SISTEMA DE PROGRAMAÇÃO FORTRAN II-D

MAXIMILIAN E. HEHL e CIBAR CACERES AGUILERA

Série INFORMAÇÕES N.º 5
outubro — 1965

INSTITUTO DE ENERGIA ATÔMICA
Caixa Postal 11049 (Pinheiros)
CIDADE UNIVERSITÁRIA "ARMANDO DE SALLES OLIVEIRA"
SÃO PAULO — BRASIL

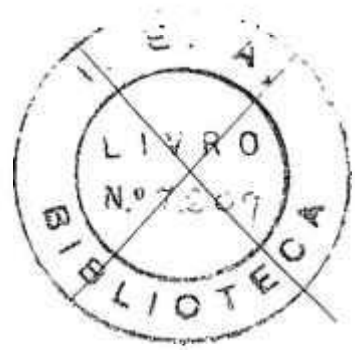
C O M P U T A D O R E S D I G I T A I S

SISTEMA DE PROGRAMAÇÃO

FORTRAN II-D

Maximilian E. Hehl e Cíbar Cáceres Aguilera

SERVIÇO DE CÁLCULO ANALÓGICO E DIGITAL
Instituto de Energia Atômica
São Paulo - Brasil



Informações nº 5

Outubro - 1965

ERRATA DA PUBLICAÇÃO DENOMINADA:

SISTEMA DE PROGRAMAÇÃO FORTRAN - II - D

Página	Linha	Onde se lê	Deve-se ler
I	19	Comandos Aritmético	Comando Aritmético
III	30	Compilação	Compilação 113
IV	2	Registro de contrôle	Registros de contrôle
IV	11	Bibliografia 130	Bibliografia 135
1	17	de agrimensor	do agrimensor
4	6-7	são obtidos	são fornecidos
7	6	binarycoded	binary-coded
9	17	mneumônica	mnemônica
9	20	mneumônicos	mnemônicos
9	24-25	mneumônica	mnemônica
9	25	linguagem básica	linguagem absoluta
9	28-29	"Compilador SPS"	"Montador SPS"
10	20-21-22	1009 (C) com ...	1001 (A) e guarde o resultado em 1001 (A+B). A 2a. instrução indica: segunda linha (020); some o que está na posição 1009 (C) com o que está na posição 1001 (A+B) e guarde o resultado em 1001 (A + +B+C = Y).
10	26	edição	adição
11	21	compilador SPS	montador SPS
14	14	na parte decimal	na forma decimal
14	14-15	na parte inteira	na forma inteira
16	13	$-.45670000 \times 10^{-4}$	$.45670000 \times 10^{-4}$
16	14	0.000000000002	-0.000000000002

Página	Linha	Onde se lê	Deve-se ler
18	21	na parte inteira	na forma inteira
18	21	na parte decimal	na forma decimal
21	19	A*B/C*D**E*F-G	A*B/C*D**E*F-G
21	21	A*(B/C)*(D^{E} *F)-G	A*(B/C)*(D^{E} **F)-G
21	25	representa *	representa **
22	8	((A*B)/C) **2	((A*B)/C) **2
22	9	((A*B)/C) **2.5	((A*B)/C) **2.5
22	14	(2*J*K)**1.5	(2**J*K) **1.5
22	16	X**Y**Z	X**Y**Z
23	12	Q1=3.0*B	Q1=3.0**B
25	5	X*A*3.1416/(2.*Z)**2	X*A*3.1416/(2.**Z)**2
27	16	A**L*4. *X	A**L*4.**X
27	21	RAIZQ=(A**L*4.*X)**.5	RAIZQ=(A**L*4.**X)**.5
30	14	RAIZQ=(A**L*4.*X)**.5	RAIZQ=(A**L*4.**X)**.5
30	20	a * 4 x	a^{1} * 4 x
31	3	Format	FORMAT
32	29	(1 = 2)	(I = 2)
33	27	5.10.-Especificação	5.10.- Especificações
35	4-5-6	ACCEPT 1,K DO 5 I=1,K 5 READ 10,K,A(I)	ACCEPT 1,K DO 5 I=1,K 5 READ 10,A(I)
35	12	READ 9 (A(I),I=1,5)	READ 9,(A(I),I=1,5)
39	13	Voltamos ao exercício	Voltemos ao exercício
39	17	RAIZQ=(A**L*4.*X)**.5	RAIZQ=(A**L*4.**X)**.5
42	28	(1HO,F8.2/1H, E14.8)	(1HO,F8.2/1H , E14.8)
44	26	ABC,	A, B, e C
44	27	*32,548	*32.548
45	12	$\|S(S-A)(S-B)(S-C)\|^{2}$	$\left[S(S-A)(S-B)(S-C)\right]^{2}$
45	22	*32.584	*32.548
47	2	V_s	V's
48	24	p 0	p>0

Página	Linha	Onde se lê	Deve-se ler
52	24	(Link),	(Link)),
54	7	$(N_1$ 10)	$(N_1 \leqslant 10)$
54	9	$(N_1$ 10)	$(N_1 > 10)$
65	2	maior que m_2	maior ou igual a m_2
67	26	$x = \frac{y^2 + z^2}{c}$ x a	$x = \frac{y^2 + z^2}{c}$. a
68	11	subtraia 2 de	subtraia 2 π de
68	12	menor que 2 .	menor que 2 π .
68	14	Se 0.999 x 1.001	Se $0.999 \leqslant x \leqslant 1.001$
76	15	12 CALC=1.+SQRTF(1.+X**X)	12 CALC=1.+SQRTF(1.+X*X)
77	13-14	com resultado	como resultado
77	20	maiores	menores
78	7	EIG=DIAGPR(X,JACK)**2	EIG=DIAGPR(X,JACK)**2
78	18	de 6 caracteres	de até 6 caracteres
84	9	se x d	se $x < d$
84	11	se x d	se $x > d$
90	4	$B(I)= \sum_{J=1}^{5} \lvert A(I,J)*X(J) \rvert$	$B(I)= \sum_{J=1}^{5} \left[A(I,J)*X(J) \right]$
90	10	A(100.J)*X(J)	A(100,J)*X(J)
92	25	$AREA= \int_{XI}^{X1000} y\ d\ x = \frac{H}{3}(y_1 + 4y_a + \ldots$	$AREA= \int_{X1}^{X1000} y\ dx = \frac{H}{3}(y_1 + 4y_2 + \ldots$
97	9	SITR(20,220)	1SITR(20,220)
104	13-14	perfure se valor	perfure seu valor
105	14	var_a de 0 à 500	ω varia de 0 à 500
112	19	formulários utilizando	formulários utilizados
113	11	só processa número	só processa códigos numéricos
116	26	POBJ	POBJP
116	28	*POBJ	*POBJP

Página	Linha	Onde se lê	Deve-se ler
121	21	tabelas 3 e 4	tabelas III e IV
129	10	Zero a uma potência	Zero à uma potência
129	12	fixo a potência	fixo à potência
129	15	flutuante a potência	flutuante à potência
129	19	Zero a potência	Zero à potência

.....

ÍNDICE

CAPÍTULO I

Introdução histórica

Cuidadosamente, a natureza forneceu aos nossos mais antigos ancestrais o modo mais simples de cálculo - os dedos - isto é, um computador digital no sentido estrito da palavra. Ainda nos dias de hoje, vemos em qualquer sala de aula a geração mais nova se utilizando de seus 10 dedos para efetuar qualquer operação aritmética. Involuntária ou voluntàriamente foi estabelecido o sistema de base decimal, com 10 caracteres, com os quais a raça humana expressa seus pensamentos numéricos.

A família de computadores tem se desenvolvido ao longo de dois caminhos perfeitamente distintos. Um dos caminhos tem como ponto de partida, o ábaco, uma extensão mecânica da idéia de contar nos dedos. Os aparelhos provenientes do ábaco, usam dígitos para representar os números e são sempre chamados de COMPUTADORES DIGITAIS. O outro caminho surgiu da construção da régua e compasso de agrimensor antigo. A evolução natural dêstes antigos equipamentos, deu origem aos COMPUTADORES ANALÓGICOS. Êstes, poderiam ser chamados de computadores "contínuos", porque se baseiam na medida contínua de um determinado comprimento ou a distância entre dois pontos. Aquêles, seriam chamados de computadores "discretos", pois êles identificam sòmente valores discretos 0,1,2 , etc, e representam êstes valores por quantidades físicas contáveis, tais como, dentes de uma engrenagem ou passos de uma cremalheira.

Cada uma das famílias de computadores, tem seus desenvolvimentos independentes da outra. Entretanto, a combinação entre as duas é possível e ocorre em muitos casos, dando origem aos chamados computadores digitais-analógicos ou analógicos-digitais.

Vamos nos referir apenas aos computadores digitais.

Em 1600, John Napier, matemático escocês, aperfeiçoou o conceito de exponenciação, dando origem aos logarítmos e preparou uma tábua de multiplicação com peças de madeira e osso, que passou a ser usada pelos astrônomos, agrimensores e navegadores por muitos séculos. Era o calculador de mesa da época.

Logo depois das descobertas de Napier, Blaise Pascal , em 1642, na França, projetou e construiu provàvelmente a primeira máquina de somar para seu pai usar em seus negócios.

Mais ou menos na mesma época, Gottfried Wilhelm Leibnitz também inventou uma máquina de somar, independentemente da idéia de Pascal. O "bom calculador" - assim se chamava - começou a ser construído em 1671 e foi completado sòmente em 1964.

Em 1786, J.H. Müller, engenheiro, concebeu a idéia de um computador automático e expressou sua idéia, no papel, em formas precisas e conceitos razoáveis. Entretanto, foi desencorajado devido as dificuldades técnicas de construção.

Quem primeiro fêz a sugestão para se construir um computador automático, foi Charles Babbage, um jovem inglês de 20 anos. Por volta de 1820, Babbage projetou um computador, baseado nas tabelas matemáticas já existentes e fazendo uso de diferenças de alta ordem. A máquina foi construida e levou o nome de "máquina de diferenças", e fornecia as respostas impressas sem auxílio do homem. Com a idade de 30 anos, Babbage foi persuadido pelo govêrno inglês, a projetar u'a máquina de dimensões maiores, a qual tomou-lhe dez anos de trabalhos ininterruptos. Em vista do alto custo, aproximadamente £ 17000, o projeto foi abandonado.

A idéia de se construir um computador automático, voltou em meados de 1937, mas se completou sòmente em 1943/1944 devido a 2a. Guerra Mundial. Estas máquinas eram ainda do tipo eletro-mecânicas.

Computadores utilizando circuitos eletrônicos começaram a ser construidos sòmente em 1946. Os primeiros a serem colocados no comércio datam de 1950.

No período de 1950 - 1960, os computadores digitais sofreram um desenvolvimento assombroso, sob vários pontos de vista, com a descoberta do transistor; pois, antes os computadores eram à válvula termoiônica.

Os computadores digitais podem se classificar em três grandes gerações:

1) de 1a. geração até 1955 - caracterizam o tempo de milisegundos (10^{-3}) - utilizam válvulas;

2) de 2a. geração até 1960 - caracterizados em microsegundos (10^{-6}) - utilizam transistores;

3) de 3a. geração após 1960 - caracterizados em nanosegundos (10^{-9}) - utilizam transistores e circuitos micro-miniaturizados.

Os computadores de 1a. geração são constituidos de comandos de contrôle em painéis de programação, onde os circuitos de contrôle são armados externamente e colocados no computador. Ao passo que, os computadores de 2a. e 3a. geração já dispõem internamente dêstes circuitos de contrôle.

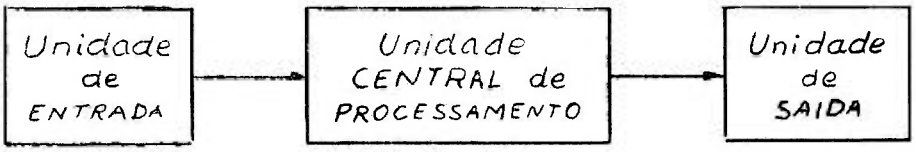
As características que distinguem um computador eletrônico de uma máquina de calcular, são:

1 - rapidez no processamento de dados;
2 - memória de armazenamento;
3 - decisões lógicas a considerar;
4 - PROGRAMA.

CAPÍTULO II

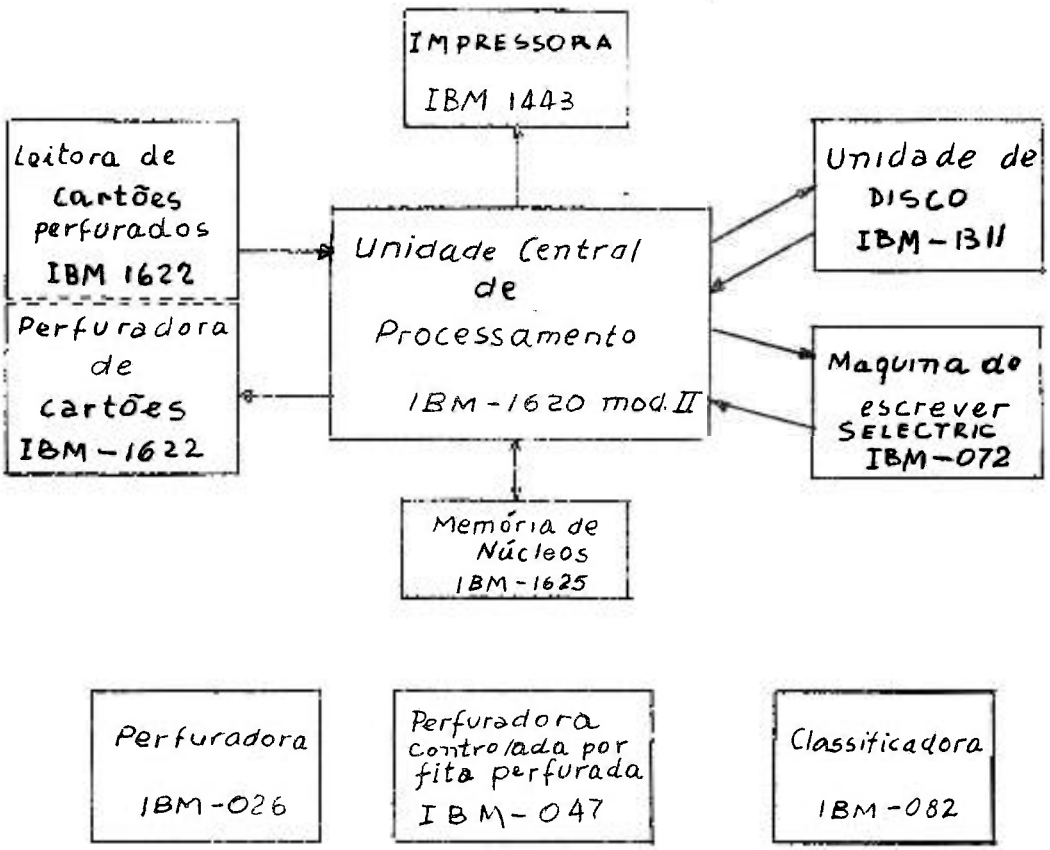
Generalidades sôbre sistemas de processamento de dados.

O sistema básico de um computador digital é constituido de 3 partes, isto é:



A unidade central de processamento (UCP) é a unidade principal onde se requer uma determinada sequência de códigos para que a máquina possa operar. A sequência de códigos é fornecida pela unidade de entrada; ao passo que, os resultados são fornecidos pela unidade de saída.

O equipamento eletrônico do I.E.A. (IBM-1620 Mod. II) é esquematizado como segue:



"Performance" das unidades:

1622 - leitora de cartões perfurados: lê 250 cartões por minuto;
1622 - perfuradora de cartões: perfura 125 cartões por minuto;
1443 - impressora: imprime 400 linhas por minuto (unidade a ser instalada);
072 - máquina de escrever: imprime 15,5 caracteres por segundo;
1625 - memória de núcleos magnéticos: capacidade para 40000 posições;
1311 - unidade de disco (memória auxiliar): capacidade para $2x10^6$ caracteres;
1620 - unidade central de processamento: é constituida de uma secção aritmética e uma secção de contrôle.

Memória de trabalho é uma parte do sistema central de computação ou processamento de dados que convenientemente comandada, armazena informações, processa informações armazenadas e transmite os resultados dêstes armazenamentos e dêstes processamentos.

Os tipos de memória de trabalho são:

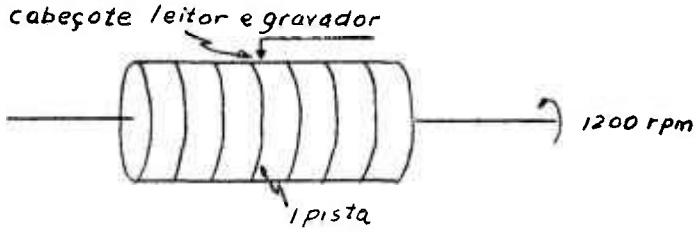
1 - tambor magnético;
2 - núcleos magnéticos;
3 - discos magnéticos (memória auxiliar; normalmente usada para fins de armazenamento de informações).

Um tambor magnético é constituído de um cilindro que gira em tôrno de um eixo à velocidade de aproximadamente 1200 rpm. A leitura dos caracteres são distribuidos na superfície do cilindro em pistas de óxido metálico; estas por sua vez são divididas em blocos; os blocos em memórias e as memórias armazenam dígitos. Por exemplo, no computador GAMMA TAMBOR da Bull, o tambor cilíndrico apresenta a seguinte configuração:

128 pistas		(de 0 à 127);
1 pista	apresenta 8 blocos	(de 0 à 7);
1 bloco	apresenta 48 memórias	(de 0 à 47);

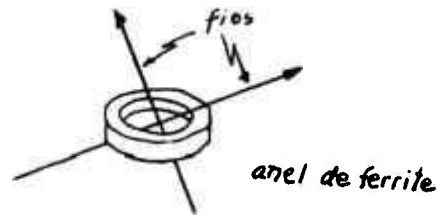
1 memória tem capacidade de armazenar 12 dígitos.

