nenhum pacote com mais de 50 quilos é aceito.

Escreva um programa que aceite o peso do pacote, em quilos, como entrada e calcule o custo de enviar o pacote. Inclua o caso dos pacotes acima do peso.

Exercício 5

Exemplos de Funcionamento

```
Informe o peso do pacote em Kg: 0
Entrada inválida: informe um valor maior que zero e menor que 50 Kg.
```

```
Informe o peso do pacote em Kg: 50.01
Entrada inválida: informe um valor maior que zero e menor que 50 Kg.
```

```
Informe o peso do pacote em Kg: 1
O custo total de envio será: R$ 10.00
```

```
Informe o peso do pacote em Kg: 2
O custo total de envio será: R$ 13.75
```

```
Informe o peso do pacote em Kg: 2.01
O custo total de envio será: R$ 17.50
```

```
Informe o peso do pacote em Kg: 35
O custo total de envio será: R$ 137.50
```

```
Informe o peso do pacote em Kg: 35.01
O custo total de envio será: R$ 151.25
```

Estruturas de Repetição

Exercício 6

Uma sequência de Fibonacci começa com os números 0 e 1 e, cada número subsequente é a soma dos dois números anteriores a ele. Por exemplo, uma sequência formada por 10 números é dada por:

0 1 1 2 3 5 8 13 21 34

Construa a função **`fib`** para retornar a sequência de Fibonacci para n números, n inteiro e maior que 2.

Exemplos de Funcionamento

```
*** Série de Fibonacci ***
Informe o número de termos: 20
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
```

```
*** Série de Fibonacci ***
Informe o número de termos: 2
O nr de termos deve ser maior do que 2.
```

Estruturas de Repetição

Exercício 7

Crie a função **`fat`** para calcular o fatorial de um número inteiro $n \geqslant 0$. Use estruturas de repetição com o comando **`while`** para praticar. Lembre-se de validar a condição $n \geqslant 0$ e que $0! = 1$.

Nota: use **`size_t`** para armazenar o fatorial.

Exemplos de Funcionamento

```
Entre um nr inteiro para calcular o fatorial: -1
O nr deve ser maior ou igual a zero.
```

```
Entre um nr inteiro para calcular o fatorial: 0
fat[0] = 1
```

```
Entre um nr inteiro para calcular o fatorial: 5
fat[5] = 120
```

Agora responda: qual o maior fatorial que você consegue calcular na sua arquitetura?

Estruturas de Repetição

Exercício 8

Faça uma função em C chamada **`is_prime`** que receba um número inteiro $n > 1$ como entrada e retorne se n é primo ou não.

Condições:

- Se $n \leqslant 1$, exibir mensagem de advertência e sair;
- Se $n = 2$, então n é primo;
- Procurar pelos divisores inteiros de n no intervalo 2 a $n/2$.

Exemplos de Funcionamento

```
Informe um inteiro para verificar se é primo: 1
O nr informado deve ser maior do que um
```

```
Informe um inteiro para verificar se é primo: 10865903071
10865903071 NÃO É primo
```

```
Informe um inteiro para verificar se é primo: 8803424081
8803424081 É primo
```

Estruturas de Repetição

Desenvolvimento em Série

Muitas funções matemáticas podem ser calculadas por meio de um somatório infinito de termos. Em cada caso, a precisão aumenta à medida que mais termos da série são considerados. Um exemplo é a função cos x:

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$$

Para cálculos computacionais, no entanto, este somatório deve terminar após um número finito de termos (penalizando a precisão do resultado).

Exercício 9 (Estimando o valor do cosseno)

Escreva a função **`cosine(x,n)`** com duas variáveis de entrada, onde a primeira variável de entrada `x` representa o ângulo em radianos e a segunda variável de entrada `n`, representa o número de termos a serem utilizados nos cálculos. Notas:

- Leia o ângulo em graus e converta para radianos antes de chamar a função;
- Chame a função `fat` do Exercício 7 para calcular os valores do denominador.

