APE - Algoritmos e Programação Estruturada

Exercícios sobre Modularização

Professor Sandro Teixeira Carvalho

Prática para Laboratório de Computação

Crie cada um dos programas da lista abaixo no IDE **Code::Blocks** ou no IDE **Dev-C++**. Crie uma pasta com o seu nome completo dentro da pasta Documentos do computador do laboratório e salve seus programas nessa pasta. Em seguida, compacte essa pasta em formato zip ou rar e envie a pasta compactada para o professor. **Não envie arquivos com extensão .exe**. Lembre-se de excluir os arquivos com extensão .exe antes de compactar a pasta para enviar para o professor pela plataforma vigente da sua universidade. Envie somente os arquivos com extensão .c para o professor. Como sugestão, para o exercício 01 desta lista, crie um programa com o nome programa01.c e assim por diante.

Você sabe a diferença entre **argumento** e **parâmetro**?

Resposta: argumento é o dado que passamos para uma função. É considerado uma 'mensagem' enviada para uma função. Parâmetro é a variável que está no cabeçalho da função para receber o argumento. No simples programa a seguir, a constante de tipo real 23,45 é o **argumento** que será passado para o **parâmetro** valor da função mostraPrecoVenda.

A palavra-chave void que aparece antes do nome da função mostraPrecoVenda indica que esta função não retorna nenhum dado para a função que a chamou, que no exemplo abaixo é a função main. Por isso a função main possui um int, que indica que ela retorna um inteiro (return 0). Quando uma função possui void dentro de seus parênteses significa que essa função não recebe nenhum argumento, isto é, não possui parâmetro(s).

```
#include <stdio.h>

void mostraPrecoVenda(float valor) {
    printf("%.2f", valor);
}

int main(void) {
    mostraPrecoVenda(23.45);
    return 0;
}
```

Note no programa acima que a função mostraPrecoVenda foi escrita acima da função main. É possível escrever as suas funções abaixo da função main, como no exemplo abaixo. Porém, para isso é necessário declarar a função no início do programa, antes da função main. Veja como isso é possível:

```
#include <stdio.h>

void mostraPrecoVenda(float);  // protótipo da função (declaração da função)

int main(void) {
    mostraPrecoVenda(23.45);
    return 0;
}

void mostraPrecoVenda(float valor) {
    printf("%.2f", valor);
}
```

Bom trabalho!

1) O que é Modularização?

2) Cite quatro vantagens da Modularização.

3) Explique o que é escopo de variável.

4) Quais são os dois escopos de variáveis da linguagem C?

5) O que significa uma variável possuir escopo global? Explique como definimos uma variável com escopo global em linguagem C

6) O que significa uma variável possuir escopo local? Explique como definimos uma variável com escopo local em linguagem C.

7) Explique porque na maioria das vezes colocamos a palavra-chave int imediatamente antes da palavra-chave main em um programa escrito em linguagem C.

8) Explique porque na maioria das vezes colocamos a palavra-chave void logo após a palavra-chave main em um programa escrito em linguagem C.

9) Se você não colocar a palavra-chave int antes da palavra-chave main, o que você deverá fazer em seu programa para que ele não apresente nenhum erro ou advertência? Explique sua resposta.

10) Se você não colocar a palavra-chave void dentro dos parênteses da função main, o que você deverá fazer em seu programa para que ele não apresente nenhum erro ou advertência? Explique sua resposta.

11) Explique a diferença entre erro em tempo de compilação e erro em tempo de execução.

Observações para as questões práticas a seguir:

- A entrada dos dados deve ser realizada pela função principal (main).
- O processamento dos dados deve ser realizado por uma nova função específica.
- A saída da informação (resposta) deve ser realizada pela função principal (main), exceto para a questão 4 que pode ser realizada pela própria função.
- Não utilize variáveis com escopo global, exceto se o enunciado exigir/permitir.

1) Escreva um programa modularizado em linguagem C que apenas imprima o seu nome completo utilizando uma função. Basta que a função main inicie chamando uma outra função que apenas imprima seu nome no vídeo.

2) Escreva um programa modularizado em linguagem C que crie uma variável x na função main inicializando esta variável com o número inteiro 5. Em seguida, a função main deve chamar uma outra função que deve receber o valor de x e imprimir esse valor.

3) Escreva um programa modularizado em linguagem C que leia um número inteiro e chame uma outra função para imprimir este número.

4) Escreva um programa modularizado em linguagem C que leia dois números inteiros e chame uma outra função para calcular a soma desses números. A impressão deve ser feita pela função main.

5) Escreva um programa modularizado em linguagem C que leia um número inteiro e verifique e imprima se este número é par ou ímpar.

6) Escreva um programa modularizado em linguagem C que leia três números inteiros e imprima o maior deles.

7) Escreva um programa modularizado em linguagem C que leia um número inteiro positivo e, através de uma função, imprima todos os números inteiros desde o número 1 até o número lido.

8) Escreva um programa modularizado em linguagem C que leia uma palavra e um número inteiro positivo e imprima a palavra lida esse número de vezes utilizando uma estrutura de repetição à sua escolha. A impressão da palavra deve ser feita por uma função.

9) Escreva um programa modularizado em linguagem C que leia dois números inteiros formando um intervalo, por exemplo, o usuário digita o número 5 e depois digita o número 20. O primeiro número deve ser inferior ao segundo número, caso contrário a mensagem “Intervalo incorreto” deve ser apresentada ao usuário e o programa deve ser encerrado. Se o intervalo estiver correto, o programa deve imprimir todos os números do intervalo. Crie uma função para ler um único número inteiro. A validação dos números lidos deve ser feita por uma função. A impressão dos números deve ser feita por uma função. Cada processo deve ser feito por uma função diferente.

10) Escreva um programa modularizado em linguagem C que preencha um array unidimensional homogêneo de tamanho 1000 com inteiros aleatórios de 1 até 10 mil. Em seguida, imprima o array e calcule e imprima o somatório de todos os inteiros contidos no array. Cada processamento/tarefa, como o preenchimento do array, a impressão e o cálculo do somatório deve ser feito cada um por uma função diferente. Somente o array deve ser declarado com escopo global. A parametrização de arrays será estudada posteriormente na disciplina ***Estruturas de Dados***.

11) Escreva um programa em linguagem C que leia as três notas de um aluno de uma universidade, calcule a sua média aritmética simples e imprima uma mensagem conforme a tabela abaixo. A leitura de uma nota e o cálculo da média devem ser feitos cada um por uma função diferente. A nota lida deve ser validada (nova válida de 0,0 até 10,0).

Média	**Mensagem**
0,0 até 3,0	Reprovado
3,1 até 6,9	Exame
7,0 até 10,0	Aprovado

12) Nesta questão você aplicará a técnica de Modularização em um programa que já foi feito em lista prática anterior. Segue o enunciado da questão novamente para simples conferência:

Escreva um programa em linguagem C que inicie mostrando o menu abaixo para o usuário:

```
Programa Calculadora Simples

(1) Somar dois números inteiros
(2) Multiplicar dois números inteiros
(3) Dividir dois números inteiros
(4) Calcular a raiz quadrada de um número inteiro
(0) Encerrar o programa

Sua opção: _
```

Assim que o usuário escolher a opção do menu desejada, o programa deverá executar a operação correspondente do menu.

A seguir, as modificações a serem feitas no programa:

a) crie uma função para imprimir o menu.
b) crie uma função para o algoritmo de cada opção do menu.
c) crie uma função específica para a leitura de um número inteiro, reduzindo a redundância de código.