Esquemàticamente, temos:



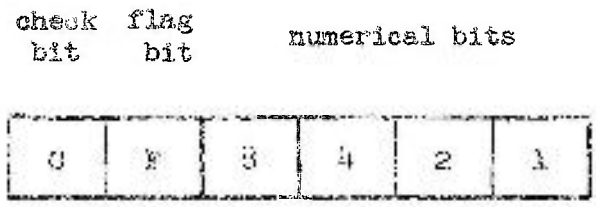
Uma memória de núcleos magnéticos é constituida por grande número de anéis de ferrite atravessados por dois fios. Cada anel admite dois estados perfeitamente diferentes: ora encontra-se magnetizado - estado que designaremos pelo algarismo 1 - ora encontra-se não magnetizado - estado 0 - .

Pela combinação dos anéis, podemos ter a representação dos diversos caracteres, lembrando a codificação do sistema binário.



0 =	0	0	0	0
1 =	0	0	0	1
2 =	0	0	1	0
3 =	0	0	1	1
4 =	0	1	0	0
5 =	0	1	0	1
6 =	0	1	1	0
7 =	0	1	1	1
8 =	1	0	0	0
9 =	1	0	0	1

Na memória de núcleos, uma posição de memória corresponde a uma linha da tabela acima; duas posições de memória corresponde à combinação de duas linhas e assim sucessivamente. Em cada linha da referida tabela ainda faltam dois "bits" (binary digits): um bit de paridade (check bit) e um bit de sinalização (Flag bit). De um modo geral temos o arranjo BCD (binary-coded decimal) do seguinte modo:



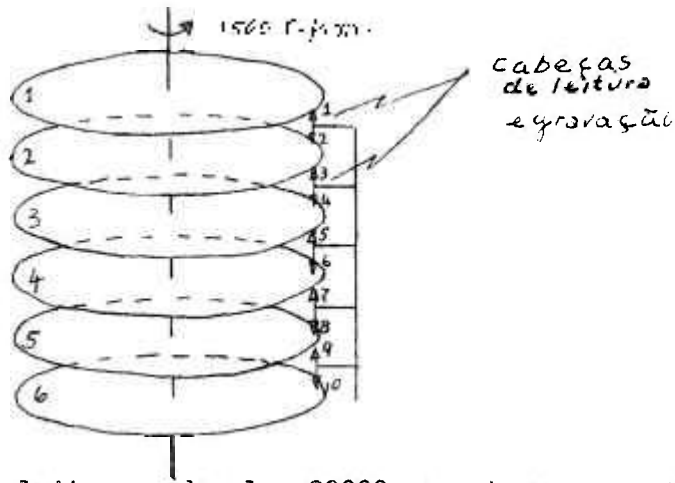
A codificação BCD é sempre usada em todos os computadores de pequeno e médio portes.

A memória auxiliar (unidade de discos IBM 1311) é constituida por seis discos presos a um eixo à uma velocidade de 1500 rpm. As superfícies utilizáveis são sòmente as internas, isto é, as superfícies externas dos discos extremos não são aproveitáveis. Dispomos então de 10 superfícies. Entre cada uma das superfícies internas existe uma cabeça gravadora-reprodutora dupla que tem a finalidade de ler e gravar caracteres na superfície do disco.

A organização e capacidade desta unidade pode ser esquematizada como segue:

100 caracteres compõem um setor;
20 setores, 2000 caracteres, compõem uma trilha;
10 trilhas, 20000 caracteres, compõem um cilindro;
10 cilindros, 2000000 de caracteres constituem os seis discos; a primeira e última face não são utilizadas, servindo só para proteção.

Esquematizando, temos:



As pontas de leitura podem ler 20000 caracteres por segundo, ou seja, um cilindro completo.

Os dois tipos de memória que constituem o equipamento eletrônico do I.E.A. são de núcleos magnéticos (40000 posições) e unidade de disco (2×10^6 posições).

CAPÍTULO III

Linguagem Fortran II-D

3.1. - Generalidades:

FORTRAN quer dizer FORmula TRANslator. O computador digital IBM 1620 mod. II utiliza como linguagem automática, o FORTRAN II e FORTRAN II-D.

No que se segue, faremos uma descrição da linguagem FORTRAN II-D, que necessàriamente usa o sistema Monitor I, exigindo a utilização da unidade de disco IBM 1311. Esta linguagem é bàsicamente o FORTRAN II, mais as instruções próprias para utilização do disco. Também trataremos das instruções necessárias para a utilização de uma impressora em linha com o computador, a IBM 1443.

Vimos que um determinado computador necessita de uma sequência lógica de códigos para que o mesmo possa operar. Esta se

quência é denominada "Programa". Então, programa nada mais é do que uma série de instruções daquilo que o computador deve processar para resolver um determinado problema.

Os principais tipos de linguagem com as quais um programa pode ser escrito e processado são:

a) linguagem de máquina
b) linguagem simbólica
c) linguagem automática

A linguagem de máquina, também denominada linguagem absoluta, é aquela que a máquina entende e processa as instruções do programa. Cada tipo de computador tem a sua linguagem absoluta própria. Para se estudar a linguagem de máquina necessitamos conhecer o computador em todos os aspectos, isto é, como computador analisa e executa cada instrução.

Um programa escrito em linguagem absoluta, denomina-se "Programa Objeto".

A linguagem simbólica, também denominada mnemônica, é uma linguagem que visa facilitar a escrita de um programa, pois como o próprio nome indica, a programação - ato de fazer um programa - é feita através de símbolos mnemônicos. Esta linguagem, para as máquinas IBM, denomina-se sistema de Programação Simbólica (SPS). Daqui . SPS . II D

Um programa escrito em SPS, denomina-se "Programa Simbólico Fonte". Como o computador não entende a linguagem mnemônica, necessitamos traduzí-lo em linguagem absoluta. Esta tradução pode ser feita pelo programador, entretanto, cada computador dispõe de um programa (em linguagem de máquina) que tem a finalidade de realizar esta tarefa; êste programa denomina-se "Monta-dor SPS".

A linguagem automática, como o próprio nome indica, é uma linguagem em que as instruções são feitas seguindo conceitos

Daqui . FORTRAN II D

semelhantes aos das formulações algébricas.

Exemplificado:

Efetuar a adição abaixo:

$Y = A + B + C$ onde: $A = 25$; $B = 249$ e $C = 18418$,

colocados na memória de alguma forma.

1 - Em linguagem de máquina:

```
21   01001   01004
21   01001   01009
```

A 1a. instrução indica: some (código 21) o que está na posição de memória 01004 (B) com o que está na posição 01001 (A) e guarde o resultado desta soma em 01001 (A + B). A 2a. instrução indica: some o que está na posição [illegible]

2 - Em linguagem simbólica (utilizando endereço direto):

```
010   A   1001,  1004
020   A   1001,  1009
```

A 1a. instrução indica: primeira linha (010); some (código A) o que está na posição 1004 (B) (os zeros de alta ordem podem ser eliminados) com o que está na posição 1009 (C) com o que está na posição 1001 (A + B) e guarde o resultado em 1001 (A + B + C = Y).

3 - Em linguagem FORTRAN:

$$Y = A + B + C$$

A única instrução indica: some A com B e o resultado desta adição some com C e guarde o resultado final em uma área da memória designada por Y.

Como pode-se observar, a linguagem automática (FORTRAN) reproduz de modo semelhante a representação algébrica da expressão dada no exemplo.

Com o aparecimento da linguagem automática, o uso do computador tornou-se acessível ao público em geral, tanto à Ciência e Engenharia como ao Comércio, pois esta linguagem não requer conhecimento da máquina em si; ao passo que as linguagens anteriormente referidas, exige-o. Devido a esta grande facilidade de programação, surgiram diversas linguagens automáticas, tais como: FORTRAN, ALGOL, COBOL, MAD, NELIAC, LISP e outros tantos.

A Association for Computers Machines (criadora do ALGOL), pretende tornar esta linguagem, o Esperanto para os computadores, entretanto, por outro lado, a International Business Machines (criadora do Fortran), pretende fazer o mesmo para sua linguagem.

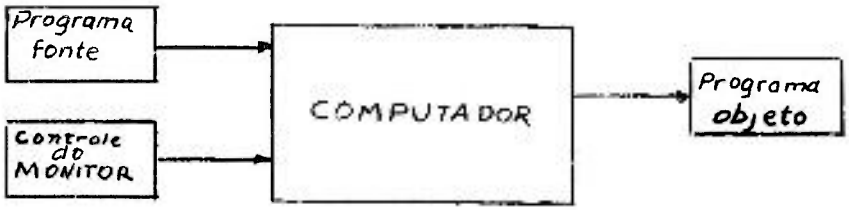
Em se tratando de FORTRAN, existem diversas versões dependendo do porte da máquina, isto é, quanto maior o seu porte, mais recursos para programação dispomos.

Assim como no SPS, a linguagem FORTRAN precisa ser traduzida em linguagem de máquina. O veículo utilizado para êste fim é denominado "Compilador FORTRAN". Uma vez o programa fonte traduzido (pelo montador SPS ou compilador FORTRAN), o programa originado recebe a denominação de "Programa Objeto", isto é, o programa objeto é o programa fonte escrito em linguagem de máquina.

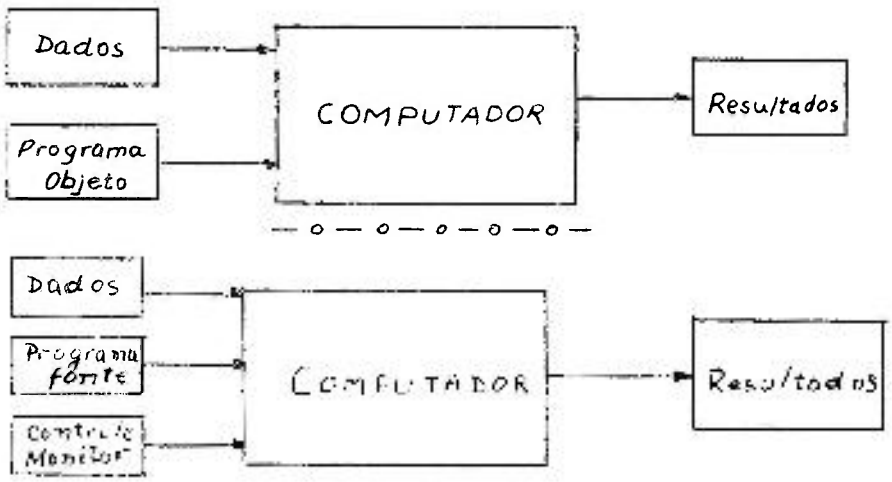
Esta operação de "traduzir" o programa fonte, é feita pelo próprio computador com auxílio do compilador. O compilador do FORTRAN II-D permite obter o programa em cartões (2 etapas), ou diretamente obter os resultados. Neste último caso o programa objeto fica temporàriamente ou permanentemente gravado no disco magnético. Esquematizando o procedimento, temos:

Compilação do programa objeto em cartões

1a. etapa:

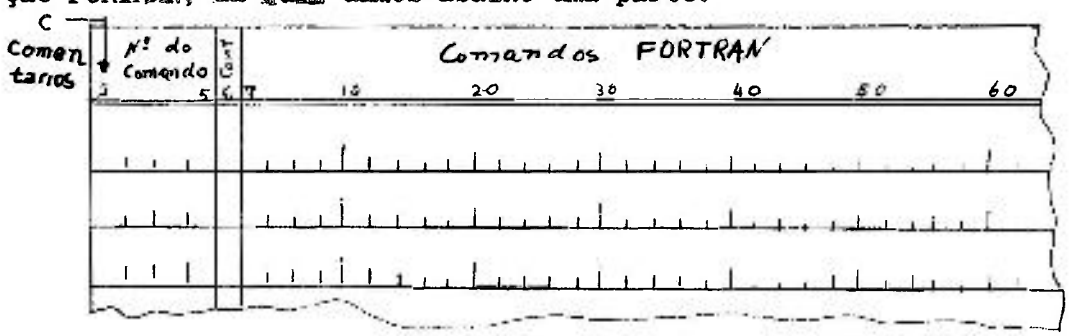


2a. etapa:



Compilação sem saída do programa objeto

Um programa FORTRAN é constituído de uma série de comandos (statements) ordenados lògicamente, de tal forma que cada comando deve ser escrito em uma linha da fôlha especial de programação FORTRAN, da qual damos abaixo uma parte:



Entretanto, o FORTRAN II-D permite que um comando continue nas linhas seguintes (no máximo até 330 caracteres, isto é, 5 linhas no

total), bastando para isso que se coloque na coluna 6 das linhas seguintes, qualquer caracter ou número de 1 a 9. O primeiro cartão deve ter a coluna 6 em branco ou zero.

Cada linha da fôlha de programação FORTRAN, dará origem a um cartão de 80 colunas perfurado, com características semelhantes à fôlha de programação FORTRAN;

A fôlha de programação FORTRAN é constituida de 72 colunas, que têm as seguintes finalidades:

a) - a coluna 1 pode ter a letra C (comentário), se se quiser fazer qualquer observação sôbre o problema em estudo; por exemplo, o nome do cálculo a ser efetuado. Todo cartão que na coluna 1 tiver a letra C, não é processado pela máquina. Os cartões "comentário" podem aparecer sempre que houver necessidade de se fazer uma observação no programa. Em geral, o (s) primeiro(s) cartão de um programa é de comentário, o qual especifica o cálculo a ser processado;

b) - as colunas de 2 a 5 se destinam aos números dos comandos, sempre que se fizer necessário. Com isto, podemos numerar os comandos, desde 1 a 9999; é norma de boa programação que se o número do comando contiver apenas um algarismo, êste deve estar na coluna 5; se dois, nas colunas 4 e 5; se três, nas colunas 3, 4 e 5; se quatro, nas colunas 2, 3, 4 e 5. Porém é indiferente que o número seja colocado em qualquer coluna, exceto nas colunas 1 e 6. Colunas em branco, sem perfuração, não são consideradas pelo compilador;

c) - a coluna 6 se destina à continuação de uma linha imediatamente anterior, conforme já explicado;

d) - as colunas de 7 a 72 se destinam à escrita do programa pròpriamente dito, de tal modo que cada caracter deve estar em uma coluna.

e) - as colunas de 73 a 80 não são processadas em hipó-

tese alguma pelo COMPILADOR. Dêste modo, podemos colocar qualquer informação de identificação, se desejarmos.

3.2. - Linguagem FORTRAN II-D pròpriamente dita:

Ao se tratar do FORTRAN II-D, as observações são válidas para o FORTRAN II, que não usa o sistema Monitor, exceto em alguns casos especiais, que salientaremos especìficamente.

3.2.1. - Precisão Aritmética

A precisão das quantidades usadas em cálculo com computadores é uma consideração importante, principalmente em muitos problemas científicos.

No FORTRAN II-D para a 1620, a precisão é bastante boa, uma vez que podemos operar até com 28 casas decimais. O usuário tem a possibilidade de definir a precisão desejada desde 2 a 28 algarismos na forma decimal, e de 4 a 10 algarismos na forma inteira. Esta definição é feita em um cartão de contrôle do Monitor, onde se coloca um asterisco (*) na coluna 1, FANDK nas colunas 2 a 6, o número de algarismos decimais desejados nas colunas 7 e 8, e o número de algarismos inteiros desejados nas colunas 9 e 10. Êste cartão deve preceder o programa fonte. No caso de ser omitido êste cartão, a máquina assume 8 algarismos para números na notação decimal, e 4 algarismos para números inteiros. (no FORTRAN II, compilador em cartões, os números inteiros são assumidos com 5 algarismos). O valor de F não pode ser menor que 02, e o valor de K não pode ser menor que 04.

3.2.2. - Constantes e Variáveis

3.2.2.1. - Constantes

Dois tipos de constantes são permissíveis no FORTRAN II-D: constante de ponto fixo (restrita aos números intei

ros), e constante de ponto flutuante (característica dos números decimais).

As constantes de ponto fixo podem ter no máximo dez (10) algarismos significativos e devem ser escritas sem o ponto decimal. O sinal (+) é opcional se a constante fôr positiva e, se negativa deve ser precedida de um sinal (-).

Exemplos: 20; +415 ou 415; -55027 .

As constantes de ponto flutuante podem ter um número qualquer de algarismos significativos na parte decimal. A máquina, entretanto, tomará os primeiros 8 algarismos significativos se o cartão de contrôle fôr omitido, ou tomará o número de algarismos significativos indicados pelo cartão de contrôle. A máquina toma os algarismos significativos decimais estipulados no cartão de contrôle, sem fazer arredondamentos. O sinal (+) é opcional se a constante fôr positiva e, se negativa deve ser precedida pelo sinal (-).

Exemplos: 17.0 ou 17. ou +17. ou +17.0; -0.003 ou -.003;

Quando o número fôr muito grande, podemos escrevê-lo utilizando a forma exponencial (E).

Exemplos: $5.0E3 = 5.0 \times 10^{3}$

$5.0E+3 = 5.0 \times 10^{3}$

$5.0E-7 = 5.0 \times 10^{-7}$

$0.1E3 = 0.1 \times 10^{3}$

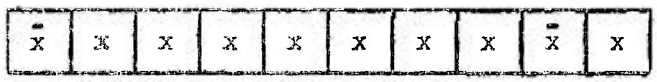
$-0.1E-3 = -0.0001$

A grandeza de um número desta forma deve estar entre os limites de 10^{-100} e $(1-10^{-f}) \times 10^{99}$ (excluindo os pontos extremos) ou deve ser zero. O valor de f varia de 2 à 28.

