UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL
IMD0030 - Linguagem de Programação I - T03 – 2016.1
Professor: Silvio Costa Sampaio

Lista de exercícios II: Funções e Recursividade; Ponteiro genérico e ponteiro de função; e Sobrecarga de Função

Notas:
1. Para cada questão de implementação em C++:
   1.1. Separe o código (*.cpp) das definições das funções, estruturas e constantes (*.h);
   1.2. Crie sempre um arquivo main.cpp e utilize a função principal para demonstrar o funcionamento da sua implementação. Sempre que possível, crie um arquivo de casos de teste (case.txt) e o utilize para testar exaustivamente sua implementação;
   1.3. Faça uso das boas práticas de programação que você já conhece.
   1.4. O código deverá ser compilado e devidamente testado.

Q01. Desenvolva funções recursivas como solução para os seguintes problemas e indique qual tipo de recursão está sendo utilizada em cada um dos algoritmos:

a. Dado um valor ***N***, calcular o valor da sequência:

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{N}$$

b. Dado um valor ***n***, calcular o valor da sequência:

$$\frac{2}{4} + \frac{5}{5} + \frac{10}{6} + \frac{17}{7} + \frac{26}{8} + \cdots + \frac{(n^2+1)}{(n+3)}$$

Q02. Converta as funções anteriores para uma versão iterativa.

Q03. Escreva um programa que contenha uma função sobrecarregada chamada **min** para determinar o menor de dois parâmetros. Teste o seu programa usando pares de números inteiros, de caracteres e de números em ponto flutuante.

Q04. Analise o código apresentado a seguir, e responda:
a) Qual o valor final (impresso) para a variável **arg1** na função **main()**? Se houve alguma alteração de seu valor inicial, em que momento isso ocorreu? Justifique, inclusive se o valor não foi alterado.
b) Qual o valor final (impresso) para a variável **arg2** na função **main()**? Se houve alguma alteração de seu valor inicial, em que momento isso ocorreu? Justifique, inclusive se o valor não foi alterado.
c) Considerando a execução, quais os valores de **a** e **b** no final das funções **funcaoA**, **funcaoB**, **funcaoC**?

```cpp
#include <iostream>

using namespace std;

void funcaoA (int &a, int b);
void funcaoB (int a, int b);
void funcaoC (int a, int &b);

void funcaoA (int &a, int b)
{
    a++;
    b++;
    funcaoB (a,b);
}

void funcaoB (int a, int b)
{
    a++;
    b++;
    funcaoC(a,b);
}

void funcaoC (int a, int &b)
{
    a++;
    b++;
}
```

```cpp

int main(int argc, char* argv[])
{
    int arg1 = 13;
    int arg2 = 15;

    funcaoA (arg1,arg2);
    cout << arg1 << " --- " << arg2 << endl;

    return 0;
}
```

Q05. Escreva um programa que contenha a função sobrecarregada ***somatório()*** que atenda aos seguintes critérios:

- Caso receba apenas um vetor de inteiros ***v*** e seu tamanho, deverá retornar o resultado da expressão: $v_1 + v_2 + v_3 + \cdots + v_n$

- Caso receba, além do vetor de inteiros ***v*** e seu tamanho, um terceiro parâmetro inteiro ***x***, deverá retornar o resultado da expressão:
  $\frac{v_1}{x} + \frac{v_2}{2x} + \frac{v_3}{3x} + \cdots + \frac{v_n}{nx}$

- Caso receba, além do vetor de inteiros ***v*** e seu tamanho, do parâmetro ***x***, um quarto parâmetro inteiro ***y***, deverá retornar o resultado da expressão:
  $\frac{v_1}{x-y} + \frac{v_2}{2x-y} + \frac{v_3}{3x-y} + \cdots + \frac{v_n}{nx-y}$.
  Considere ***y*** como tendo o valor padrão (default) de 0 (zero).

Q06. Escreva uma função chamada quadrado que mostre, na margem esquerda da tela, um quadrado de caracteres cujo lado é especificado por um parâmetro do tipo inteiro chamado ***lado***. Em caso de omissão do caractere a ser usado, deverá ser utilizado o caractere * (asterisco) como padrão.

Assim, a chamada **quadrado(4)** produziria a saída:

```
****
****
****
****
```

enquanto que a chamada **quadrado(4, 'a')** :

```
aaaa
aaaa
aaaa
aaaa
```

Q07. Escreva uma função para manipulação de um vetor de inteiros com a seguinte assinatura: ***int faztudo_vetor (int colecao [ ], int tamanho, <tipo de operação> operação, int value = 0);***
Onde, o tipo de operação deve ser implementado como uma enumeração para indicar as operações:

opMax – retorna o maior valor presente na coleção
opMin – retorna o menor valor presente na coleção
opSum – retorna a soma de todos os valores presentes na coleção
opAvg – retorna o valor médio (a parte inteira) de todos os valores presentes na coleção.
opHig – retorna a quantidade de elementos no vetor com valor superior a ***value***
opLow – retorna a quantidade de elementos no vetor com valor inferior a ***value***

Implemente cada operação como uma função. A chamada da função referente à operação deve ser realizada através de um ponteiro de função.

Exemplo: para um vetor v = [12, 4, 60, 5, 23] (tamanho => MAX = 5),

faztudo_vetor (v, MAX, opMax) => 60 (o maior valor no vetor **v** é 60)
faztudo_vetor (v, MAX, opMin) => 4 (o menor valor no vetor **v** é 4)
faztudo_vetor (v, MAX, opSum) => 109 (a soma de todos os valores do vetor **v**)
faztudo_vetor (v, MAX, opAvg) => 21 (a parte inteira da média 21,8)
faztudo_vetor (v, MAX, opLow, 15) => (há apenas 3 valores menores do que 15)
faztudo_vetor (v, MAX, opHig, 25) => 1 (há apenas 1 valor maior do que 25)

Q08. Sobrecarregue a função ***faztudo_vetor*** da Q07 para permitir a operação sobre um vetor de reais (double). Considere ainda que a operação padrão (default) deve ser opMax.

Q09. Que valores serão impressos pelo programa listado abaixo? Para cada resultado impresso pelo programa, justifique a sua resposta, baseado nos conceitos discutidos em sala de aula.

```cpp
#include <iostream>
#include <bitset>

using namespace std;

int somaA (int a, int b)
{
    a++;
    int result = a + b;
    return result;
}

int somaB (int &a, int b)
{
    a++;
    int result = a + b;
    return result;
}


void somaC (int a, int b, int *result)
{
    a++;
    (*result) += a + b;
}


void somaD (int a, int b, int &result)
{
    a++;
    result += a + b;
}

void somaE (int a, int b, int result)
{
    a++;
    result += a + b;
}

int main(int argc, char* argv[])
{
    int arg1 = 5;
    int arg2 = 6;

    cout << somaA(arg1,arg2) << endl;
    cout << somaB(arg1,arg2) << endl;

    int resultado = 0;
    somaC(arg1,arg2,&resultado);
    cout << resultado << endl;

    somaD(arg1,arg2,resultado);
    cout << resultado << endl;

    somaE(arg1,arg2,resultado);
    cout << resultado << endl;

    return 0;
}
```

FIM DA LISTA.