```
        fim repita
        N ← I - 1
                        {Para a disciplina Geometria Analítica:}
        DISCIPLINA(NMAX,N,FM[1],NUM,NOTAG,FREQG)
                        {Para a disciplina Programação de Computadores}
        DISCIPLINA(NMAX,N,FM[2],NUM,NOTAP,FREQP)
                        {Para a disciplina Cálculo I:}
        DISCIPLINA(NMAX,N,FM[3],NUM,NOTAC,FREQC)
fim_algoritmo
```

3.3. CONSIDERAÇÕES SOBRE A MODULARIZAÇÃO DE PROGRAMAS

Existem várias vantagens e, naturalmente, algumas desvantagens na modularização de programas. Algumas destas desvantagens são específicas de cada linguagem de programação que se escolha para a implementação do algoritmo.

Dentre as vantagens da modularização de programas podem-se citar:

1. Partes comuns a vários programas ou que se repetem dentro de um mesmo programa, quando modularizadas em uma sub-rotina ou função, são programadas e testadas uma só vez, mesmo que tenham que ser executadas com variáveis diferentes.
2. Podem-se constituir bibliotecas de programas, isto é, uma coleção de módulos que podem ser usados em diferentes programas sem alteração e mesmo por outros programadores. Por exemplo, as funções pré-definidas das linguagens de programação, como citadas no item anterior, constituem uma biblioteca de módulos.
3. A modularização dos programas permite preservar na sua implementação os refinamentos obtidos durante o desenvolvimento dos algoritmos.
4. Economia de memória do computador, uma vez que o módulo é armazenado uma única vez, mesmo que utilizado em diferentes partes do programa. Permite, também, que, em um determinado instante da execução do programa, estejam na memória principal apenas o módulo ou os módulos necessários à execução desse trecho de programa.

Dentre as desvantagens pode-se citar que existe um acréscimo de tempo na execução de programas constituídos de módulos, devido ao tratamento adicional de ativação do módulo etc.

Conclui-se que, sem dúvida, a modularização dos programas é uma técnica altamente recomendável, tanto pela eficiência no projeto e desenvolvimento dos mesmos quanto pela quantidade, confiabilidade e manutenção do produto elaborado.

Finalmente, o critério de modularização, utilizado nos exemplos apresentados, foi o de "funcionalidade", isto é, a tarefa de construção do algoritmo foi dividida em partes, segundo a função que estas teriam de executar. Entretanto, existem outros critérios para modularização. Dentre eles, pode-se citar o "ocultamento da informação" [Parnas, 1972], que geralmente produz algoritmos cuja manutenção posterior é mais efetiva.

Estes e outros detalhes, referentes à modularização, são tratados de forma mais adequada na disciplina denominada "Engenharia de Programas" [STAA, 1983], cujo conteúdo foge ao escopo deste texto.

3.4. EXERCÍCIOS PROPOSTOS

PROBLEMAS GERAIS

△ **3.4.1.** Refazer o exercício proposto em 1.12.18, do capítulo 1, usando uma função para calcular o número de pontos de uma etapa. Esta função deve ser ativada para se obter os pontos de cada etapa para cada equipe.

△ **3.4.2.** Reescrever o algoritmo do exemplo 1.41, do capítulo 1, sendo que o cálculo da potência de cada cômodo deve ser feito por uma função.

▲ **3.4.3.** A avaliação de aproveitamento de uma certa disciplina é feita através de 4 provas mensais no valor de 20 pontos e uma prova final no valor de 40 pontos. A nota final é obtida somando-se as 3 melhores notas, dentre as provas mensais, com a nota da prova final.

O conceito final é dado atendendo-se ao seguinte critério:

de 90 a 100 — conceito A
de 80 a 89 — conceito B
de 70 a 79 — conceito C
de 60 a 69 — conceito D
de 40 a 59 — conceito E
de 0 a 39 — conceito F

Fazer uma sub-rotina que, recebendo como parâmetro 4 (quatro) números inteiros, devolva ao módulo que a chamou a soma dos 3 (três) maiores números dentre os 4 (quatro) números recebidos.

Fazer um algoritmo que:
- leia um conjunto de 80 linhas contendo, cada uma, o número do aluno, as 4 notas mensais e a nota da prova final;
- calcule, para cada aluno, sua nota final, utilizando a sub-rotina anterior;
- verifique o conceito obtido;
- escreva, para cada aluno, o seu número, a sua nota final e o seu conceito.

△ **3.4.4.** Fazer um algoritmo para um programa de apostas da LOTO. O algoritmo deverá ler, inicialmente, as cinco dezenas sorteadas e, a seguir, ler várias linhas, uma para cada aposta, contendo:
- número da aposta;
- quantidade de dezenas apostadas (no máximo, 10);
- as dezenas apostadas.

A última linha, que não entrará nos cálculos, conterá o número da aposta igual a zero.