É conveniente se saber como uma constante é armazenada na memória. Se é uma constante de ponto flutuante, o número é re-

duzido entre 0.10000000 à 0.99999999, sendo que 10 posições de memória são reservadas para cada número, quando não se usa cartão de controle; no caso de se usar o cartão de controle, são reservadas f + 2 posições de memória.

Esquematizando, temos, para ponto flutuante:

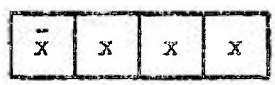


mantissa — característica (expoente)

Exemplos:

Número	Convertido em	Forma que é armazenado
148.50728	$.14850728 \times 10^{3}$	$\bar{1}485072\bar{8}\bar{0}\bar{3}$
0.248075	$.24807500 \times 10^{0}$	$\bar{2}480750\bar{0}\bar{0}$
-18.0075031246	$-.18007503 \times 10^{2}$	$\bar{1}800750\bar{3}\bar{0}\bar{2}$
0.00004567	$.45670000 \times 10^{-4}$	$\bar{4}567000\bar{0}\bar{0}\bar{4}$
-0.000000000002	$-.20000000 \times 10^{-11}$	$\bar{2}000000\bar{0}\bar{1}\bar{1}$

Para Ponto Fixo: No caso de constante de ponto fixo, o número é guardado usando 4 posições de memória, quando não se usa cartão de controle. No caso de se usar cartão de controle, será guardado em K posições de memória.



Número Inteiro

Exemplos:

Número	Forma que é armazenado
3	$\bar{0}\bar{0}\bar{0}3$
144	$\bar{0}\bar{1}44$
5412	$\bar{5}412$

3.2.2.2. - Variáveis:

Dois tipos de variáveis são permissíveis: variável de ponto fixo (restrita aos valores inteiros) e variáveis de ponto flutuante.

As variáveis podem ser constituidas de 1 à 6 caracteres alfabéticos ou numéricos, sendo que o primeiro deve ser sempre alfabético, e os demais caracteres são agrupados arbitràriamente.

As variáveis de ponto fixo devem ter o seu primeiro caracter formado por uma das letras I, J, K, L, M ou N, não excedendo nunca de 6 caracteres.

Exemplos: I; M2; JOBNO; L15MOA; N45702; IVOLT; IEA.

As variáveis de ponto flutuante devem ter o seu primeiro caracter diferente de I, J, K, L, M ou **N**, podendo assumir qualquer valor de grandeza dentro do intervalo 10^{-100} à $(1 - 10^{-f}) \times 10^{99}$ ou zero.

Exemplos: ALFA; Z; YIJKLM; H234K8; VOLT; CNEN; DER.

Deve-se tomar cuidado para que os nomes escolhidos para as variáveis não sejam um dos seguintes: LOGF; SINF; COSF; EXPF; SQRTF; ATANF e ABSF; LOG; SIN; COS; EXP; SQRT; ATAN; ABS, pois êstes constituem as Funções de Biblioteca.

3.2.2.3. - Subscritos:

Qualquer variável de ponto fixo ou flutuante pode ser subscritada, onde o subscrito deve aparecer sempre entre parênteses, e a variável subscritada pode ter de 1 a 3 subscritos, só de ponto fixo, diferentes de zero.

Se representarmos por v uma variável positiva de ponto fixo e por c (ou c') uma constante positiva de ponto fixo, o subscrito pode ser uma expressão das seguintes formas:

v, c, v + c, v - c, c * v, c * v + c' ou c * v - c'

onde o símbolo * indica multiplicação.

Exemplos: I; 3; MU + 2; 5 * J; 5 * J - 2

Então as variáveis subscritadas tomam o aspecto:

A(I); K(3); BETA (7 * J - 2); IGAMA(I,J); MATRIZ (L,M,N);etc.

Cada variável subscritada deve ter o tamanho da formação , isto é, os valores máximos que os subscritos assumem, especificado no comando DIMENSION, precedendo o aparecimento da variável no programa fonte.

EXERCÍCIOS

1. Os seguintes números de comandos são aceitáveis no FORTRAN II ? Porque?

   a) 10234
   b) 23.08
   c) 475
   d) 0.01
   e) 5.0E3
   f) 114
   g) 0

2. Se quizermos trabalhar com uma precisão aritmética de 6 algarismos na parte inteira e 17 algarismos na parte decimal de uma coleção de números, o que devemos fazer?

3. No caso de se esquecer do cartão de contrôle, no exercício anterior, o que irá suceder?

4. Porque as constantes de ponto fixo relacionadas abaixo são inaceitáveis:?

a) 145,205.0

b) 20.

c) + 0.2

5. As variáveis seguintes são de ponto fixo ou ponto flutuante? Porque?

a) NOVE

b) ZETA

c) CÁLCULO

d) I19B

e) X

f) 14HJ

g) OK

h) 109

6. Os seguintes pares de constantes representam o mesmo número? Porque?

a) 1.0 e 1.

b) +23.84 e 23.84

c) 0.003 e .3E-2

d) 0.003 e .3E-02

e) 245. e 2.45E2

f) .906E5 e +906.0E + 2

g) - 0.1E-3 e - 0.01E01

7. As variáveis subscritadas são permissíveis? Porque?

a) I (I)

b) AMP (10)

c) A (20*J)

d) BETA (K + 3)

e) B (K)

f) X (2 * J + 3)

g) X (5 * K -8)

h) Y (J + 1, 8*N, M-4)

CAPÍTULO IV

4.1. - Tipos de Comandos FORTRAN II - D

A linguagem automática FORTRAN II-D é constituida de 31 comandos que podem ser classificados do seguinte modo:

Tipos	Finalidade	Quantidade de comandos
Comando Aritmético	Especificam um cálculo numérico	1
Comandos de Entrada/Saída	Prevê as rotinas necessárias de entrada e saída	10
Comandos de Contrôle	Governam a sequência lógica do programa	10
Comandos de Especificação	Prevê informações requeridas (ou desejáveis) para tornar o programa objeto eficiente	5
Comandos de Subprogramas	Possibilita ao programador definir e usar os subprogramas	5

Estudaremos cada um dos tipos de comando separadamente, completando os conceitos teóricos com uma série de exemplos e exercícios, ao mesmo tempo que iremos evoluindo em programação FORTRAN II - D.

4.2. - Expressões - Comando Aritmético

Uma expressão em FORTRAN II - D é qualquer sequência de constantes, variáveis (subscritadas ou não) e funções, separadas

por símbolos de operação, vírgulas e parênteses de acôrdo com as regras de construção de expressões.

Em operações tipo aritmético, os seguintes símbolos de operações são utilizados:

+	adição
-	subtração
*	multiplicação
/	divisão
**	exponenciação ou involução
( )	parênteses (único símbolo de reunião entendido pelo compilador do FORTRAN II - D)

Em uma expressão onde se omite o parênteses, a hierarquia das operações tem a seguinte ordem:

Ordem	Símbolos	Operações
1	**	exponenciação
2	* e /	multiplicação e divisão
3	+ e -	adição e subtração

Por exemplo,

$$A + B/C + D^{**}E * F - G$$

é interpretada como:

$$A + (B/C) + (D^{E} * F) - G$$

Se o parênteses fôr omitido em uma sequência de multiplicações e divisões consecutivas (ou adições e subtrações consecutivas) é entendida como agrupadas a partir da esquerda. Então, se " . " representa * ou/ (+ ou -), a sequência

A. B. C. D. E

é interpretada da seguinte maneira: ((((A.B).C). D). E).

As regras para construção de expressões são as seguintes:

1a.) - Quantidades em ponto fixo e ponto flutuante não podem ser misturadas numa mesma expressão. Quantidades de ponto fixo podem aparecer em expressão de ponto flutuante sòmente como subscrito ou como expoente.

Exemplos:

$\left(\frac{a + b}{c}\right)^2$ ——— ((A + B) /C)**2

$\left(\frac{a + b}{c}\right)^{2.5}$ ——— ((A + B) /C)**2.5

$f(x_3)$ ——— F (X (3))

2a.) - Expressões de ponto fixo não podem ter expoentes de ponto flutuante.

Exemplo:

$(2j + k)^{1.5}$ ——— (2*J + K)**1.5 (não é permitido)

Obs. Uma expressão ambígua tal como X**Y**Z não é permitida. Deve ser escrita como X **(Y ** Z) ou como (X ** Y) ** Z, conforme a operação desejada.

3a.) - Uma expressão precedida de um sinal (+) ou (-) não afeta o módulo da expressão resultante. Por exemplo, P, + P e - P são expressões de mesmo módulo.

4a.) - Colocando uma expressão entre parênteses, o seu módulo não é afetado. Por exemplo, B, (B), ((B)), são expressões de mesmo módulo.

5a.) - Dois sinais operacionais não podem aparecer juntos.

Exemplos: * - ; ** * ; * *

utilize A * (-B) e não A * - B

Como vimos, todos os símbolos de operações tem o mesmo significado que em matemática, com exceção do sinal de igual (=). É êste símbolo que define o único comando aritmético, o qual podemos colocá-lo na forma geral, isto é:

| a = b |

onde a é uma variável subscritada ou não sem sinal, e b é uma expressão.

Exemplos:

Q1 = 3.0 * B

A(I) = B(I) * FUN (C (I))

O significado do sinal (=) é o seguinte: "coloque o resultado da expressão à direita do sinal de igual em uma área da memória definida pela variável à esquerda do sinal de igual". No caso do primeiro comando acima, é definido uma área de memória Q1 onde é armazenado o resultado da expressão 3.0 * B. No outro caso, é definido uma área de memória A(I) onde é armazenado o resultado da expressão B(I) * FUN (C(I)). O resultado é estocado em forma de ponto fixo se a variável à esquerda do sinal de igual é uma variável de ponto fixo; em forma de ponto flutuante se a variável fôr de ponto flutuante.

Se a variável à esquerda é de ponto fixo e a expressão à direita é de ponto flutuante, o resultado estocado fica truncado da parte decimal. Por exemplo, se o resultado é * 3.9752, o número de ponto fixo armazenado é * 3, e não *4 (não faz arredondamento). Se a variável à esquerda é de ponto flutuante e a expressão à direita é de ponto fixo, o resultado armazenado é convertido e estocado em ponto flutuante. Por exemplo, se o resultado é

* 324, o número estocado é * 324.

Exemplos de comandos aritméticos:

	Significado
A = B	Armazene o valor de B em A.
K = X	Trunque a parte decimal de X e armazene a parte inteira em K.
A = L	Transforme L em ponto flutuante e armazene em A.
M = M + 1	Some 1 unidade a M e armazene em M.
A = I * B	Não é permitido pois se trata de uma expressão mixta.
B = 2.* D	Armazene em B o produto 2 D.
B = 2 * D	Não é permitido pois a expressão é mixta.

NOTA: R * R é mais rápido que R ** 2.

EXERCÍCIOS

1) Escreva em FORTRAN as seguintes expressões matemáticas:

a) $(x + y)^2$

b) $x + y^2 + z^3$

c) $a + \frac{b}{c}$

d) $\frac{a + b}{\sqrt{c}} - \frac{h}{x^2}$

e) $a^2 + \frac{b}{c + d}$

f) $\frac{1}{x} + \frac{x}{2!} + \frac{x^2}{3!}$

g) $\left(\frac{x}{y}\right)^{x^2 - 1}$

h) $\frac{-b + \sqrt{b^2 - 4ac}}{2 * a}$

i) $\frac{1}{6} \pi \; h \; (3 \; r^2 + h^2)$

2) É dado abaixo algumas expressões matemáticas e as correspondentes em FORTRAN. Indique se existe algum êrro ou não.

a) $\frac{x + 2}{y + 1}$ ——— X + 2.0/Y + 4.0

b) $\left(\frac{x + a + \pi}{2 z}\right)^2$ ——— X + A + 3.1416 /(2.*Z)** 2

c) $\left(\frac{x}{y}\right)^{r-1}$ ——— (X/Y) ** (R-1)

d) $a+bx + cx^2 + dx^3$ —— A + X * (B + X * (C + D * X)

3) As expressões abaixo são aceitáveis, entretanto, contêm pelo menos um par de parênteses que podem ser removidos sem alterar o significado das mesmas. Reescreva-as com o mínimo de parênteses possível:

a) (A + X) * (B/Y)

b) (((A) + (B)) + (C) * (D) ** 2)/(((A+2.8) **(I-1) +
+ B/(C+D)) * (A+6.))

c) (A/C) * B

4) Qual o valor de A ou I armazenado, como resultado dos seguintes comandos aritméticos:

a) A = 2 * 6 - 1
b) A = 2/3
c) A = 2. * (10/4)
d) I = 2. * (10/4)
e) I = 2 + 3 * 4
f) A = 4. * 3./2.
g) I = 19/4 + 5/4

5) Escreva os comandos aritméticos para cada uma das fórmulas abaixo:

a) $y = -\frac{b}{a}$

b) área $= \frac{4}{3} \quad r^2$

c) $h = \frac{\frac{1-x}{r}}{\beta - 1}$

d) $E = m\, c^2$

e) $e = e_o + v_o\, t - \frac{1}{2}\, a\, t^2$

f) $y = \sqrt[3]{(x - 1)\,(x - 2)^2}$

g) $y = \sqrt{a^2 + x^2 + \sqrt{b^2 + (d - x)^2}}$

6) Nos comandos aritméticos abaixo existe pelo menos um êrro. Indique-os:

a) 4 = I
b) V - 3.96 = X ** 1.67
c) K6 = I * * A
d) X = (A + 6) * * 2
e) + V = A + B
f) Y = 2. X + A
g) A = ((X + Y)A * * 2 + (R - S) * * 2 / 1,67

CAPÍTULO V

5.1. - Comandos de Entrada/saída

Como já nos referimos anteriormente, existem 10 coman-

dos no FORTRAN II disponíveis para especificar a transmissão de informações, durante a execução do programa objeto, entre a memória e qualquer dispositivo de entrada/saída, isto é, leitora de cartões, leitora de fita de papel, perfuradora de cartões e máquina de escrever. No equipamento do IEA não tem leitora de fita de papel.

Os comandos de entrada e saída são dez: (READ, ACCEPT , PUNCH, PRINT, TYPE, FIND, FETCH, RECORD, PUNCH TAPE e ACCEPT TAPE). Têm a finalidade de transmitir uma lista de quantidades especificadas nos respectivos comandos, entre a memória e qualquer dispositivo de entrada/saída. Os dois últimos comandos não iremos estudar.

Exemplificando: Se o problema apresentar uma lista de dados pequena, êstes podem entrar no programa como constantes, em comandos aritméticos; seja calcular a raiz quadrada de

$$A ** L + 4. * X$$

o programa seria:

```
A = 9.
L = 3
X = 2.
RAIZQ = (A ** L + 4. * X) ** .5
```

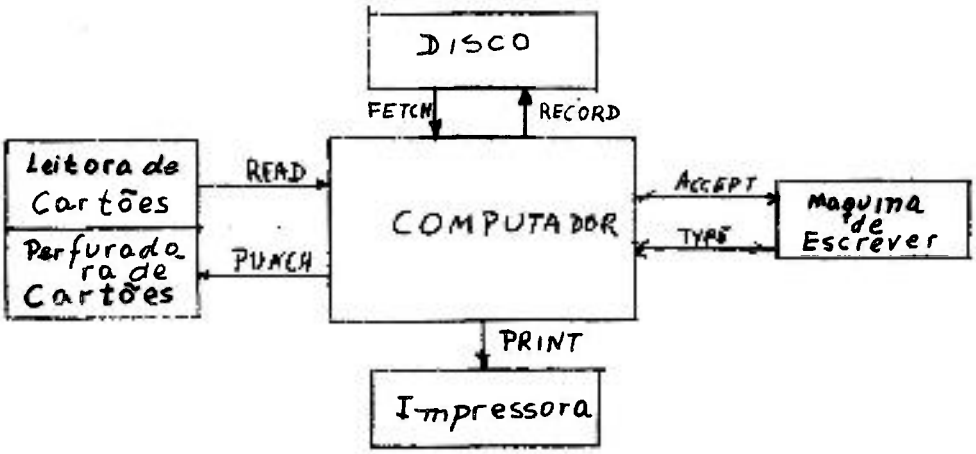
(mais alguns comandos para mandar os resultados da RAIZQ para algum dispositivo de saída. Aqui já vemos a necessidade de um comando de saída).

Esta não é comumente a melhor maneira de se entrar com os dados. Suponhamos que não se conheça os valores de A,L e X, que serão colocados em um cartão perfurado; teremos, então que ter uma maneira de entrar com os dados, os quais virão através de um dispositivo de entrada/saída, em vez dos três comandos acima.

Os dados entram na memória como consequência de um coman-

do de leitura, que lista os nomes das variáveis para as quais os novos valores serão lidos. Assim, no nosso sistema podemos, quando conveniente, comandar a leitura de cartões (entrada), perfurar cartões (saída), receber os dados pela máquina de escrever (entrada) ou imprimir os resultados pela máquina de escrever (saída), ou pela impressora em linha com o computador (saída), ler do disco magnético (entrada), e gravar no disco magnético (saída).

Esquematizando, temos:



5.2. - Comando READ

A forma geral dêste comando é:

```
READ  n, lista de variáveis de entrada
```

onde n é o número de um comando FORMAT, que será explicado mais tarde, entre os comandos de especificação.

Exemplo:

```
READ 1, A, B, C, D(3)
```

O comando READ manda ler cartões da Leitora-Perfuradora, onde se encontram os valores numéricos das variáveis A, B, C, D(3), sendo que o número 1 que aparece no exemplo é o número do comando FORMAT correspondente. Êste número é escolhido arbitrària

mente, mas tem um importante significado, pois é por intermédio dêle que podemos fazer referência cruzada no programa.

