Universidade Federal do Ceará – Departamento de Engenharia de Teleinformática
Atividade Didática Remota – 2020.1
Introdução à Programação
Prof. Tarcisio Ferreira Maciel, Dr.-Ing.

Objetivos:

- Explorar programas que utilizem funções.
- Explorar programas que utilizem funções recursivas.
- Explorar programas que utilizem uma biblioteca de funções próprias.

Comece criando uma aplicação do tipo "Console application" seguindo o passo-a-passo descrito em laboratório. Para questões que não exigem a criação de uma função específica, um projeto independente denominado **QuestaoXX** pode ser criado, onde **XX** indica o número da questão. Para questões que exigem a criação de uma ou mais funções, as mesmas devem ser fazer parte de uma biblioteca própria de funções composta por dois arquivos: i) o arquivo de declarações que deverá se chamar **`funcoes.hpp`** e; ii) pelo arquivo de implementação **`funcoes.cpp`**. Nesse caso, todas as questões devem ser solucionados na mesma **`main`** com uma mensagem `"Questao XX -> Inicio"` e uma mensagem `"Questao XX <- Fim"` impressas usando **`cout`** no início e no fim de cada questão, respectivamente. O termo **XX** corresponde ao número da questão. As questões que não puderem ser solucionadas no horário do laboratório deverão ser consideradas exercícios a serem solucionados em casa.

1 Funções

Exercício 1. (L03) Faça uma função **`mapMark`** que recebe a média final de um aluno por parâmetro e retorna o seu conceito, conforme a tabela abaixo:

Nota	Conceito
0,0 a 4,9	D
5,0 a 6,9	C
7,0 a 8,9	B
9,0 a 10,0	A

Na função **`main`**, utilize a referida função para determinar e imprimir o conceito de um aluno cuja média foi fornecida via teclado.

Exercício 2. (L03) [1, Número perfeito] Um número inteiro positivo **`n`** é chamado perfeito se a soma de seus divisores, excluindo-se ele mesmo, é igual a ele. Por exemplo, o número 6 é perfeito pois $1 + 2 + 3$ é igual a 6. Escreva uma função chamada **`perfeito`** que retorna verdadeiro se o número **`n`** fornecido como parâmetro for perfeito, e retorna falso caso contrário. Na função **`main`**, leia um número $n > 0$ e use a função **`perfeito`** para determinar n é ou não perfeito. Na função **`main`**, imprima uma mensagem adequada para cada um dos dois casos.

Exercício 3. (L03) Escreva uma função **`collatz`** que retorna a soma dos termos da sequência de Collatz para um número inteiro positivo n fornecido como parâmetro. Os termos a_i (em que $i = 1, 2, 3, \ldots$) da sequência gerada para o número n são obtidos da seguinte forma:

$$a_i = \begin{cases} n, & \text{se } i = 1, \text{ (primeiro termo)} \\ \dfrac{a_{i-1}}{2}, & \text{se } a_{i-1} \text{ é um número par, (termo anterior é par)} \\ 3 \cdot a_{i-1} + 1, & \text{se } a_{i-1} \text{ é ímpar, (termo anterior é ímpar)} \end{cases}$$

A sequência acaba quando a_i se torna 1 e o primeiro termo da sequência é o próprio número n. Por exemplo: se $n = 2$ a sequência é 2 1 e a soma é 3; se $n = 5$ a sequência é 5 16 8 4 2 1 e soma é 36; se $n = 3$, a sequência é 3 10 5

16 8 4 2 1 e a soma é 49. Na função **main**, leia 5 números válidos e utilizando a função **collatz**, imprima na função **main** a soma dos termos da sequência de Collatz para cada um dos 5 números lidos.

Exercício 4. (L03) Faça uma função **isPrime** que recebe por parâmetro um valor inteiro **n** e positivo e retorna o valor lógico Verdadeiro caso o valor seja primo e Falso em caso contrário. Na função **main**, utilize a referida função para verificar e imprimir uma mensagem se um número **x** lido via teclado é ou não primo.

Exercício 5. (L03) [2] Quando toma-se dinheiro emprestado, tipicamente o empréstimo é pago através do pagamento de parcelas. Suponha que o valor do empréstimo é L, r é taxa de juros ao ano, m é número de parcelas por ano e que o empréstimo é por t anos, e que $i = \frac{r}{m}$. Então o valor da parcela é

$$R = \frac{L \cdot i}{1 - (1+i)^{-mt}}.$$

O balanço do empréstimo após k parcelas pagas pode ser expresso como

$$L' = R\left(\frac{1 - (1+i)^{-(mt-k)}}{i}\right).$$

Escreva uma função **pmt** que recebe L, r, m e t e que calcula e retorna o valor da parcela do empréstimo. Escreva uma função **balanco** que recebe R, i, m, t e k e que calcula e retorna o valor do balanco do empréstimo. Na função **main** leia os valores de L, r, m, t e k e calcule e imprima o valor da parcela e o balanco do empréstimo após k parcelas pagas.