O algoritmo deverá escrever o número de todas as apostas que tiverem três, quatro ou cinco dezenas sorteadas e, ao final, a quantidade de apostadores que fizeram o terno (três dezenas sorteadas), a quadra (quatro dezenas sorteadas) e a quina (cinco dezenas sorteadas). Neste algoritmo, deverá ser utilizada uma sub-rotina que faça a avaliação do número de pontos de cada aposta.

▲ **3.4.5.** Construir uma função que receba como parâmetro de entrada um número inteiro positivo e devolva um dígito verificador conforme o processo de cálculo descrito no exercício 2.5.3.6.

Escrever um algoritmo capaz de:
- ler um conjunto indeterminado de linhas contendo, cada uma, o nome de uma pessoa e seu número de CPF (n.º de inscrição no Cadastro de Pessoas Físicas).
- imprimir, para cada pessoa, os seus dados de entrada mais a mensagem "VÁLIDO, ou "INVÁLIDO", conforme a situação do número de CPF.
- utilize a função acima para calcular os dígitos verificadores.

Obs: Um n.º de CPF é validado através de seus dois últimos dígitos (dígitos verificadores, denominados *controle*). Por exemplo, o CPF de número 23086025620 é validado pelos dígitos verificadores 20. O esquema de verificação é o seguinte:

$$230860256 \xrightarrow{\text{função}} \text{dígito verificador igual a } 2$$

$$2308602562 \xrightarrow{\text{função}} \text{dígito verificador igual a } 0$$

△ **3.4.6.** Fazer uma sub-rotina que, dados N números, determine o número que apareceu mais vezes. Supor que os valores possíveis de cada número estão entre 1 e 6, inclusive, e que sempre haverá um único número vencedor.

Sabendo-se que um jogo de dados ocorre 40 vezes por dia e que a cada dia é digitada uma linha contendo os 40 números que saíram, fazer um algoritmo que:
- leia os dados contidos em 30 linhas, correspondentes a um mês de jogo;
- determine o número ganhador do dia, utilizando-se a sub-rotina anterior;
- escreva este número e a mensagem "RESULTADO DIARIO";
- verifique também qual o número ganhador do mês;
- escreva este número e a mensagem "RESULTADO MENSAL DO JOGO".

▲ **3.4.7.** Fazer uma sub-rotina, cujo cabeçalho é dado por:

QUANTOSDIAS(DIA,MES,ANO,N)

onde:

DIA, MES e ANO são parâmetros de entrada;

N é um parâmetro de saída que conterá o número de dias do ano até a data fornecida.

cada mês. O mês de fevereiro pode ter 28 ou 29 dias, dependendo de o ANO ser bissexto ou não (ver o exercício 1.12.23).

Fazer um algoritmo que:

- leia um conjunto de linhas contendo, cada uma, duas datas. A última linha, que será utilizada como *flag*, conterá os valores 0, 0, 0, 0, 0, 0;
- verifique se as datas estão corretas (1 <= MES <=12, dia de acordo com o mês e se ambas estão dentro do mesmo ano). Se alguma das datas não estiver correta, escreva "DATA INCORRETA" e os valores de DIA, MES e ANO das duas datas;
- verifique, se as datas estiverem corretas, qual a diferença, em dias, entre essas duas datas;
- escreva as datas lidas e a diferença entre elas.

△ **3.4.8.** Escrever um algoritmo que leia um conjunto de 1.000 linhas contendo, cada uma, uma palavra em inglês e a sua tradução em português. Em seguida, leia um número indeterminado de linhas contendo, cada uma:

- a letra I (indicando inglês) e uma palavra qualquer das 1.000 em inglês; ou
- a letra P (indicando português) e uma palavra qualquer das 1.000 em português.

Para cada uma dessas linhas, escrever a palavra lida e a sua tradução. A última linha, indicando o fim de dados, terá a primeira letra diferente de I e P.

A tradução da palavra lida deve ser feita através de uma sub-rotina que recebe as listas de palavras em inglês e português, a letra I ou P e a palavra que se deseja traduzir, devolvendo a tradução da mesma.

PROBLEMAS DE APLICAÇÃO EM CIÊNCIAS EXATAS

▲ **3.4.9.** Escrever uma função que receba dois números inteiros, positivos, e determine o produto dos mesmos, utilizando o seguinte método de multiplicação:

- dividir, sucessivamente, o primeiro número por 2, até que se obtenha 1 como quociente;
- paralelamente, dobrar, sucessivamente, o segundo número;
- somar os números da segunda coluna que tenham um número ímpar na primeira coluna. O total obtido é o produto procurado.

Exemplo:

	9×6	
9	$6 \rightarrow$	6
4	12	
2	24	
1	$48 \rightarrow$	+ 48
		54

A seguir, escrever um algoritmo que leia 10 pares de números, calcule e escreva os respectivos produtos, usando a função anterior.