5.3. - Comando ACCEPT

A forma geral dêste comando é:

```
ACCEPT  n, lista de variáveis de entrada
```

onde n é o número de um comando FORMAT.

Exemplo:

```
ACCEPT 999,  MATRIZ (I,J), VOLT, IAMP (4)
```

O comando ACCEPT induz à máquina de escrever ficar preparada para receber dados das variáveis discriminadas no comando acima, sendo que o nº 999 é o número de um comando onde se encontra um FORMAT.

5.4. - Comando PUNCH

A forma geral dêste comando é:

```
PUNCH  n, lista de variáveis de saída
```

onde n é o número de um comando FORMAT

Exemplo:

```
PUNCH 44, (A (J), J = 1,10)
```

O comando PUNCH manda perfurar em cartões os resultados das variáveis discriminadas neste comando, de acôrdo com o comando FORMAT de nº 44.

5.5. - Comando TYPE

A forma geral dêste comando é:

```
TYPE  n, lista de variáveis de saída
```

onde n é o número de um comando FORMAT.

Exemplo:

```
TYPE 185, ((ARRAY (I,J),I = 1,3), J = 1,5)
```

O comando TYPE causa a saída de dados pela máquina de escrever. A esfera de tipos da máquina de escrever se posiciona e sucessivas linhas são impressas de acôrdo com o comando FORMAT localizado no comando nº 185. O número máximo de caracteres que a máquina pode imprimir em uma linha é 85.

Voltando ao exercício anterior, podemos programá-lo do seguinte modo:

```
      READ 10, A, L, X
   10 FORMAT  ( ------ )
      RAIZQ = (A**L + 4.*X)**.5
      TYPE 125, RAIZQ
  125 FORMAT  ( ------ )
```

Significado: O primeiro comando manda ler um cartão de dados onde se encontram os valores numéricos de A, L, X, no formato especificado no comando 10 (segundo comando); o terceiro comando manda calcular a raiz quadrada de $a^4 + 4x$ e armazenar o resultado em uma área da memória designada por RAIZQ; o quarto comando é uma instrução que manda sair pela máquina de escrever o valor numérico armazenado na área denominada por RAIZQ, de acôrdo com o formato que se encontra no comando número 125.

5.6. - Comando PRINT

A forma geral dêste comando é:

```
PRINT  n,  lista de variáveis de saída
```

onde n é o número do comando FORMAT. O comando PRINT faz sair os dados pela impressora em linha com o computador, sendo 120 ou 144 caracteres por linha impressa conforme indicado no comando FORMAT.

Êste comando é usado quando está ligada a IBM 1443 ao computador. Caso não exista esta ligação, o comando PRINT cumpre as mesmas funções que o comando TYPE.

Exemplo:

```
PRINT 2, A, B, J
```

Os valores A, B, J, serão impressos de acôrdo com as especificações do comando FORMAT numerado com 2.

5.7. - Comando FIND

A forma geral dêste comando é:

```
FIND  (I)
```

onde (I) especifica o número do registro onde a leitura ou gravação deve ter início. O parâmetro (I) deve ser: 1) - Uma variável de ponto fixo não subscritada; exemplo:

```
FIND  (IND)
```

ou

2) - Uma variável de ponto fixo com subscrito; exemplo:

```
FIND  (JIM(5))
```

O comando FIND posiciona o braço de acesso sôbre o cilindro onde serão lidos ou gravados os dados. É um comando que deve preceder os comandos FETCH ou RECORD, quando se deseja ganhar tempo com processos adicionais, enquanto o braço está em movimento. O parâmetro (I) deve ser o mesmo para ambos os comandos. O valor de (I) começa com 1, e cada valor corresponde a um setor gravado, quando se especifica registros de um setor no comando DEFINE DISK. No

caso de se especificar registros de dois setores, o valor de (I) começa também de 1, mas cada valor corresponde aos dois setores do registro. Unicamente, áreas definidas no comando DEFINE DISK poderão ser especificadas no FIND.

5.8. - Comando FETCH

A forma geral dêste comando é:

FETCH(I) lista de variáveis a ser lida

onde (I) especifica o número do registro onde começará a leitura. Para a memória de núcleos, serão transferidos os dados na ordem em que foram gravados no registro especificado por (I), e serão identificados com as variáveis da lista na ordem em que figuram nessa lista.

Exemplos:

1 - INDICE(5) = 10
FETCH (INDICE(5))(A(J), J=1,25)

Êste comando indica que serão lidas do disco para a memória de núcleos as variáveis da lista, começando no setor especificado pelo valor de (INDICE(5)), identificando com A(1) os primeiros dígitos dêsse setor 10, de acôrdo com o comprimento da "palavra" usado, com A(2) os primeiros dígitos do 2º setor (11), com A(3) os primeiros dígitos do 3º setor (12), etc.

2 - I = 1
FETCH (1) A, B, X, Y, EJ, OK, E, F, Z, AM, J

Se foram usadas "palavras" flutuantes de comprimento 10 e "palavras" fixas de comprimento 4, êste comando indica que serão lidos do disco os valores que estão a partir do setor nº 1 do cilindro reservado pelo comando DEFINE DISK, identificando os primeiros 10 dígitos com a variável A, os segundos 10 com B, os 10 últimos com AM, e os primeiros 4 dígitos do segundo setor (I=2)

serão identificados com a variável J.

Se a lista especifica mais valores de variáveis do que pode ser obtido de um registro, como no 2º exemplo, o valor do índice ou parâmetro (I) será automàticamente incrementado de 1, e a leitura continua no próximo registro sequencial. Êste procedimento continua até que a lista seja "satisfeita", ou até atingir o fim da área reservada pelo N_2 do comando DEFINE DISK. Ao finalizar a operação de leitura, o valor de I é o número do último registro lido mais 1. O parâmetro ou índice (I) é o mesmo já descrito no comando FIND.

A instrução compilada do comando FETCH transfere o contrôle à rotina de Entrada e Saída do Monitor.

5.9. - Comando RECORD

A forma geral dêste comando é:

```
RECORD(I) lista de variáveis a ser gravada
```

(I) especifica o número do registro onde começará a gravação. Êste comando, que pode ser precedido pelo comando FIND com o mesmo valor de (I), grava no disco os valores das variáveis especificadas na lista, na ordem em que estão nela. Se a lista especifica mais variáveis do que as que podem estar contidas num registro, o valor de I é incrementado de 1, e a gravação continua no próximo registro sequencial até esgotar a lista, ou até alcançar o valor final especificado por N_2 no comando DEFINE DISK. Ao finalizar a operação de gravação o valor de I é o número do último registro gravado, mais 1. A instrução compilada transfere o contrôle à rotina de Entrada e Saída do Monitor.

5.10. - Especificações de Quantidades listadas

Os comandos de entrada/saída que estudamos servem para transmitir informações entre o computador e o meio utilizado

de entrada e saída, e dêste modo devem incluir uma lista ordenada das quantidades a serem transmitidas. A ordem da lista tem que ser a mesma (para entrada ou para saída) que se deseja utilizar através de um dispositivo de entrada/saída.

A formação e o significado de uma lista de quantidades é descrita abaixo com um exemplo. Suponhamos que o valor de K tenha sido prèviamente definido.

```
A,B(3), (C(I),D(I,K),I=1,10),((E(I,J),I=1,10,2),F(J,3),J=1,K)
```

Se esta lista é usada em um comando de saída, a informação que será impressa estará na seguinte ordem:

```
A,B(3),C(1),D(1,K),C(2),D(2,K),..., C(10),D(10,K),
E(1,1),E(3,1),...,E(9,1),F(1,3),E(1,2),E(3,2),...,
E(9,2),F(2,3),...,E(1,K),K(3,K),...,E(9,K),F(K,3)
```

De igual modo, se esta lista é usada em um comando de entrada, as quantidades serão encaminhadas, para a memória, nesta mesma ordem. A lista é lida da esquerda para a direita, com repetição para as variáveis entre parênteses. Sòmente variáveis podem ser listadas. Para as constantes não é permitida esta listagem.

Se uma lista de variáveis é usada, a sua execução é exatamente como um laço de um DO. A ordem da lista acima pode ser considerada equivalente ao seguinte "programa":

```
1 sair com A
2 sair com B(3)
3 Do 5 I=1, 10
4 sair com C(I)
5 sair com D(I,K)
6 DO 9 J=1,K
7 DO 8 I=1,10,2
8 sair com E(I,J)
9 sair com F(J,3)
```

Sendo K definido anteriormente, um comando da forma:

```
READ 10,K,(A(I),I=1,K)
```

substitui perfeitamente a série de comandos:

```
ACCEPT 1,K
DO 5 I=1,K
5 READ 10,A(I)
```

5.11. - Informações de entrada/saída

A linguagem FORTRAN II-D permite uma notação do tipo do DO, para transmitir dados. Por exemplo, queremos transmitir 5 (cinco) quantidades A(1),A(2), A(3), A(4) e A(5). Será dado o comando:

```
READ 9,(A(I), I=1,5),
```

que é equivalente a:

```
    DO 12 I=1,5
12  READ 9,A(I)
```

Ao comando:

```
TYPE 2, (ARRAY(J), J=1,10,2)
```

temos em correspondência uma saída pela máquina de escrever, na seguinte ordem:

ARRAY(1), ARRAY(3), ARRAY(5), ARRAY(7), ARRAY(9)

Aos comandos:

```
DIMENSION A(25)
PUNCH 1,A
```

vão corresponder cartões perfurados onde se encontram as quantidades A(1),A(2),...A(25). A quantidade de variáveis perfuradas em cada cartão dependerá da especificação do comando FORMAT correspondente.

5.12. - Entrada e Saída em forma Matricial

Por exemplo, o comando:

READ 1, ((A(I,J), I = 1,2), J = 1,3)

provoca a leitura de uma matriz IxJ (no caso, 2 x 3). Os dados são lidos para dentro da memória na mesma ordem em que êles são encontrados no dispositivo de entrada.

Quando se deseja uma entrada/saída de uma matriz, de uma vez, uma notação abreviada pode ser usada para uma lista de comandos de entrada/saída; sòmente o nome da matriz será dado e a informação subscritada pode ser omitida.

Então, se A é definido prèviamente num comando de especificação, DIMENSION, o comando

READ 1,A

é suficiente para ler todos os elementos da matriz. Se A não fôr definido prèviamente num comando DIMENSION, sòmente será lido o primeiro elemento de A.

CAPÍTULO VI

6.1. - Comandos de Especificação

Os cinco comandos de especificação no FORTRAN II-D são:

FORMAT, DIMENSION, DEFINE DISK, EQUIVALENCE e COMMON.

Êstes comandos não são executáveis; sòmente fornecem instruções ou informações necessárias ao compilador para a produção do programa objeto.

6.2. - Comando FORMAT

A forma geral dêste comando é:

FORMAT (S_1, S_2, ... S_n)

onde cada S_1 é uma especificação do formato desejado. Esta especificação, isto é, a maneira de transmitir os dados pode ser dada de três formas:

a) forma de variável de ponto flutuante (F);
b) forma de variável de ponto fixo (I);
c) forma exponencial (E).

a) Forma de variável de ponto flutuante:

Êste formato tem o seguinte tipo: Fw.d, onde:

F especifica ponto flutuante;
w indica o número total de caracteres, incluindo o ponto e o sinal;
d indica o número de casas decimais que existe no número

Exemplos: Representar os números 235,65 e -2,84576 no formato F.

+235,65 —— F 7.2 { w = 7, pois temos 5 algarismos; mais um sinal (+) e a vírgula (total: 7 caracteres)
d = 2, pois são duas casas decimais

- 2,84576 —— F 8.5 { w = 8
d = 5

b) Forma de variável de ponto fixo:

Êste formato tem o tipo: I w

onde, I especifica ponto fixo;
w indica o número total de caracteres incluindo o sinal.

+1258 ———— I 5 { w = 5, pois temos 4 algarismos; mais o sinal (total: 5 caracteres)

-4 ———— I 2 { w = 2

c) Forma exponencial:

Êste formato tem o seguinte tipo: E w.d

onde,

E especifica exponencial;

w indica o número total de caracteres inclusive a vírgula, o sinal e ainda os caracteres necessários para se escrever a potência de 10;

d indica o número de casas decimais.

O formato E requer uma transformação intermediária;coloca-se o número dado com o primeiro algarismo significativo após a vírgula e ajusta-se o expoente de base 10.

Exemplo: transformar intermediàriamente o número .....
236,78917

236,78917 ———————— + .23678917 E + 03

Uma vez transformado o número, pode-se escrevê-lo no formato Ew.d.

No exemplo acima temos:

+.23678917E + 03 ——— E 14.8 { d = 8 , pois temos 8 casas decimais; w = 14 , pois temos um total de 14 caracteres.

Exemplos: Representar os números abaixo no formato E.

		Convertido em:	No formato E
1º)	123,45678	+.12345678 E + 03	E 14.8
2º)	0.0076543	+.76543 E - 02	E 11.5
3º)	-0.00000058	-.58 E - 06	E 8.2
4º)	-4,8530489245	-.48530489245 E + 01	E 17.11
5º)	6.0	+. 6 E + 01	E 7.1

	Convertido em:	No formato E
6º) 0.325×10^{60}	.325 E + 60	E 9.3

Quando duas ou mais variáveis em sequência tem o mesmo formato, pode-se escrever, por exemplo:

2 F 5.2

que corresponde à F5. 2, F5.2

Exemplo: seja A = 23.834 e B = -10,221. O comando FORMAT na forma F seria:

FORMAT (F 7.3, F 7.3) ou
FORMAT (2 F 7.3)

Êste modo de escrever a instrução FORMAT é válido para qualquer das três formas F, I e E.

Voltemos ao exercício anterior (da raiz quadrada) e vamos programá-lo completamente. Os valores das variáveis A = 9. ; L = 3 e X = 2.

```
      READ  10, A, L, X
      RAIZQ = (A ** L + 4. *X)**.5
      TYPE 20, RAIZQ
   20 FORMAT (E14.8)
   10 FORMAT (F 3.0, I 2, F 3.0)
```

Se o primeiro comando fôsse:

READ 10, A, X, L

O comando número 10 seria:

10 FORMAT (2 F 3.0, I 2)

A instrução número 20, indica que o resultado da raiz se

rá dado pela máquina de escrever, e na forma

$$\pm \; . \; X \; X \; X \cdot X \; X \; X \; X \; X \; E \pm \; X \; X$$

6.3. - Formato de campos em branco;

Êste formato é do tipo: W X

onde, W indica quantos espaços em branco devem ser deixados;
X especifica o formato em branco

Exemplo:

Analisemos o seguinte formato de saída pela máquina de escrever:

FORMAT (F6.2 , 2X , I4 , 3X , I2)

A linha impressa será do tipo:

$$\pm \; XX.XX \; bb \pm \; XXX \; bbb \pm \; X$$

onde o símbolo $\underline{b}$ indica espaço em branco.

6.4. - Formato alfabético

1. - Formato tipo H

Êste formato é do tipo : W H

onde, W indica a quantidade de caracteres alfabéticos incluindo espaços em branco deixados entre as palavras;
H especifica o formato alfabético.

Exemplos:

a) - Se quizermos imprimir pela máquina de escrever a frase: "computador digital 1620 modêlo II", devemos dar o seguinte comando de formato

FORMAT (34 H COMPUTADOR DIGITAL 1620 MODÊLO II)

b) - O comando FORMAT (3HXY = F8.3)
produz qualquer uma das linhas abaixo:

XY = 9999.999
XY = b-93.210
XY = bb28.768

2. - Formato tipo A

A forma geral é do tipo Aw

Êste tipo de formato é utilizado para ler ou escrever caracteres alfanuméricos, seja uma variável ou nome de um arranjo, ou título de um trabalho, etc. O máximo valor de w é o valor de f/2, ou k/2, dependendo de que a variável, título ou nome do arranjo, seja flutuante ou fixo.

Êstes campos são manejados como um campo aritmético, e quando armazenada uma variável como ponto flutuante, terá um zero como expoente, o que não terá efeito sôbre a entrada ou saída. Porém, se o primeiro caráter no campo é branco, ponto decimal ou parênteses, o campo será tratado como zero pelas subrotinas de ponto flutuante.

Exemplo:

Suponhamos que deve ser lido um título que deverá aparecer no início de um cálculo, e que êsse título muda a cada vez que o cálculo recomeça. No primeiro cálculo o título deve ser REATORES DE POTÊNCIA, e no segundo cálculo deve ser REATORES DE PESQUISAS. Determinamos o número de caracteres, incluindo espaços, para a frase mais longa, neste caso 21 para o segundo título. Determinamos uma variável dimensionada, por exemplo TITUL, com o valor da área reservada igual à quantidade de caracteres alfabéticos a serem lidos, neste caso 22 para ser número par. Como o nome da variável é

de ponto flutuante, f = 8 e w = 4; usaremos os seguintes comandos:

```
      DIMENSION TITUL (6)
      READ 10, TITUL
   10 FORMAT (6A4)
```

Consideramos 24 caracteres (6 x 4), por ser múltiplo de 4. Ambos os títulos entrarão cada um em um cartão de dados.

6.5. - Formato para a utilização da Impressora

Quando se dispõe de uma impressora em linha com o Computador (Printer IBM 1443), o comando FORMAT, além do uso normal já indicado, também fornece dados necessários para deixar espaços ou realizar saltos entre duas linhas de impressão. Êste comando deve começar sempre com 1H seguido do código de contrôle que especifica a operação desejada. Êstes códigos são:

branco - espaço simples antes de imprimir
0 - espaço duplo antes de imprimir
1 a 9 - salto imediato ao canal 1 a 9

Exemplo:

```
      PRINT 33, A, B, J
   33 FORMAT (1HO, F8.2, F8.2,I8)
```

Êstes comandos fornecerão duplo espaço entre a linha que está sendo impressa e a linha impressa prèviamente.