Exercício 9

Exemplos de Funcionamento

```
Informe o ângulo em graus: 60
Informe o nr de termos da série: 7
cos[60] = 0.500000000021778
```

```
Informe o ângulo em graus: 30
Informe o nr de termos da série: 9
cos[30] = 0.866025403784439
```

```
Informe o ângulo em graus: 45
Informe o nr de termos da série: 9
cos[45] = 0.707106781186547
```

Estruturas de Repetição

Método de Newton

Os antigos babilônios usavam a seguinte aproximação (baseada no Método de Newton) para calcular $\sqrt{a}$:

$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{a}{x_k}\right)$$

Exercício 10 (Estimando o valor da raiz quadrada)

Crie a função **`square_root`** para calcular $\sqrt{a}$ usando a aproximação acima:

- A função deve receber um valor inicial aproximado para a raiz (x_k);
- Execute o cálculo até que $|x_{k+1} - x_k| < \epsilon$, onde ϵ é um valor suficientemente pequeno (0.001, por exemplo);
- Use a função para calcular $\sqrt{21}, \sqrt{3}, \sqrt{2}, \ldots$ e compare com os valores fornecidos pelo Python.

Exercício 10

Exemplos de Funcionamento

```
Informe o valor a ser calculado: 21
Dê um chute inicial para a raiz: 4
sqrt[21] = 4.582575699086481
```

```
Informe o valor a ser calculado: 3
Dê um chute inicial para a raiz: 1
sqrt[3] = 1.732050810014727
```

```
Informe o valor a ser calculado: 2
Dê um chute inicial para a raiz: 1.1
sqrt[2] = 1.414213730689758
```

Exercício 11

Disponível em breve...

Exercício 12

Disponível em breve...

Exercício 13

Disponível em breve...

Exercício 14

Disponível em breve...

Exercício 15

Disponível em breve...

Exercício 16

Disponível em breve...

Exercício 17

Disponível em breve...

Exercício 18

Disponível em breve...

Exercício 19

Disponível em breve...

Exercício 20

Disponível em breve...

Exercício 21

Disponível em breve...

Exercício 22

Disponível em breve...

Exercício 23

Disponível em breve...

Exercício 24

Disponível em breve...

Exercício 25

Disponível em breve...

Importante!

Dicas para realizar uma boa prova:

- Resolver e entender os exercícios da Lista.
- Rever os conceitos apresentados durante as aulas.
- Consultar a bibliografia sugerida sobre o assunto quando surgir dúvidas.
- Procurar ajuda se as dúvidas persistirem!

Referências Bibliográficas I

Aguilar, L. J. (2008).

Programação em C++: Algoritmos, Estruturas de Dados e Objetos.
McGraw-Hill, São Paulo.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., e Stein, C. (2002).

Algoritmos: Teoria e Prática.
Elsevier, Rio de Janeiro.

Drozdek, A. (2009).

Estrutura de Dados e Algoritmos em C++.
Cengage Learning, São Paulo.

Forbellone, A. L. V. e Eberspacher, H. F. (2005).

Lógica de Programação: A Construção de Algoritmos e Estrutura de Dados.
Pearson Prentice Hall, São Paulo, 3 edition.

Knuth, D. E. (2005).

The Art of Computer Programming.
Addison-Wesley, Upper Saddle River, NJ, USA.

Pinheiro, F. d. A. C. (2012).

Elementos de Programação em C.
Bookman, Porto Alegre.

Referências Bibliográficas II

Sedgewick, R. (1998).

Algorithms in C: Parts 1-4, Fundamentals, Data Structures, Sorting, Searching.
Addison-Wesley, Boston, 3rd edition.

Szwarcfiter, J. L. e Markenzon, L. (1994).

Estruturas de Dados e Seus Algoritmos.
LTC, Rio de Janeiro.

Tenenbaum, A. A., Langsam, Y., e Augenstein, M. J. (1995).

Estruturas de Dados Usando C.
Makron Books, São Paulo.