Primeiro Exercício-Programa

Prof. Luciano Antonio Digiampietri

Prazo máximo para a entrega: 11/06/2023

1 Implementação de diversas funções matemáticas

Nesse exercício prático você deverá implementar cinco funções, muitas delas relacionadas a operações matemáticas/aritméticas, mas não poderá usar funções prontas de outras bibliotecas.

Deverá, dessa reforma, implementar suas funções utilizando os operadores aritméticos (por exemplo, $\%, /, *, +, -$), condicionais, laços etc.

Função 1 - ***int somaDigitos(int valor)*** : esta função deverá receber um número inteiro como parâmetro (*valor*). Se este número for menor ou igual a zero ou maior que 99999, a função deverá retornar o valor -1. Caso contrário, a função deverá somar os dígitos do número de acordo com suas unidades, dezenas, centenas, milhares e dezenas de milhares e retornar esta soma. Por exemplo, para o número 2345, deve retornar 14 que corresponde a $2+3+4+5$.

Função 2 - ***int somaIntervalo(int inicio, int fim)*** : esta função deverá receber dois números inteiro (*inicio* e *fim*). Se qualquer desses números for negativo ou nulo a função deverá retornar -1 (apenas como uma exceção e não para representar o resultado da operação); se o valor de *fim* for menor do que o valor de *inicio*, a função também deverá retornar -1. Caso contrário, a função deverá retornar a soma de todos os números inteiros de *inicio* até *fim*. Por exemplo, para *inicio*=3 e *fim*=4, a função deverá retornar 7, valor obtido pela soma dos seguintes valores: $3 + 4$. Já para *inicio*=3 e *fim*=6, a função deverá retornar 18, valor obtido pela soma dos seguintes valores: $3 + 4 + 5 + 6$.

Função 3 - ***int somaInteiros(int inicio, int limite)*** : esta função deverá receber dois números inteiro (*inicio* e *limite*). Se qualquer desses números for negativo ou nulo a função deverá retornar -1 (apenas como uma exceção e não para representar o resultado da operação); se o valor de *limite* for menor do que o valor de *inicio*, a função também deverá retornar -1. Caso contrário, a função deverá retornar o maior valor da soma dos inteiros consecutivos a partir de *inicio*, desde que esse valor seja menor ou igual ao valor de *limite*. Por exemplo, para *inicio*=3 e *limite*=5, a função deverá retornar 3, valor obtido apenas pelo número inicial: 3 (pois se somarmos 3 e 4 a soma será 7 que é maior do que o limite). Já para *inicio*=4 e *limite*=17, a função deverá retornar 15, valor obtido pela soma dos seguintes valores: $4+5+6$ (se, adicionalmente, somarmos 7, o valor ultrapassaria o valor de *limite*).

Função 4 - ***double dividindoPorDois(double valor, int divisoes)*** : esta função deverá receber um número real (*valor*) e um número inteiro (*divisoes*). Se o parâmetro *divisoes* for nulo ou negativo, a função deverá retornar -1. Caso contrário, a função deverá retornar o resultado de $valor/2^{divisoes}$ (isto é, *valor* dividido por dois elevado a *divisoes*). Este resultado deverá ser obtido a partir de sucessivas divisões. Por exemplo, para *valor*=8 e *divisoes*=4, a função deverá retornar 0.5, valor obtido pela seguinte conta: 8/2/2/2/2.

Função 5 - ***double valorDePI(double limiar)*** : Existem diversas funções computacionais para calcular valores aproximados de diferentes constantes ou funções. A função a seguir computa uma aproximação para o valor de pi[1]. Esta função recebe o número de iterações e calcula o valor de pi de acordo com este número (quanto maior o número, melhor será a aproximação).

```
void calculoDoValorDePI(int iteracoes){
    double meuPI = 4; // valor inicial de pi ("primeira iteracao")
    int sinal = -1;
    double divisor = 3;
    for (int i=2;i<=iteracoes;i++){
      meuPI += sinal * 4.0/divisor; // atualiza o valor de pi
      divisor += 2; // incrementa o valor do divisor
      sinal *= -1; // inverte o sinal da variavel sinal
    }
    printf("(%i) pi: %f\n", iteracoes, meuPI);
}
```

A função *calculoDoValorDePI* recebe como parâmetro o número de iterações que serão realizadas para o cálculo do valor aproximado de π. Porém, neste exercício você deve implementar uma nova função: *double valorDePI(double limiar)*, baseada na ideia da função acima, porém que, ao invés de executar um número de iterações passado como parâmetro, irá receber como parâmetro um limiar de precisão que dirá quando sua função deverá parar de calcular a aproximação, de acordo com a seguinte lógica: a cada iteração, você deverá comparar o valor atual de π com o valor anterior (valor da iteração anterior), enquanto essa diferença (em valor absoluto [isto é, sem considerar o sinal]) for maior do que o valor de *limiar*, sua função deverá continuar calculando o valor de π. Assim que essa diferença for menor ou igual ao valor de *limiar*, sua função deverá retornar o valor atual de pi. Observação: se o valor do parâmetro *limiar* for menor ou igual a 0,000001 (um milionésimo), sua função deverá retornar -1, caso contrário deverá retornar o valor aproximado de π conforme explicado. Sua função não deverá imprimir nada na tela.

[1] `https://pt.wikipedia.org/wiki/Pi`

2 Material a Ser Entregue

Um arquivo, denominado *NUSP*.c (sendo NUSP o seu número USP, por exemplo: 123456789.c), contendo o código das cinco funções solicitadas (e qualquer outra função auxiliar que você ache necessário). Para sua conveniência, `completeERenomeie.c` será fornecido, cabendo a você então completá-lo e renomeá-lo para a submissão.

Atenção!

1. Não modifique a assinatura das cinco funções solicitadas neste EP!
2. Nenhuma das funções implementadas deve imprimir algo na tela.
3. Para avaliação, apenas estas cinco funções serão invocadas diretamente. Em especial, qualquer código dentro da função `main()` será ignorado. Então certifique-se de que o problema seja resolvido chamando-se diretamente as funções implementadas.

3 Entrega

A entrega será feita única e exclusivamente via sistema e-Disciplinas, até a data final marcada. Deverá ser postado no sistema um arquivo .c, tendo como nome seu número USP:

`seuNumeroUSP.c  (por exemplo, 123456789.c)`

Não esqueça de preencher o cabeçalho constante do arquivo .c, com seu nome, número USP, turma etc.

A responsabilidade da postagem é exclusivamente sua. Por isso, submeta e certifique-se de que o arquivo submetido é o correto (fazendo seu *download*, por exemplo). Verifique também se você efetivou a submissão. Problemas referentes ao uso do sistema devem ser resolvidos com antecedência. Se, na data limite da submissão, o sistema e-Disciplinas estiver fora do ar, então serão aceitas submissões por e-mail (apenas se o sistema estiver fora do ar e dentro do prazo máximo de submissão).

4 Avaliação

A nota atribuída ao EP será focada nas funcionalidades solicitadas, porém não esqueça de se atentar aos seguintes aspectos:

1. Documentação: se há comentários explicando o que se faz nos passos mais importantes e para que serve o programa (tanto a função quanto o programa em que está inserida);
2. Apresentação visual: se o código está legível, identado etc;
3. Corretude: se o programa funciona.

Além disso, algumas observações pertinentes ao trabalho, que influenciam em sua nota, são:

- Este exercício-programa deve ser elaborado individualmente;
- Não será tolerado plágio;
- Exercícios com erro de sintaxe (ou seja, erros de compilação), receberão nota ZERO.