A primeira especificação de contrôle é aplicável à primeira linha de impressão, sòmente. Se mais que uma linha deve ser impressa, a especificação deve preceder a cada linha que será impressa.

Exemplo:

```
33 FORMAT (1HO,F8.2/1H, E14.8)
```

Êste comando fornecerá espaço duplo entre a linha já impressa e a

correspondente à primeira especificação F8.2, depois proverá espaço simples entre esta última linha impressa e a linha correspondente à especificação E14.8.

Observações Importantes

1) - Os comandos FORMAT não são executados; sua função é simplesmente suprir informações para o programa objeto. Dêste modo, êles podem ser colocados em qualquer lugar do programa fonte, exceto como primeiro comando no intervalo ou alcance de um comando de contrôle denominado DO.

2) - Ao comandarmos uma saída de resultados pela máquina de escrever SELECTRIC, devemos tomar cuidado para não ultrapassar a capacidade de caracteres por linha; o número máximo de caracteres que podem ser impressos por linha é 85.

3) - O mesmo se aplica aos cartões; o número máximo de caracteres perfurados em um cartão não deve ultrapassar 80.

4) - O comando FORMAT fornece a impressão de 120 caracteres por linha quando acompanha o comando PRINT, correspondente à impressora IBM.1443. Também pode fornecer 144 caracteres por linha, se o sistema foi definido para tal quantidade de caracteres.

5) - Os dados de entrada devem estar no mesmo formato que foram definidos nos comandos FORMAT do programa fonte, porém o uso do ponto decimal real é opcional: Se êle é proporcionado pelo cartão de dados, passa-se por alto sôbre d, nas especificações tipo F ou E.

6.6. - EXEMPLO:

Vamos agora dar um exemplo ilustrativo, empregando a maioria dos conhecimentos até aqui estudados. Após a programação iremos analisar comando por comando.

Desejamos calcular s e ar definidos por:

$$s = \frac{a + b + c}{2}$$

$$ar = (s(s - a)\ (s - b)\ (s - c))^2$$

onde os valores numéricos de a, b, e c estão perfurados nesta seqüência em um cartão de dados. Escrever um programa para calcular s e ar e mandar imprimir os dados e resultados pela máquina de escrever, isto é, imprimir a, b, c, s e ar, reservando 14 espaços para cada quantidade, e deixando 2 espaços em branco entre cada quantidade. Em outras palavras, a saída deve ser impressa do seguinte modo:

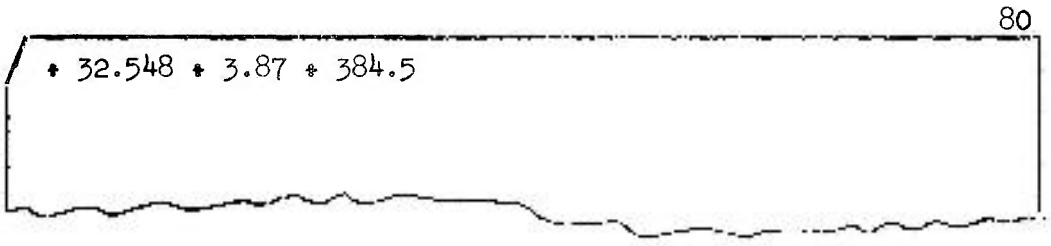
1a. linha	a	b	c	s	ar
2a. linha	valor numérico de a	valor numérico de b	valor numérico de c	valor numérico de s	valor numérico de ar

Programa FORTRAN

```
C  CÁLCULO DE DUAS EXPRESSÕES
   READ 5,A,B,C
   S = (A + B + C)/2.
   AR = (S*(S-A)*(S-B)*(S-C))**2
   TYPE  10
10 FORMAT (1HA,15X,1HB,15X,1HC,15X,1HS,15X,2HAR)
   TYPE 12, A, B, C, S, AR
 5 FORMAT (F7.3, F5.2, F6.1)
12 FORMAT (F7.3,9X,F5.2,11X,F6.1,10X,E14.8,2X,E14.8)
   (mais dois comandos para indicar fim do cálculo  e
    fim do programa).
```

O cartão de dados terá os valores numéricos de ABC, isto é, +32.548; +3.87; +384.5, respectivamente.

Cartão de dados:



Análise do programa acima:

1º comando: apenas um comentário; não é processado.

2º comando: lêr o cartão de dados (que é reproduzido acima) onde se encontra os valores numéricos de A, B e C, de acôrdo com o formato estipulado no comando número 5 (8º comando).

3º comando: calcule $\frac{A + B + C}{2}$ e guarde o resultado em uma área da memória designada por S.

4º comando: calcule $\left[S\,(S-A)(S-B)(S-C) \right]^2$ e guarde o resultado em uma área da memória designada por AR.

5º comando: imprima uma linha pela máquina de escrever com os caracteres A, B, C, S e AR de acôrdo com o comando de formato número 10 (6º comando).

7º comando: imprima outra linha pela máquina de escrever com os valores numéricos de A, B, C, S e AR de acôrdo com o formato número 12 (9º comando).

As linhas impressas apresentarão o seguinte aspecto:

A	B	C	S	AR
+32.548	+3.87	+384.5	±.XXXXXXXX ± XX	±XXXXXXXXE ± XX

EXERCÍCIOS

1) Programar a seu critério as expressões matemáticas:

a) $y = \sqrt{a^{i} + 4x} - a^{k}$

b) $x = \dfrac{y^2 + z^2}{c} .a$

2) Faça um programa para calcular:

$$x = \frac{e.h.p}{\frac{h^2}{16} + h^2 p^2}$$

onde e, h e p são lidos em um cartão de dados e p é sempre inteiro.

Perfure o resultado num cartão, perfurando na seguinte ordem: e, h, p e o resultado x. A especificação do formato fica a seu critério.

3) Leia a, x e s pela máquina de escrever e calcule:

$$y = x^2 - a^2$$

$$z = \frac{x.s}{2} - \frac{a^2}{2} (x + s)$$

Imprima o resultado y e z na máquina de escrever e também perfure em um cartão.

6.7. - Comando DIMENSION

A forma geral é a seguinte:

```
DIMENSION V,V, ...V
```

onde, cada V é o nome de uma variável, subscritada com uma, duas,

ou três constantes de ponto fixo sem sinal. Qualquer quantidade de V pode ser dado.

Exemplo:

DIMENSION A(10),B(5,15),CVAL(3,4,5)

Êste comando fornece instruções ao compilador, para definir as variáveis indexadas e guardar as posições de memória para a quantidade dos valores que o índice vai assumir durante o programa. O comando DIMENSION pode aparecer em qualquer parte do programa , mas deve preceder o primeiro aparecimento da variável subscritada que é por êle definida, e nunca pode ser o primeiro comando do intervalo do comando DO.

O exemplo acima indica que B é bi-dimensional, para o qual os subscritos nunca devem exceder 5 e 15.O comando DIMENSION guarda 75(5x15) áreas onde serão colocados os elementos do arranjo B.

Exemplo:

DIMENSION X(100),A(3,10),K(125)

O compilador reservará 100 locações para X, 30 para A e 125 para K.

É de responsabilidade do programador escrever o programa de tal maneira que o índice nunca fique maior que o mencionado no DIMENSION, que também não pode ser menor que 1.

Se estas restrições forem violadas, o programa fonte será compilado, entretanto, o programa objeto dará resultados falsos ao ser processado.

Tôda vez que num programa vai-se usar uma variável com índice, deve aparecer o nome da variável numa lista do comando DIMENSION, em caso contrário o programa fonte não será compilado.

6.8. - Comando EQUIVALENCE

A forma geral é a seguinte:

EQUIVALENCE (a,b,c,...),(d,e,f,...)

onde, a,b,c,d,e,f, ... são variáveis que podem conter como subscritos apenas constantes.

Exemplo:

EQUIVALENCE (A,B(1),C(5)),(D(17),E(3))

O comando EQUIVALENCE estabelece o privilégio de controlar a localização de dados na memória. Em outras palavras, quando a lógica do programa permite, a quantidade de localizações usadas pode ser reduzida para causar, numa mesma posição de memória, localização de duas ou mais variáveis. Note que êste comando não é usado para representar a igualdade matemática entre dois ou mais elementos. Se variáveis de ponto fixo e flutuante são equivalenciadas, seus comprimentos têm que ser os mesmos, isto é f+2 tem que ser igual a k.

Um comando EQUIVALENCE poderá ser colocado em qualquer parte do programa fonte; exceção feita ao primeiro comando do alcance do DO, pois neste comando devemos ter sempre uma instrução executável. Cada par de parênteses de uma lista no comando EQUIVALENCE, envolve os nomes de duas ou mais quantidades, as quais serão armazenadas na mesma posição de memória, durante a execução do programa objeto. Qualquer quantidade de pares de parênteses pode ser dada.

No comando EQUIVALENCE, um têrmo genérico C(p), para $p > 0$, significa a localização pésima dum arranjo C. Por exemplo, consideremos C(5); então C está na 5a. posição da lista. Se p não fôr especificado, será interpretado como 1(um).

No exemplo acima dado, indica que as quantidades A, B e C, são para ser armazenadas na memória, de tal modo que os elementos A, B(1), C(5) vão ocupar a mesma área de memória. Completando o comando, temos que D(17) e E(3), ambas vão ocupar uma outra área de memória.

Quantidades ou arranjos que não são especificados no comando EQUIVALENCE, ocupam uma única área da memória para cada quantidade ou arranjo.

No comando EQUIVALENCE, sòmente localizações entre variáveis podem ser distribuídas. Não são aceitas distribuições entre constantes.

Para se planejar a separação das posições de memória, é preciso ter um perfeito conhecimento de certos comandos FORTRAN II-D, pois poderá causar um nôvo valor a ser armazenado na mesma posição. Êstes comandos são em número de quatro:

a) a execução de uma fórmula aritmética, armazena um nôvo valor para a variável à esquerda do sinal de igualdade.

b) a execução de um comando DO, armazena novos valores para o índice.

c) a execução dos comandos READ ou ACCEPT, armazena novos valores para as variáveis mencionadas na listagem.

O comando EQUIVALENCE é útil de duas maneiras distintas,

1º) Permite ao programador definir dois ou mais nomes de variáveis que signifiquem a mesma coisa, como poderia ocorrer que após escrever um programa muito longo o programador percebe que inadivertidamente trocou o nome de algumas variáveis e que por exemplo X, XIS e RES referem-se à mesma variável ; então, em lugar de mudar os nomes das variáveis no programa, o que seria um processo além de trabalhoso, sujeito a êrros, basta simplesmente escrever

EQUIVALENCE (X, XIS, RES)

2º) Permite utilizar o mesmo lugar de armazenamento, para a localização de duas ou mais variáveis que não se necessita ao mesmo tempo. Suponhamos que num comando inicial READ de um programa aparece a variável MAR que é utilizada imedia

tamente num comando aritmético e que não será mais utilizada, durante o programa aparece um índice I de um comando DO, mas que é usado apenas nesta interação e posteriormente a variável MAS é usada com o mesmo propósito. Tal como estão, estas três variáveis ocuparão três diferentes áreas da memória; se o programa estiver necessitando de posições de memória, basta o programador colocar as três variáveis numa mesma posição, escrevendo

EQUIVALENCE (MAR, I, MAS)

6.9. - Comando COMMON

A forma geral é a seguinte:

```
COMMON  A, B, .....
```

onde A, B, ... são nomes de variáveis e nomes de arranjos não subscritados.

Exemplo:

COMMON X, ANGLE, MATA, MATB

Para as variáveis ou arranjos que aparecem no COMMON , são designadas áreas da memória de modo mais ou menos análogo ao que faz o comando EQUIVALENCE. Estas áreas são utilizadas pelo programa e seus subprogramas. Onde a lógica do programa permite, êste comando pode resultar em grande economia de espaço na memória.

Nomes de arranjos contidos no comando COMMON, terão que aparecer também no comando DIMENSION do mesmo programa.

A área de memória onde se localizam as variáveis do COMMON, é o fim da memória, partindo da posição 39.999 e abaixando em ordem sequencial decrescente. Por exemplo:

COMMON A, B, C

com f=10, A, B, C serão armazenadas nas posições 39.999, 39.987 e 39.975. Se C é dimensionado, como C(10), o endereço 39.975 é o

endereço de C(10), que é o último elemento do conjunto, e 39.867 é o endereço de C(1).

Devido a interação complexa dos comandos COMMON e EQUIVALENCE, o programador deve se prender às duas regras seguintes:

1. Quando as mesmas variáveis devem aparecer tanto no comando COMMON como no comando EQUIVALENCE, o COMMON que contém estas variáveis deve preceder ao comando EQUIVALENCE.

Exemplo:

```
COMMON  A
EQUIVALENCE (A,B,C)
```

A ordem das variáveis do EQUIVALENCE não tem importância, e a regra 1 se aplica tanto para as variáveis comuns B ou C.

2. Dentro de uma lista do EQUIVALENCE não pode estar mais do que uma variável a qual já foi:

a) equivalenciada
b) colocada na memória comum

A sequência seguinte dos comandos não é permitida no FORTRAN II-D:

```
EQUIVALENCE (A, B, C)
EQUIVALENCE (X, Y, Z)
EQUIVALENCE (A, Z)    viola a), tanto A como Z já aparece
                                ram em um EQUIVALENCE
COMMON D
EQUIVALENCE (D, X, P) viola a combinação de a) e b),
                                D aparece num comando COM
                                MON e X num comando EQUI-
                                VALENCE prévios
```

O compartilhamento das localizações de memória desejadas acima, pode ser realizado com os comandos:

```
COMMON D
EQUIVALENCE (D, X, P)
EQUIVALENCE (A, B, C, X)
EQUIVALENCE (X, Y, Z)
```

ou

```
COMMON D
EQUIVALENCE (D, A, P, B, C, X, Y, Z)
```

No caso de serem violadas uma das regras acima, uma mensagem de êrro será impressa pela máquina de escrever.

6.10. - Comando DEFINE DISK

A forma geral é a seguinte:

DEFINE DISK (N_1, N_2)

onde:

N_1 é uma constante de ponto fixo que especifica o número de variáveis contidas num registro de dados.

N_2 é o número total de registros de dados que será usado pelo programa principal e os sub-programas associados a êle.

O comando DEFINE DISK indica ao compilador as dimensões (número de setores) que deve reservar na área de trabalho do disco para transferir da memória ao disco e vice-versa, os valores dos registros de dados que precisarão do uso do disco.

Êste comando deve aparecer no programa principal (ou no programa ligado (Link)), e aparecerá só uma vez, quando o progra-

ma ou sub-programas associados utilizam qualquer comando de entrada ou saída de disco.

Todos os sub-programas usados pelo programa principal, ou programas ligados, devem ter as dimensões de registros definidas neste comando.

O valor de N_1 depende de duas coisas:

1) - Do comprimento da palavra (w) a ser compilada.
2) - Do comprimento do registro de dados no disco, o qual pode ser 1 ou dois setores físicos.

Se w vêzes N_1 é menor ou igual a 100, o comprimento do registro no disco será de um setor de disco (100 dígitos)

Se w vêzes N_1 é maior que 100, e menor ou igual a 200, o comprimento serão 2 setores de disco.

Exemplo:

Consideremos que o comprimento das palavras sejam: para ponto fixo, k = 4, e para ponto flutuante, f + 2 = 10. Como um registro pode conter só ponto fixo ou só ponto flutuante, ou ambos ao mesmo tempo, o valor de w a ser considerado deve ser o maior dos dois, neste caso w = 10. Com isto, se um registro de dados deve estar contido num setor físico do disco (100 posições), N_1 deve estar compreendido entre 1 e 10. O uso mais eficiente do disco seria com N_1 = 10.

Se o registro de dados deve estar contido em dois setores físicos do disco, N_1 assumirá um valor entre 11 e 20. Um registro de dados não pode ser maior que dois setores (200 dígitos).

Se devem ser lidos ou gravados arranjos e são usadas variáveis de 10 dígitos, o uso mais eficiente do disco seria com arranjos de 9, 19, 29, 39 ... etc, variáveis, tal que uma marca de grupo fique no mesmo setor que registrar as variáveis. No ca-

so de usar 10 variáveis de 10 dígitos por setor, a marca de registro será colocada no início do setor seguinte ao que contém as variáveis especificadas, ficando desocupadas as restantes posições, pois as outras 10 variáveis serão gravadas começando no terceiro setor, e assim por diante.

O número de setores reservado pela especificação de N_2 dependerá do valor de N_1. Se registros de um setor ($N_1 \leq 10$) foi especificado serão reservados N_2 setores; se foi definido registro de dois setores ($N_1 > 10$), serão reservados duas vêzes N_2 setores.

Quando um subprograma usa um comando de entrada ou saída de disco, não é necessário escrever o comando DEFINE DISK, êle já está especificado no programa principal.

CAPÍTULO VII
Comandos de Contrôle

Dos 10 comandos de contrôle, existe um único que não é processado. Começaremos por êsse.

7.1. - Comando END

A forma geral é a seguinte:

```
END
```

ou

```
END (I1,I2,I3,I4,I5)
```

onde I_i é 0,1 ou 2.

Exemplos:

```
END
```

ou

```
END (1,0,0,2,1)
```

Êste comando difere dos demais comandos de contrôle, uma vez que

êle não é processado, isto é, não afeta o fluxo natural do programa objeto. Um comando END serve como aviso ao compilador de que o programa fonte chegou ao fim e a produção do programa objeto deve começar.

O último comando de um programa fonte deve ser sempre um comando END. O comando END(I_1,I_2,I_3,I_4,I_5) é permissível; todavia os I_i são inoperantes no FORTRAN II da 1620.

Sem o comando END o programa fonte não será compilado.

7.2. - Comando PAUSE

A forma geral é a seguinte:

```
PAUSE
```

ou

```
PAUSE  n
```

onde n é uma constante de ponto fixo composta por 5 dígitos e sem sinal, dentro do intervalo de endereços válidos na 1620.