Exercício 6. (L06) [1, Q. 6.2] Crie uma função que calcule o fatorial de um número n e utilizando essa função escreva uma outra função que calcule e retorne o valor de S dado por

$$S = \frac{1!}{2!} + \frac{3!}{4!} + \frac{5!}{6!} + \frac{7!}{8!} + \ldots$$

utilizando os N primeiros termos da série, onde $N \geq 1$ é fornecido pelo teclado. Na função **main**, leia o valor de N e utilize a função para calcular S. Imprima o resultado na função **main**.

Exercício 7. (L05) Escreva uma função **coprime** que determine se dois valores inteiros e positivos **A** e **B** fornecidos como parâmetros são primos entre si, retornando o valor lógico verdadeiro ou falso de acordo com o caso. Dois números inteiros são ditos primos entre si, caso não exista divisor comum (diferente de 1) aos dois números. Por exemplo, 10 (divisível por 1, 5 e 10) e 9 (divisível por 1, 3 e 9) são primos entre si. Na função **main**, leia dois números e utilize a função **coprime** para determinar se os números são primos entre si e imprima na **main** uma mensagem adequada a cada caso.

Exercício 8. (L03) Faça uma função **roots** que recebe por parâmetro os valores necessário para o cálculo da fórmula de báskara e retorna, também por parâmetro, as suas raízes, caso seja possível calcular. A função deve retornar o valor lógico Verdadeiro caso as raízes sejam reais e o valor lógico falso caso as raízes sejam complexas. Na função **main**, utilize a referida função para calcular e imprimir as raízes ou uma mensagem de erro de acordo com dados fornecidos pelo usuário via teclado.

2 Funções recursivas

Exercício 9. (L04) O máximo divisor comum dos inteiros **x** e **y** é o maior inteiro que é divisível por **x** e **y**. Escreva uma função recursiva **mdc** em C++ que retorna o máximo divisor comum de **x** e **y**. O mdc de **x** e **y** é definido como segue: se **y** é igual a 0, então **mdc(x,y)** é **x**; caso contrário, **mdc(x,y)** é **mdc(y, x % y)**, em que % é o operador resto.

Exercício 10. Os números tribonacci são definidos pela seguinte recursão

$$f(N) = \begin{cases} 0, & \text{se } N = 0 \\ 0, & \text{se } N = 1 \\ 1, & \text{se } N = 2 \\ f(N-1) + f(N-2) + f(N-3), & \text{se } N > 3 \end{cases}$$

Faça uma função recursiva que receba um número **N** e retorne o **N**-ésimo termo da sequência de tribonacci. Na função **main**, leia um valor para **N**, utilize a função para calcular o **N**-ésimo termo da série e o imprima na função **main**.

Exercício 11. A sequência de Padovan é uma sequência de naturais $P(N)$ definida pelos valores iniciais

$$P(0) = P(1) = P(2) = 1$$

e a seguinte relação recursiva

$$P(N) = P(N-2) + P(N-3) \text{ se } N > 3.$$

Alguns valores da sequência são: 1, 1, 1, 2, 2, 3, 4, 5, 7, 9, 12, 16, 21, 28, ... Faça uma função recursiva que receba um número **N** e retorne o **N**-ésimo termo da sequência de Padovan. Na função **main**, leia um valor para **N**, utilize a função para calcular o **N**-ésimo termo da série e o imprima na função **main**.

Exercício 12. (L06) Escreva uma função recursiva **sumDigits** que soma os dígito de um número inteiro não-negativo **n** fornecido à mesma como parâmetros. Por exemplo, se o número fornecido à função for 762021192, o resultado será 7+6+2+0+2+1+1+9+2 = 30. Na função **main**, leia um valor para **n**, valide os dados de entrada, e aplique a referida função para calcular a soma dos dígitos de **n**. Imprima o resultado na **main**.

Exercício 13. (L06) Faça uma função recursiva que imprime um número inteiro **N** com seus dígitos invertidos. Ex: 123 <-> 321. Na função **main**, leia os dados de entrada necessários para aplicação da função, aplique a mesma e imprima os resultados obtidos.

Referências

[1] M. A. F. de Souza, M. M. Gomes, M. V. Soares, and R. Concilio, *Algoritmos e lógica de programação*, 1st ed. Cengage Learning, 2008.

[2] D. S. Malik, *C++ programming: from problem analysis to program design*, 6th ed. Cengage Learning, 2013.