▲ **3.4.10.** Determinar os números inteiros, menores que 5.000, que são quadrados perfeitos e, também, são capicuas.

Capicuas são números que têm o mesmo valor se lidos da esquerda para a direita ou da direita para a esquerda. Exemplo: 44, 232, 1661, etc.

Deverão ser escritos os seguintes algoritmos:

- um módulo principal;
- uma função que calcule quantos algarismos tem um determinado número inteiro;
- uma sub-rotina para separar um número em n algarismos;
- uma sub-rotina para formar o número na ordem inversa.

△ **3.4.11.** Dado um polinômio na forma

$$P = a_1x^n + a_2x^{n-1} + \ldots + a_nx + a_{n+1}$$

a) Fazer uma sub-rotina que retorne o valor do polinômio e o de sua derivada no ponto x, recebendo como parâmetros de entrada a ordem do polinômio, os coeficientes e o x.

b) Fazer um algoritmo que:

- leia a ordem do polinômio e os seus coeficientes;
- utilizando a sub-rotina da alínea a, calcule o valor do polinômio e o de sua derivada para diversos valores de x. Esses valores deverão estar digitados um por linha, sendo que a primeira linha desse conjunto de dados contém o número de valores de x a serem lidos;

△ **3.4.12.** Escrever uma sub-rotina que calcule a distância entre dois vetores:

$$X = [x_1, x_2, \ldots, x_n] \text{ e } Y = [y_1, y_2, \ldots, y_n]$$

Escrever um algoritmo que, utilizando a sub-rotina anterior, calcule e escreva a distância entre M pares de vetores. O valor de M será fornecido.

▲ **3.4.13.** Calcular e escrever a área total de 10 tetraedros, dadas as coordenadas de cada um de seus quatro vértices. Para tanto, deverão ser utilizadas as seguintes sub-rotinas:

a) que calcula a distância entre dois pontos do espaço;

b) que calcula a área de um triângulo em função de seus lados (AREA = $\sqrt{p \times (p-a) \times (p-b) \times (p-c)}$, onde p é o semi-perímetro do triângulo).

△ **3.4.14.** Escrever uma sub-rotina que calcule o valor de π através da série

$$S = 1 - \frac{1}{3^3} + \frac{1}{5^3} - \frac{1}{7^3} + \ldots \qquad \text{, sendo } \pi = \sqrt[3]{32 \times S}$$

Deverá ser fornecido à sub-rotina o número de termos da série para o cálculo de π.

Escrever um algoritmo que, fornecendo à sub-rotina, sucessivamente, o número de termos (1, 2, 3, ..., N), escreva uma tabela com o valor de π e número de termos utilizados. O valor de N deverá ser lido.

▲ **3.4.15.** Escrever uma sub-rotina que calcule o valor de e através da série:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots$$

O número de termos da série deverá ser fornecido à sub-rotina como parâmetro.

Escrever um algoritmo que, utilizando a sub-rotina anterior, determine o número de termos da série necessário para calcular o valor de e, cuja diferença em relação ao valor obtido através da função EXP(1) seja menor que 0,0001.

△ **3.4.16.** Escrever uma sub-rotina que determine o conjunto interseção entre dois conjuntos A e B de caracteres.

Escrever uma sub-rotina que determine o conjunto união entre esses mesmos conjuntos A e B.

Escrever um algoritmo que leia 50 pares de conjuntos de 100 caracteres cada, determine e escreva a interseção e a união desses conjuntos, utilizando as sub-rotinas anteriormente definidas.

▲ **3.4.17.** Segundo a conjectura de Goldbach, qualquer número par, maior que 2, pode ser escrito como a soma de dois números primos.

Exemplo:

$$8 = 3 + 5,\ 16 = 11 + 5,\ 68 = 31 + 37 \text{ etc.}$$

Dado um conjunto de números inteiros positivos, pares, fazer um algoritmo que calcule, para cada número, um par de números primos cuja soma seja igual ao próprio número. Adotar como *flag* um número negativo.

Para verificar se um número é primo, fazer uma sub-rotina que deverá retornar em uma variável lógica o valor verdadeiro, se o número for primo, e falso, em caso contrário.

△ **3.4.18.** Fazer uma sub-rotina que, recebendo como parâmetro dois conjuntos de números inteiros e os tamanhos destes conjuntos, devolva ao módulo principal um outro conjunto de números inteiros, contendo a interseção dos dois conjuntos recebidos e o tamanho desse novo conjunto formado.

Fazer uma sub-rotina que, recebendo como parâmetro um número inteiro, devolva ao módulo principal um conjunto de números inteiros, contendo todos os divisores do número recebido e o tamanho desse conjunto.

Fazer um algoritmo que:

- leia um conjunto de 30 pares de números inteiros;
- escreva, para cada par de números lidos, os seus valores e os seus divisores comuns, fazendo uso das sub-rotinas anteriormente definidas.