Exemplos:

```
PAUSE
```

ou

```
PAUSE  33333
```

Êste comando provoca uma parada no processamento. Pressionando a tecla START do computador, o programa objeto reinicia a execução à partir da instrução imediatamente seguinte. No comando PAUSE n, o número n pode ser exibido no Registro de endereços da memória, do painel de contrôle da 1620, colocando a chave seletora do MAR em OR-2.

7.3. - Comando STOP

A forma geral é:

```
STOP
```

ou

```
STOP n
```

onde n é uma constante de ponto fixo sem sinal composta por 5 dígitos, dentro de intervalo de endereços válidos na 1620.

Exemplos:

```
STOP
```

ou

```
STOP  33333
```

Êste comando causa uma parada definitiva na execução do processamento do programa objeto; pressionando-se a tecla START, não há o mesmo efeito que no comando PAUSE, pois a instrução STOP é característica de fim de cálculo. Quando êste comando é executado, a esfera de tipos da máquina volta e imprime a mensagem "STOP"; pressionando-se o START, transfere-se o contrôle à Rotina Analisadora de Registros de Contrôle do Monitor.

Como no comando PAUSE n, o valor de n pode ser exibido no console do computador.

7.4. - Comando CALL EXIT

A forma geral é:

```
CALL EXIT
```

Êste comando é usado ao fim de um programa FORTRAN, em lugar do comando STOP, para transferir novamente o contrôle do computador à Rotina Analisadora de Registros de Contrôle do Monitor.

Quando é usado êste comando, tôda vez que a execução do programa chegue ao fim, sairá uma mensagem pela máquina de escrever, indicando fim do trabalho que o computador estêve processando, e ficará pronto para iniciar um nôvo trabalho.

7.5. - Comando GO TO incondicional

A forma geral é a seguinte:

```
GO TO n
```

onde n é o número de um comando.

Exemplo:

```
GO TO 3
```

Êste comando provoca um desvio incondicional, na sequência natural de execução, para o comando número n. O programa, então, continua naturalmente a partir do comando designado por n.

7.6. - Comando GO TO computado

A forma geral é

```
GO TO (n1,n2,...nm),i
```

onde n_1, n_2,...n_m, são números de comandos;
i é uma variável de ponto fixo não subscritada.

Exemplo:

```
GO TO (13,928, 2,2347, 47), L
```

Êste comando provoca um desvio para os comandos n_1,n_2,... n_m, dependendo do valor de i. No exemplo, se L = 3 em um determinado tempo de execução, um desvio é provocado para o terceiro número de comando listado na instrução GO TO computado; no caso, 2.

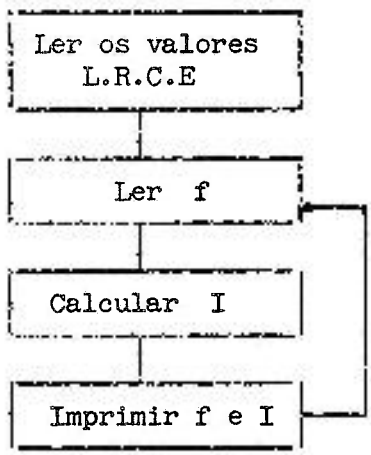
Exercício ilustrativo do comando GO TO n

Deseja-se calcular a corrente de um circuito R, L e C, para diversos valores da frequência f, alimentados por uma bate -

ria que fornece uma tensão E. A fórmula a ser calculada é:

$$I = \frac{E}{\sqrt{R^2 + (2\pi f L - \frac{1}{2\pi f c})^2}}$$

As etapas a serem seguidas no problema podem ser esquematizadas do seguinte modo:



O programa para êste cálculo é o seguinte:

```
      READ 10, OHM,HENRY,FARAD,VOLT
10    FORMAT(F10.0, 3F10.0)
15    READ 10,FREQ
      Y1 = (6.28*FREQ*HENRY-1./(6.28*FREQ*FARAD))**2
      Y2 = (OHM**2+Y1)**.5
      AMP = VOLT/Y2
      PRINT 20,FREQ,AMP
20    FORMAT (1HO,2E14.8)
      GO TO 15
      END
```

Note que a máquina ao passar, no 1º ciclo, pela 9a. instrução (GO TO 15), um desvio é realizado para o comando número

15 e é lido um outro cartão com outro valor da frequência, e assim sucessivamente. Quando os cartões onde estão os valores numéricos de frequência se esgotarem, a máquina para, porque não há mais cartões para ler. Esta não é uma maneira elegante de se parar o computador, entretanto, tolera-se.

Exercício ilustrativo do comando GO TO $(n_1, n_2, \ldots, n_m)$

Calcular os cinco primeiros polinômios de Legendre, de acôrdo com uma variável inteira LEG, já calculada.

Se

LEG=1	calcule	P0(X)=1
LEG=2	"	P1(X)=X
LEG=3	"	$P2(X)= \frac{3}{2} X^2 - \frac{1}{2}$
LEG=4	"	$P3(X)= \frac{5}{2} X^3 - \frac{3}{2} X$
LEG=5	"	$P4(X)= \frac{35}{8} X^4 - \frac{15}{4} X^2 + \frac{3}{8}$

O programa será:

```
. . . . . .
. . . . . .
    GO TO (15, 18, 10, 12, 57), LEG
15  P=1.
    GO TO 120
18  P=X
    GO TO 120
10  P=1.5*X**2-0.5
    GO TO 120
12  P=2.5*X**3-1.5*X
    GO TO 120
57  P=4.375*X**4-3.75*X**2+0.375
```

```
120 PUNCH 16, X, P
 16 FORMAT (2E20.8)
```

. . . . . .

. . . . . .

7.7. - Comando IF

A forma geral é a seguinte:

IF(a)n_1,n_2,n_3

onde,

a é uma expressão em FORTRAN II - D

n_1,n_2,n_3 são números de comandos.

Exemplo:

IF(X-A)15,132,7

Êste comando provoca um desvio para n_1, n_2 ou n_3, dependendo do sinal da expressão a, isto é, se $a < 0$ desvie para n_1; se a=0 desvie para n_2; se $a > 0$ desvie para n_3.

No exemplo acima, se a diferença $(X-A) > 0$, desvie para o comando número 7; se (X-A) = 0, desvie para 132 e se $(X-A) < 0$ desvie para 15.

7.8. - Comando IF(SENSE SWITCH)

A forma geral é:

IF(SENSE SWITCH i)n_1,n_2

onde, i é 1,2,3 ou 4 (chaves de programa localizadas no painel de controle);

n_1,n_2 são números de comandos.

Exemplo:

```
IF(SENSE SWITCH 2)14,128
```

Êste comando provoca um desvio para n_1 ou n_2, dependendo da chave i se estiver ligada ou desligada, respectivamente. No exemplo, se a chave 2 estiver ligada desvie para 14; se estiver desligada desvie para 128. Êste comando permite a interferência do programador.

Exercício ilustrativo do comando IF

Calcular:

$Y=0.5x + 0.95$ se $x \leq 2.1$

$Y=0.7x + 0.53$ se $x > 2.1$

Programa:

```
    READ 2,X
 2  FORMAT(F10.0)
    IF(X-2.1)50,50,40
50  Y=0.5*X+0.95
    GO TO 60
40  Y=0.7*X+0.53
60  TYPE 4,X,Y
 4  FORMAT (2E14.8)
    STOP
    END
```

Exercício ilustrativo do comando IF(SENSE SWITCH)

No exercício ilustrativo do comando GO TO n, a máquina para de um modo incomum. Com o comando IF(SENSE SWITCH) colocado de maneira apropriada, tem-se uma parada mais elegante, pois ao ler o último cartão com o valor da FREQ, desliga-se o switch 3, provocando um desvio para o comando número 30 (STOP).

O programa seria:

```
      READ 10,OHM,HENRY,FARAD,VOLT
10    FORMAT(F10.0,3F10.0)
15    READ 10, FREQ
      Y1 = (6.28*FREQ*HENRY-1./(6.28*FREQ*FARAD))**2
      Y2 = (OHM**2+Y1)**.5
      AMP=VOLT/Y2
      PRINT 20, FREQ, AMP
20    FORMAT (1HO,2E14.8)
      IF(SENSE SWITCH 3) 15,30
30    STOP
      END
```

7.9. - Comando DO

A forma geral é:

DO n i=m_1,m_2,m_3

ou

DO n i=m_1,m_2

onde,

n é o número de um comando subsequente até onde, e inclusive, deve atuar o comando DO;

i é uma variável de ponto fixo, não subscritada e sem sinal;

m_1,m_2,m_3 são constantes de ponto fixo ou variáveis de ponto fixo, não subscritas e sem sinal. m_1 é o valor inicial da variável i. m_2 é o valor final e m_3 é o incremento. Quando o valor do incremento é igual a 1, m_3 é desnecessário; então, vale a 2a. forma geral acima.

O comando DO é o mais poderoso no FORTRAN II-D. Êle torna possível executar uma seção do programa, repetitivamente com mudanças do valor de uma variável de ponto fixo. Conjugado com as variáveis subscritadas, o comando DO provê uma maneira de calcular que seria muito complicada usando simplesmente outros comandos.

Os comandos que seguem o comando DO são executados repetidamente, primeiro com o valor $i=m_1$, depois com $i=m_1 + m_3$, depois $i=m_1 + 2m_3$ e assim sucessivamente até o maior valor de i, não excedendo m_2, isto é, no máximo igual a m_2.

Exemplo:

```
DO n J=1,20,3
```

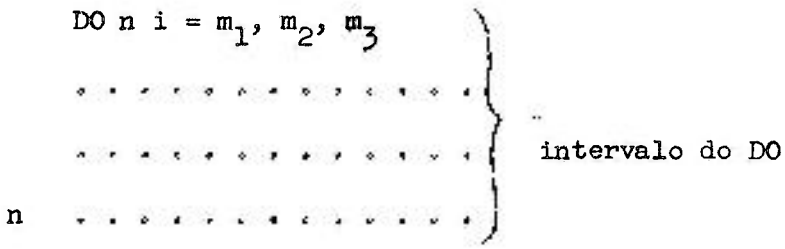
Êste comando faz com que a variável J assuma inicialmente o valor 1, depois 4, depois 7 e assim sucessivamente até 19. Não atinge o máximo valor de J (20) pois o incremento não o permite.

O intervalo de execução de um DO compreende todos os comandos que vão sofrer iteração. Pode ser apenas dois (mínimo) ou vários comandos.

Exemplos:

```
. . . . . .

. . . . . .

10   DO 11 I=1,10  }
11   A(I)=I*N(I)   }  intervalo do DO

. . . . . .

. . . . . .
```

ou

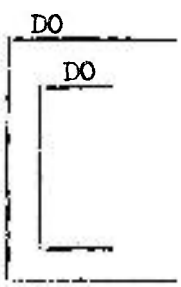


Assim que um comando DO é satisfeito, isto é, a variável i se torna igual ou maior que m_2, o programa tem sua sequência natural até que encontre outro comando de iteração ou desvio.

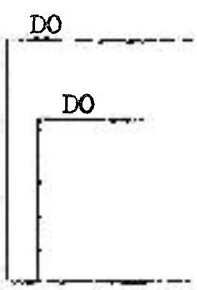
Regras para uso do comando DO

1. O primeiro comando no intervalo de um DO tem que ser um comando executável. Não pode, por exemplo, ser um comando de FORMAT, DIMENSION, etc.

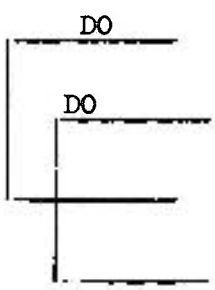
2. É permitido no intervalo de um DO, conter outro(s) DO, não podendo, contudo, se sobreporem, isto é;:



Permissível



Permissível



Não Permissível

3. O último comando do intervalo de um DO não pode ser um comando de transferência, tais como: IF, IF(SENSE SWITCH), GO TO ou GO TO computado, embora possam ser usados livremente no meio do intervalo.

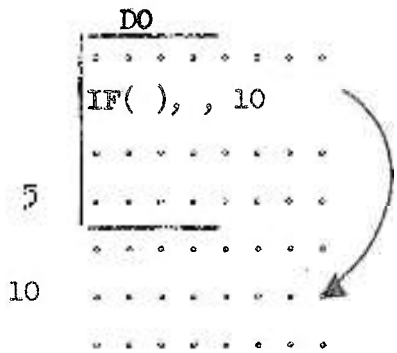
4. Nenhum comando no intervalo do DO pode redefinir ou alterar os parâmetros do DO; isto é, mudar os valores de i, m_1, m_2 e m_3.

5. É permitido sair de dentro do intervalo do DO, por dois processos:

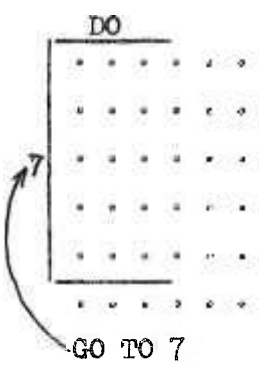
a) saída normal: quando o DO está satisfeito, sendo que i é "maior que m_2". ($\geq m_2$)

b) através de um comando IF ou GO TO, què porventura possa existir dentro do intervalo do DO. Isto facilita a escolha do limite superior do intervalo do DO (em alguns casos desconhecidos) pois, dentro dêste intervalo pode-se colocar uma instrução para testar se a precisão desejada foi obtida.

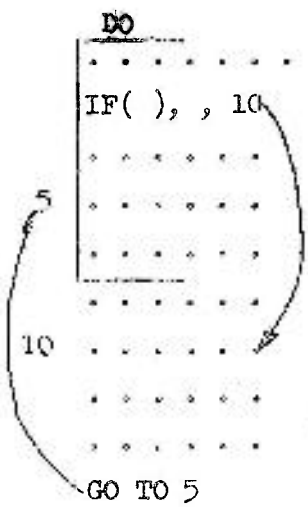
Quando o contrôle é transferido para fora do DO, antes mesmo de ser satisfeito, o índice i fica com o último valor atingido (isto às vêzes é muito valioso). Depois de uma saída normal, i não tem mais valor e pode usar-se outra vez com qualquer nôvo valor inicial.



6. Não é permitido entrar no meio do intervalo do DO. Existe uma única exceção à esta regra: pode-se transferir um comando para o interior de um DO, se anteriormente saiu-se dêste DO.

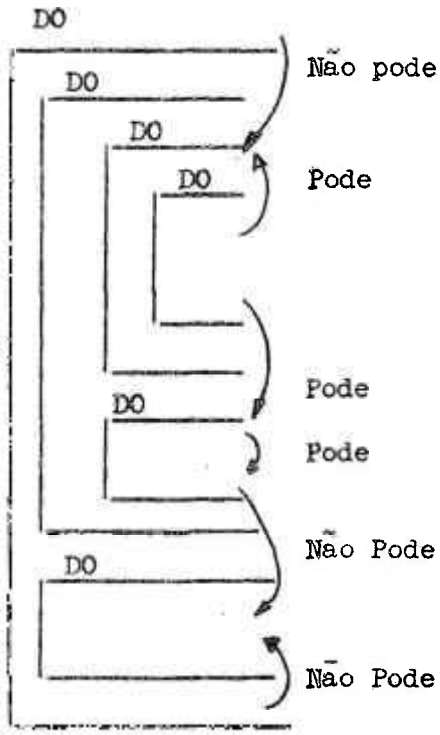


Não é permitido



É permitido

Se tivermos um conjunto de DO embutidos, podemos fazer uma transferência de contrôle de um DO interno para um externo , porém, não vice-versa.



7.10. - Comando CONTINUE

A forma geral é a seguinte:

```
CONTINUE
```

Exemplo:

```
CONTINUE
```

Êste comando não causa nenhuma instrução no Programa Objeto. Onde êle é mais comumente usado é no último comando do DO , quando êste acabar por um comando de transferência (IF ou GO TO). Então, a regra 3 para uso do comando DO fica completa.

Exercício ilustrativo do comando DO

Queremos calcular os produtos dos números inteiros de 1 a M. O produto deve ser expresso em ponto flutuante, por razões de capacidade.

```
      PROD = 1.
      M = 1000
      DO 10 I = 2,M
      AI = I
   10 PROD = PROD*AI
      . . . . . .
```

Êste exemplo, mostra que o índice I pode começar por uma constante de ponto fixo diferente de 1.

Exemplo ilustrativo dos comandos DO e CONTINUE

```
      . . . . . .
      A = 0.3
      DO 12 L = 1,15
      B = A**L
      IF(B-0.43257)16,16,16
   12 CONTINUE
      . . . . . .
      . . . . . .
   16 STOP
      END
```

EXERCÍCIOS

1) Calcular a expressão matemática:

$$x = \frac{y^2 + z^2}{c} \quad e \; a$$

onde devemos obter os valores de x, atribuindo valores a, y e z, de acôrdo com os intervalos abaixo, e fixando os valores de a e c, até que x seja maior ou igual a 10^6.

Variação de y: de 0 à 1000 de 2 em 2
Variação de z: $z_o = 1$ e $z = 10$.

2) São lidos os valores de A1, A2, A3 e A4 de um cartão. Êstes valores são quaisquer e estão em ordem qualquer. Faça um programa que imprima êstes 4 valores na ordem decrescente de seus valores.

3) Um ângulo chamado THETA é conhecido ser positivo e menor que 30 radianos. Subtraia 2π de THETA quantas fôr necessário até reduzir a um ângulo menor que 2π. Deixe o valor reduzido em THETA.

4) Se $0.999 \leq x \leq 1.001$ transfira para o comando 63, senão transfira para o comando 67.

5) Y é para ser calculado como função de X, sendo X compreendido entre 1.0 e 9.9 em intervalos de 0.1, onde

$$Y = 16.7\ x + 9.2\ x^2 - 1.02\ x^3$$

Imprima X e Y para cada um dos valores de X.

CAPÍTULO VIII

Sub-rotinas, Sub-programas e Funções

8.1. - Sub-rotinas

Já vimos que um programa ou código, ou ROTINA, consiste em um conjunto de instruções arranjado numa sequência lógica que permitirá a solução de um problema determinado. Geralmente, um programa contém uma curta sequência de instruções, isto é, sub-conjuntos que formam parte do programa completo, e que são usa -

dos para resolver uma parte particular do problema. Estas partes do programa ou rotina são chamadas SUB-ROTINAS.

Usualmente, uma sub-rotina realiza uma função específica, que é comum a vários programas e pode ser executada várias vêzes durante o curso do programa do qual forma parte.

Um eficiente procedimento de programação é òbviamente aquêle no qual tôdas as sub-rotinas necessárias são codificadas só uma vez e são incorporadas ao programa sempre que sejam requeridas.

Foi desenvolvido, para o FORTRAN II-D, um grupo de sub-rotinas que são mais frequentemente requeridas pela sua aplicação geral, e são fornecidas como parte do sistema, em linguagem de máquina. Estas sub-rotinas podem ser do tipo aberto ou fechado, dependendo do mecanismo de inserção no programa objeto.

As sub-rotinas abertas requerem sòmente umas poucas instruções de máquina, e são inseridas no programa objeto cada vez que é necessária sua utilização.

As sub-rotinas fechadas são, em geral, consideràvelmente mais longas, e são inseridas no programa objeto em continuação ao programa principal, e sub-programas correspondentes.

As sub-rotinas podem classificar-se em duas categorias:

1) - Sub-rotinas aritméticas e de entrada e saída;
2) - Sub-rotinas de biblioteca.

As sub-rotinas aritméticas e de entrada e saída são sub-rotinas básicas necessárias para a própria execução do programa objeto, e são carregadas sem que sejam especìficamente chamadas.

As sub-rotinas de biblioteca do FORTRAN II-D são 16 sub-rotinas "recolocáveis", e que são carregadas só quando são requeridas, mas antes da execução do programa objeto.

8.2. - Funções de Biblioteca

Dentre as sub-rotinas de biblioteca mencionadas em 8.1, temos sete sub-rotinas que chamamos Funções de Biblioteca, e que são:

Logarítmo natural
Seno trigonométrico
Coseno trigonométrico
Exponencial
Raiz quadrada
Arco Tangente
Valor absoluto

Tôdas estas funções possuem duas entradas, isto é, cada uma delas é posta em função escrevendo-se um dos dois nomes seguido dos argumentos entre parênteses.

Os nomes ou entradas são respectivamente:

LOGF ou LOG
SINF ou SIN
COSF ou COS
EXPF ou EXP
SQRTF ou SQRT
ATANF ou ATAN
ABSF ou ABS

O argumento pode ser uma variável (subscritada ou não), ou uma expressão.

Exemplos:

```
A = COSF (B)
Y = A - SIN (B*SQRT(C))
```

Suponhamos querer calcular a expressão seguinte:

$$V = \frac{1}{\cos X} + \log (\mathrm{Tg}\ X/2)$$

O comando será:

```
V = 1./COSF(X) + LOGF(SINF(X/2.)/COSF(X/2.))
```

Como vemos, é permissível que o argumento de uma função contenha outra função. Os erros que se comentem nas funções de biblioteca não são, em geral, maiores que 1 na última casa da mantissa.

O logarítmo natural deve ter argumento positivo e é calculado por desenvolvimento em série.

A função arc tang. dá o resultado em radianos. O resultado da raiz quadrada tem uma precisão de ± 1 no último dígito da mantissa. As funções seno e coseno devem ter argumento em radianos.

A expressão A^B é calculada como EXPF(B*LOGF(A)). São utilizadas três sub-rotinas, logarítmo, multiplicação e exponenciação. Um êrro numa destas sub-rotinas pode propagar-se nas outras sub-rotinas. Normalmente a magnitude do êrro não deve exceder ... 10^{1-f}.

8.2.1. - Funções de Biblioteca Adicionais

Até 14 funções adicionais podem ser acrescentadas às sub-rotinas de biblioteca. Estas funções devem ser definidas em linguagem de máquina ou em linguagem SPS e ser carregadas ao tempo da montagem ou mais tarde, utilizando o Programa Utilidade de Disco.

A forma geral é:

```
NAME (A)
```

onde, NAME é o nome da função que pode ser a combinação de 1 a 6 caracteres alfabéticos ou numéricos, devendo ser o primeiro caracter sempre alfabético, não podendo incluir caracteres especiais. A é o argumento colocado entre parênteses.

Exemplo:

```
TIME (A)
```

Observações:

1) As funções de biblioteca produzem um único valor para um determinado argumento.

2) O modo é determinado pelo argumento. Exemplo:

COSH(A) - ponto flutuante
SINH(I) - ponto fixo

3) Qualquer uma das funções de biblioteca é chamada só com o comparecimento do nome da função e argumento correspondente, numa expressão aritmética.

4) As funções de biblioteca fornecidas com o sistema, para o FORTRAN II-D, só admitem argumentos de ponto flutuante, com exceção da função ABSF.

8.3. - Funções Aritméticas

Estas funções são definidas (escritas) por um único comando aritmético em FORTRAN II-D e se aplicam sòmente a programas particulares nos quais elas aparecem. O nome da função que consiste de 1 a 6 caracteres (não especiais) devendo o primeiro ser alfabético, é seguido dos argumentos entre parênteses e separados por vírgulas, quando forem mais que um.

Exemplo:

```
FIRS  (X)
SRTH  (F,G)
```

O comando da função aritmética tem a forma geral seguinte:

```
NAME (ARG) = E
```

onde, NAME é o nome da função seguido pelo(s) seu(s) argumento(s), (os quais terão que ser variáveis distintas e não subscritadas)separados por vírgulas.

E é uma expressão que não pode envolver uma variável subscritada. Qualquer função que apareça em E tem que ser disponível para o programa e ser definida num comando precedente ao seu uso.

Exemplos:

```
FRST (X) = A*X+B
SCDN (X,B) = A*X+B
THRD (D) = FRST (E)/D
FRTH (F,G) = SCDN (F,THRD(G))
FFTH (I,A) = 3.0*A**I
SXTH (J) = J+K
```

Observação:

No comando de definição da função aritmética, o argumento não pode ser subscritado, entretanto, quando se utilizar a instrução no programa, pode.

Exemplo:

```
DIMENSION X(3)
F(Z)=A*Z+B      comando de definição (não é permitido subscritos)
READ 12,A,B,X(1),X(2),X(3)
W=X(2) **.5
DELTA = F(X(3)) comando de utilização (é permitido subscritos)
```

Durante o cálculo, os vários X apresentam o Z definido na função aritmética.

Como nas funções de biblioteca, o próprio nome da função serve para "chamá-la" à utilização no programa. O exemplo acima ilustra o que acabamos de dizer, pois no 5º comando, "chamamos" a função F(X(3)), que está definida no 2º comando, como F(Z).

Para cada comando de função aritmética, sòmente um va-

lor é produzido, e nenhuma função pode ser usada como argumento dela mesma.

Se FRSTF é definido como uma função aritmética, por:

FRSTF(X) = A*X+B

então, quando nos referirmos no programa a FRSTF(Y), será produzida a operação ay + b, e computado o seu valor de acôrdo com os valores em curso de a, b, e y.

Do exposto, conclui-se que se tivermos um tipo de função a ser calculada diversas vêzes para argumentos diferentes, define-se esta função genèricamente num comando de função aritmética precedente à sua utilização.

Os argumentos de uma função definida por um comando de função aritmética, são variáveis mudas e portanto inoperantes (entretanto, indicam se o argumento é de ponto fixo ou flutuante), e assumirão o valor indicado no comando de utilização que chama essa função.

FRSTF (Z+Y(I))

como resultado de uma prévia definição de FRSTF, causará a operação $a(z+y_i)+b$ que será computado como base nos valores em curso de a, b, y_i e z.

O modo do valor da função é determinado pelo nome da função. Se o nome começa com I,J,K,L,M,N, o valor será de ponto fixo; em caso contrário, o valor será de ponto flutuante.

Todos os comandos de funções aritméticas, se um programa os tiver, deverão preceder o primeiro comando executável do programa.

8.4. - Comandos de Sub-programas

Um sub-programa é definido como um programa escrito em linguagem FORTRAN e é utilizado por outro programa fonte FORTRAN.

Dois tipos de sub-programas são disponíveis no FORTRAN II-D: O sub-programa FUNCTION e o sub-programa SUBROUTINE. Para definir êstes sub-programas são necessários quatro comandos:

FUNCTION, SUBROUTINE, RETURN e CALL

A utilidade dos sub-programas baseia-se em que, enquanto as funções aritméticas estão limitadas a um único comando e calculam um valor só, o sub-programa FUNCTION elimina a primeira limitação e o sub-programa SUBROUTINE elimina ambas as limitações; e, principalmente, porque são sub-programas, e portanto são compilados separadamente do programa principal de que fazem parte ; os nomes das variáveis são totalmente independentes dos nomes das variáveis usadas no programa principal e outros sub-programas, porém é bastante simples estabelecer "comunicação" entre o programa principal e os sub-programas. Com isto um programa muito longo pode ser dividido em partes que podem ser compiladas individualmente, tornando possível corrigir erros no sub-programa, sem a necessidade de compilar novamente o programa principal.

Devemos distinguir cuidadosamente entre a definição e o uso.

Vejamos cada um dêstes comandos.

8.5. - Comando FUNCTION

Êste comando define o sub-programa como uma Função Fortran, e é sempre o primeiro comando do sub-programa.

A forma geral é a seguinte:

FUNCTION Name $(a_1,a_2,\ldots a_n)$

onde, Name é o nome simbólico da função, e $a_1,a_2,\ldots,a_n$ são os argumentos, variáveis não subscritadas, devendo existir ao menos um dêles. O nome consiste de 1 a 6 caracteres alfabéticos ou numéricos (não especiais), devendo ser o primeiro alfabético.

Exemplos:

```
FUNCTION ARCSN (RADS)
FUNCTION ROOT (B,A,C)
FUNCTION Y (X)
```

O nome da função deve aparecer numa lista de variáveis de um comando de entrada, ou pelo menos uma vez como variável do lado esquerdo de um comando aritmético.

Exemplos:

```
      FUNCTION CALC(X)
      IF(X) 10,11,12
   10 CALC = 1. * SQRTF (1.*X*X)
      RETURN
   11 CALC = 0
      RETURN
   12 CALC = 1.* SQRTF (1.*X*X)
      RETURN
      END

      FUNCTION EXEM(A,B,X)
      READ 10, N,EXEM, R,S
   10 FORMAT (I4,3F8.2)
      . . . . .
      . . . . .
      RETURN
      END
```

O modo da função é determinado pelo nome.

Exemplos:

```
FUNCTION AMAST (A,K) - ponto flutuante.
FUNCTION IAMAST(A,K) - ponto fixo.
```

Os argumentos colocados em seguida ao nome, na definição

da função, são nomes de variáveis mudas, serão substituidos durante a execução do programa objeto pelos argumentos reais. Êstes argumentos devem coincidir em número, ordem e modo com os da definição. Se o argumento mudo é o nome de um arranjo, o correspondente argumento real também deve ser o nome de um arranjo, e êles devem aparecer na lista de comandos DIMENSION similares, em ambos os programas.

Nenhuma das variáveis mudas deve figurar na lista de um comando EQUIVALENCE dentro do sub-programa FUNCTION.

Como nas funções de biblioteca e aritméticas, o aparecimento do nome da Função Fortran serve para "chamá-la" à utilização no programa, o qual fornece os argumentos reais que serão computados. A Função Fortran fornecerá então um único valor como resultado.

Exemplo:

Suponhamos querer achar o produto dos elementos da diagonal principal de um arranjo quadrado, o que será necessário calcular várias vêzes no transcurso de um programa. Consideremos também que o maior arranjo será de 10 x 10, mas os arranjos podem ser menores; entretanto, sua dimensão deve ser dada na forma de uma variável de ponto fixo.

O programa será o seguinte:

```
      FUNCTION DIAGPR (A,N)
      DIMENSION A(10,10)
      DIAGPR = A(1,1)
      DO 69 I = 2,N
   69 DIAGPR = DIAGPR * A(I,I)
      RETURN
      END
```

Se desejarmos agora o produto dos elementos da diagonal principal de um arranjo chamado DATA, em que a dimensão real é

dada pelo valor de LAST, podemos escrever no programa principal:

```
DET = DIAGPR (DATA,LAST)
```

Para obter o quadrado do produto dos elementos da diagonal principal do arranjo denominado X, em que o número das colunas e linhas está dado pelo valor de JACK, coloca-se no programa principal o comando.

```
EIG = DIAGPR (X, JACK) **2
```

O fato de aparecer o nome do sub-programa no programa principal, com os argumentos adequados, será suficiente para que o fluxo natural da execução desvie para o início do sub-programa, começando o cômputo, e retornará ao comando seguinte ao de chamada quando execute as instruções equivalentes ao comando RETURN;

8.6. - Comando SUBROUTINE

A forma geral dêste comando é:

```
SUBROUTINE Name (a1,a2,a3,...,an)
```

onde, Name é o nome simbólico do sub-programa e cada um dos argumentos a_1, a_2,...,a_n, se existe, é o nome de uma variável não subscritada. A formação do nome é, como sempre, de ATÉ 6 caracteres alfabéticos e/ou numéricos (não especiais), devendo o primeiro ser alfabético.

Exemplos:

```
SUBROUTINE DIAGPR (A,N)
SUBROUTINE MATP (B,A,C,ROOT1, X2)
```

O comando SUBROUTINE define um subprograma, e deve ser o primeiro comando dêsse sub-programa. Êste tipo de sub-programa pode conter qualquer comando FORTRAN II-D, exceto FUNCTION, DEFINE DISK, ou outro comando SUBROUTINE;

O programa principal "chama" o sub-programa SUBROUTINE por meio do comando CALL, que especifica o nome do sub-programa e seus argumentos.

Os argumentos do sub-programa são nomes de variáveis mudas que adquirem o valor determinado no momento da execução, utilizando os valores fornecidos pelo comando de chamada. Portanto, deve haver correspondência em número, ordem e modo entre os dois conjuntos de argumentos.

Além disso, se o argumento mudo é o nome de um arranjo , o correspondente argumento real também deve ser nome de um arranjo, e portanto, ambos devem aparecer em listas de comandos DIMENSION, similares nos respectivos programas.

Exemplo:

Se o sub-programa é encabeçado pelo comando SUBROUTINE MATMP (A,N,M,B,L,C), entrará em ação quando no programa principal aparecer o comando

CALL MATMP (X,5, 0,Y, 7,Z)

Se as variáveis mudas A,B,C, são nomes de arranjos, o sub-programa possuirá um comando DIMENSION com as dimensões de A, B,C, as variáveis reais X,Y,Z devem ser também nomes de arranjos, e aparecerão numa lista do comando DIMENSION do programa principal, que especificará as mesmas dimensões que no sub-programa.

Nenhuma das variáveis mudas pode aparecer num comando EQUIVALENCE do sub-programa.

Um sub-programa SUBROUTINE pode não ter argumentos especificados em continuação ao nome. Neste caso, tanto as variáveis mudas como as reais, devem aparecer num comando COMMON dentro dos respectivos programas.

Os sub-programas devem ser compilados em separado,ou formarem parte de uma compilação múltipla, com outros, mas sempre co

mo um programa independente.

8.7. - Comando CALL

A forma geral é:

CALL Name $(a_1,a_2,\ldots a_n)$

onde, Name é o nome de um sub-programa SUBROUTINE e $a_1,a_2,\ldots a_n$ são os argumentos. O comando CALL transfere o contrôle do fluxo de execução ao sub-programa, com os valores dos argumentos especificados neste comando de chamada, ou que estão na área COMMON.

Cada um dos argumentos pode ser de qualquer dos tipos seguintes:

1) - Constante de ponto fixo
2) - Constante de ponto flutuante
3) - Variável de ponto fixo com ou sem subscrito
4) - Variável de ponto flutuante com ou sem subscrito
5) - Expressão aritmética.

Os argumentos apresentados pelo comando CALL devem corresponder-se em número, ordem, modo e dimensões de arranjos, com os argumentos mudos do comando SUBROUTINE do sub-programa chamado. Nenhum argumento pode ter o mesmo nome que o sub-programa que é chamado.

Exemplos:

```
CALL MATMP (X,5,10,Y,7,Z)
CALL QDRT (P*9,732,Q/4.536,R-S*2.0,X1,X2)
```

O contrôle retornará ao fluxo normal do programa principal, quando no sub-programa sejam executadas as instruções correspondentes ao comando RETURN, voltando ao comando seguinte àquele de chamada.

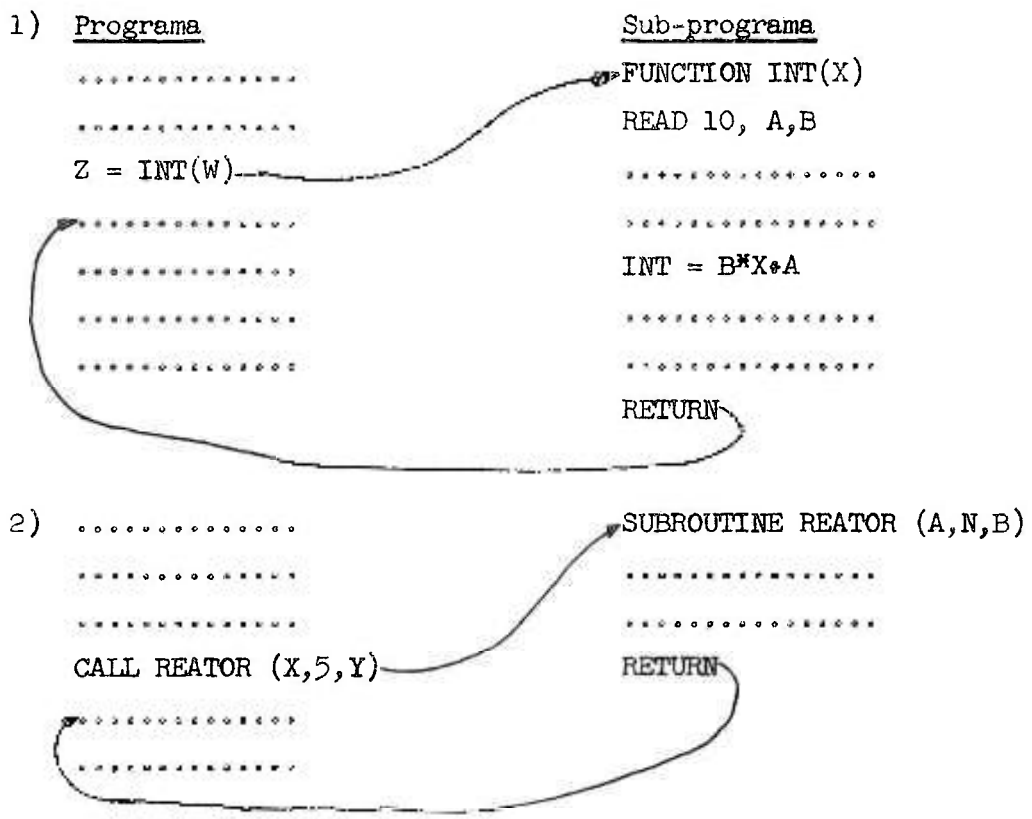
8.8. - Comando RETURN

A forma geral é a seguinte:

```
RETURN
```

Êste comando deve ser o último comando executável de um sub-programa, seja do tipo encabeçado pelo comando **FUNCTION**, ou pelo comando SUBROUTINE. Êle faz com que se retorne ao programa principal no comando imediatamente seguinte ao comando que "chamou" o sub-programa.

Esquematizando, temos:



O comando RETURN não precisa ser o último comando do sub-programa, fìsicamente, mas deve ser o último comando alcançado seguindo um caminho de execução.

Dentro do sub-programa pode ser usado qualquer número

de comandos RETURN.

Exemplo:

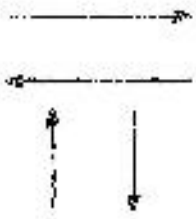
```
      SUBROUTINE OUTPUT
      COMMON A,B,C,X1R,X1I,X2R,X2I
      IF (X1I) 90,91,90
   91 TYPE 95, A,B,C,X1R,X2R
   95 FORMAT (4E12.4,E15.4)
      RETURN
   90 IF(X1R) 100,101,100
  101 IF(X2R) 100,102,100
  102 TYPE 103,A,B,C,X1I,X2I
  103 FORMAT (3E12.4,2E15.4)
      RETURN
  100 TYPE 110, A,B,C,X1R,X1I,X2R,X2I
  110 FORMAT (3E12.4//4E15.4)
      RETURN
      END
```

CAPÍTULO IX

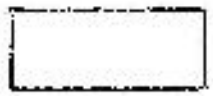
9.1. - Diagrama de Blocos

O diagrama de blocos, também chamado fluxo-grama, carta de fluxo ou organograma é uma técnica usada para esquematizar os passos de um programa. O diagrama de bloco tem por finalidade visualizar a lógica do programa e deve ser feito antes de escrevermos o programa em linguagem FORTRAN.

Os símbolos usados são os seguintes:

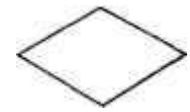


Direção de escoamento

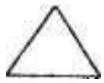


Um processamento

Entrada/Saída



Decisão



Decisão usando chave de programa

Conexão entre dois pontos num programa, onde uma linha iria dificultar a visualização.

Ponto de iniciação ou de parada de um programa.

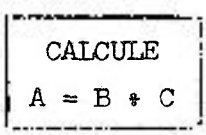
A direção de escoamento visualiza a relação entre um símbolo e outro.

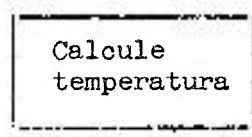
Exemplo:

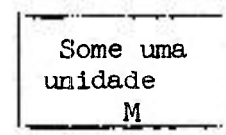
A figura acima mostra que A é executado e depois B é executado.

O símbolo de um processamento é usado para representar qualquer instrução do programa, (exceto decisão lógica) as quais não são representadas por símbolos especiais.

Exemplos:

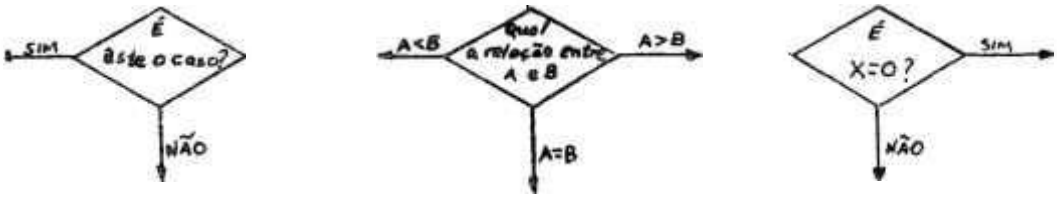






O símbolo de decisão representa qualquer decisão lógica que está contida no programa.

Exemplos:



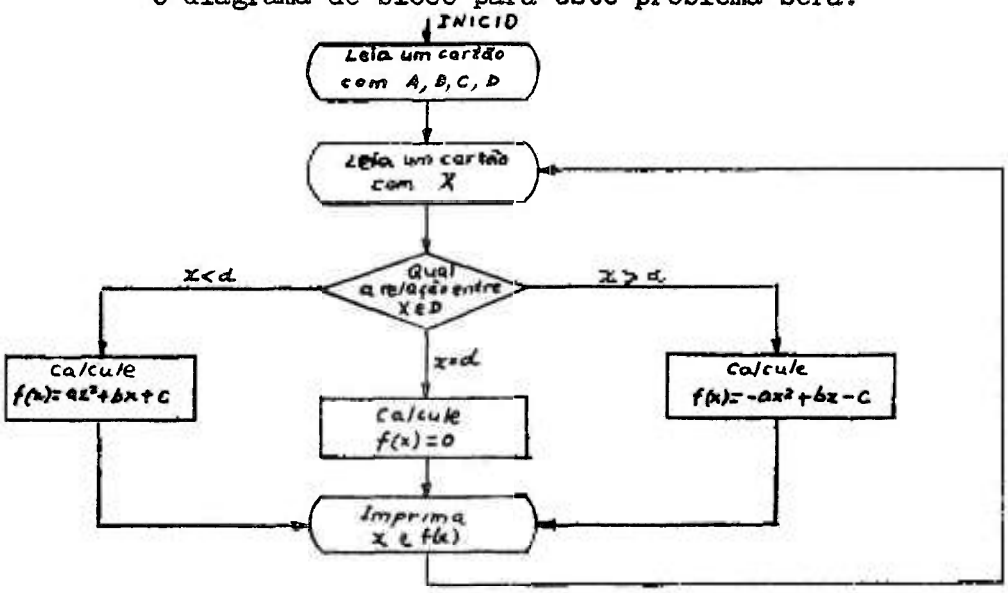
9.2. - Exemplo:

Dados os valores de a, b, c e d perfurados em um cartão, e um conjunto de valores para a variável x perfurados um por cartão, calcular a função definida por:

$$f(x) = \begin{cases} ax^2 + bx + c & \text{se } x < d \\ 0 & \text{se } x = d \\ -ax^2 + bx - c & \text{se } x > d \end{cases}$$

para cada valor de x, e imprime x e f(x).

O diagrama de bloco para êste problema será:



O programa FORTRAN, será:

```
C       PROBLEMA DA FUNÇÃO DE X
  100   READ 1,A,B,C,D
    6   READ 1,X
  101   IF(X-D)2,3,4
    2   FUNX = A*X**2+B*X+C
  102   GO TO 5
    3   FUNX = 0.0
  103   GO TO 5
    4   FUNX = -A*X**2+B*X-C
    5   TYPE  1,X,FUNX
  104   GO TO 6
    1   FORMAT (4F14.5)
```

Uma saída típica onde A = 10.0, B = 11.0, C = 12.0 , D = 13.0, será:

```
    9.50000      1018.99996
   10.00000      1121.99988
   10.50000      1229.99985
   11.00000      1342.99986
   11.50000      1460.99986
   12.00000      1583.99988
   12.50000      1711.99988
   13.00000         0.
   13.50000     -1685.99977
   14.00000     -1817.99966
   14.50000     -1954.99973
   15.00000     -2096.99982
```

O diagrama bloco, além da ajuda de fornecer ao programador uma visualização fácil das inter relações dentro do programa, facilita enormemente o intercâmbio entre programadores , sendo portanto, uma parte muito valiosa na documentação de um

programa.

CAPÍTULO X

Exercícios resolvidos e por resolver

10.1. - Exercícios resolvidos

10.1.1. - Suponhamos um conjunto de 20 cartões, nos quais temos 100 elementos de um arranjo de nome X, sendo que estão perfurados cinco quantidades em cada cartão. Temos, então:

```
      DIMENSION X(100)
      I=1
    2 READ 10, X(I),X(I+1),X(I+2),X(I+3),X(I+4)
   10 FORMAT (5F10.0)
      I = I+5
      IF(I-96)2,2,3
    3 continuação do programa
```

Também se poderia escrever o mesmo programa assim:

```
      DIMENSION X(100)
      READ 10,(X(I),I=1,100)
   10 FORMAT (5F10.0)
      continuação do programa.
```

Se tivermos o mesmo problema, sendo agora uma quantidade perfurada em cada cartão, isto é, 100 cartões, teremos:

```
      DIMENSION X(100)
      I=1
    2 READ 10,X(I)
   10 FORMAT(F10.0)
      I=I+1
      IF(I-100)2,2,3
    3 continuação do programa
```

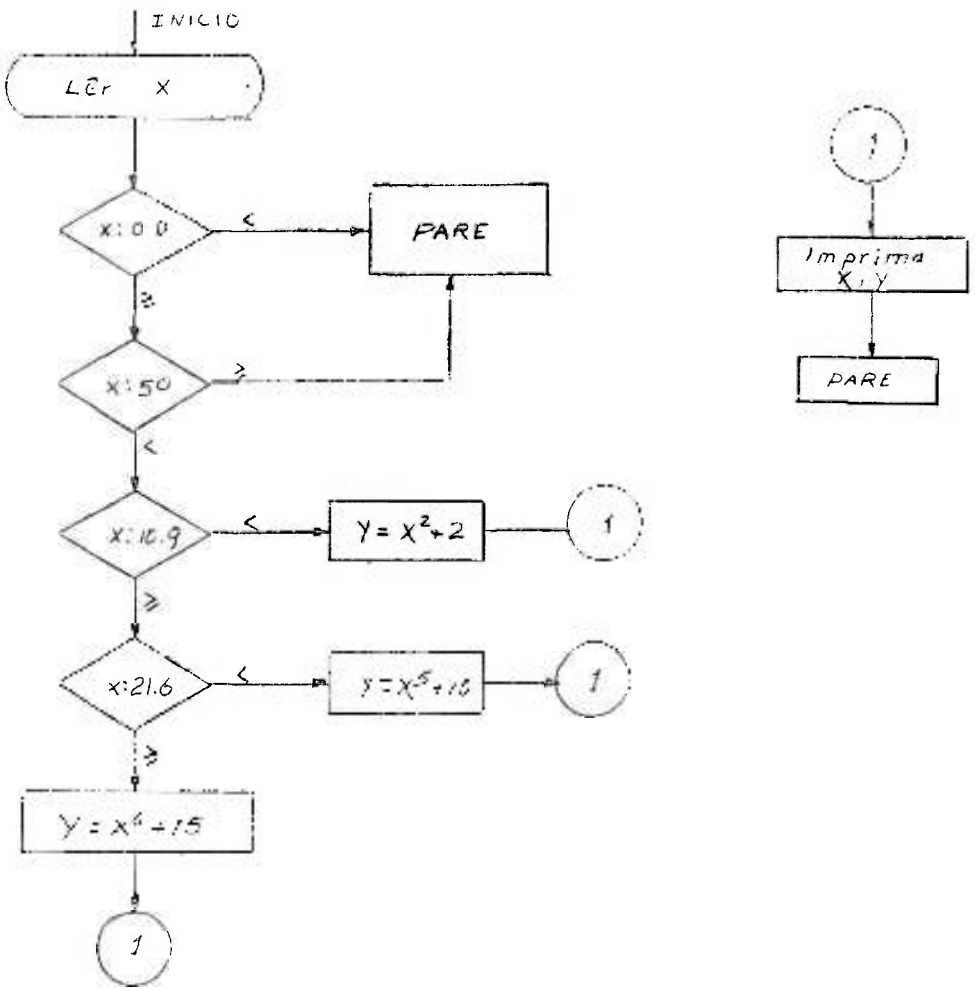
10.1.2. - Suponhamos o problema em que desejamos ler um valor de X, conferí-lo, e computar o valor de Y de acôrdo com o valor de X.

Assim:

$$Y = \begin{cases} X^2 + 2 & \text{se } 0.0 \leq X < 10.9 \\ X^5 + 10 & \text{se } 10.9 \leq X < 21.6 \\ X^6 + 15 & \text{se } 21.6 \leq X < 50 \end{cases}$$

Se $X < 0.0$ e $X \geq 50$ pare não calcule nada.

Diagrama de Bloco:

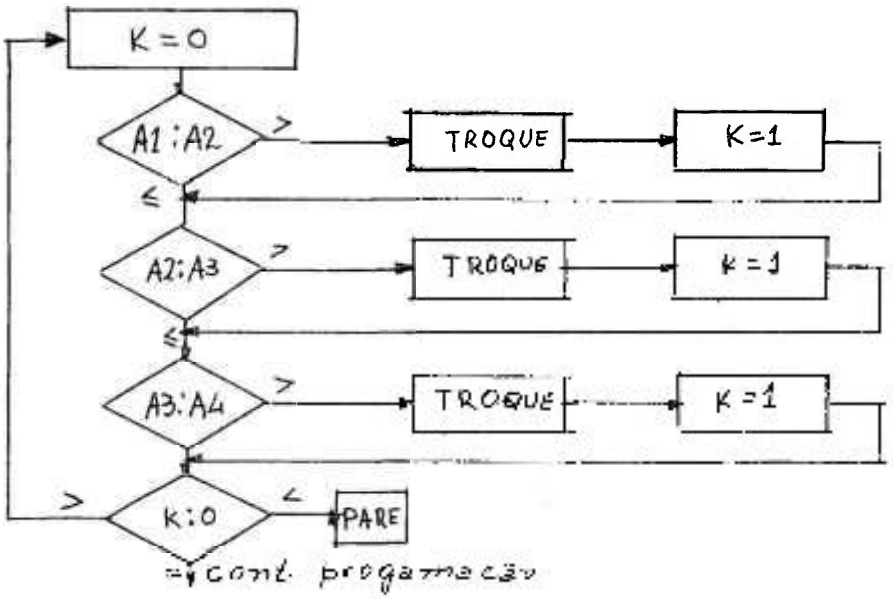


Programa FORTRAN:

```
      READ   18,X
      IF(X) 20,21,21
21    IF(X-50.)22,20,20
22    IF(X-10.9)23,24,24
23    Y=X**2*2.
      GO TO 25
24    IF(X-21.6)26,27,27
26    Y=X**5*10.
      GO TO 25
27    Y=X**6*15.
25    TYPE 28,X,Y
18    FORMAT(F10.0)
28    FORMAT(2E14.8)
20    STOP
      END
```

10.1.3. - Temos 4 valores para A1,A2,A3 e A4 e queremos fazer um programa que coloque na ordem crescente tais valores, ficando o menor em A1 e o maior em A4.

Diagrama de Bloco:



```
      Programa FORTRAN
      . . . . . .
    2 K=0
      IF(A1-A2)11,11,10
   10 AUXI=A1
      A1=A2
      A2=AUXI
      K=1
   11 IF(A2-A3)13,13,12
   12 AUXI=A2
      A2=A3
      A3=AUXI
      K=1
   13 IF(A3-A4)15,15,14
   14 AUXI=A3
      A3=A4
      A4=AUXI
      K=1
   15 IF(K)16,17,2
   16 STOP
   17 continuação do programa
```

10.1.4. - Solução de multiplicação de matrizes

Consideremos o produto abaixo:

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ & & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & & \\ a_{1001} & a_{1002} & a_{1003} & a_{1004} & a_{1005} \end{vmatrix} \times \begin{vmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \end{vmatrix} = \begin{vmatrix} b_1 \\ b_2 \\ \cdots \\ \cdots \\ b_{100} \end{vmatrix}$$

Simbòlicamente, temos:

$$A(I,J) \times X(J) = B(I)$$

Apliquemos o algorítmo:

$$B(I) = \sum_{J=1}^{5} \left[A(I,J) \times X(J) \right]$$

Fazendo $I = 1,2,\ldots,100$ e $J = 1,2,\ldots,5$, teremos:

$I=1 \quad J=1,2,\ldots,5 \qquad B(1) = \sum_{J=1}^{5} A(1,J) \times X(J)$

$I=2 \quad J=1,2,\ldots,5 \qquad B(2) = \sum_{J=1}^{5} A(2,J) \times X(J)$

. . . . . .

. . . . . .

$I=100 \quad J=1,2,\ldots,5 \qquad B(100) = \sum_{J=1}^{5} A(100,J) \times X(J)$

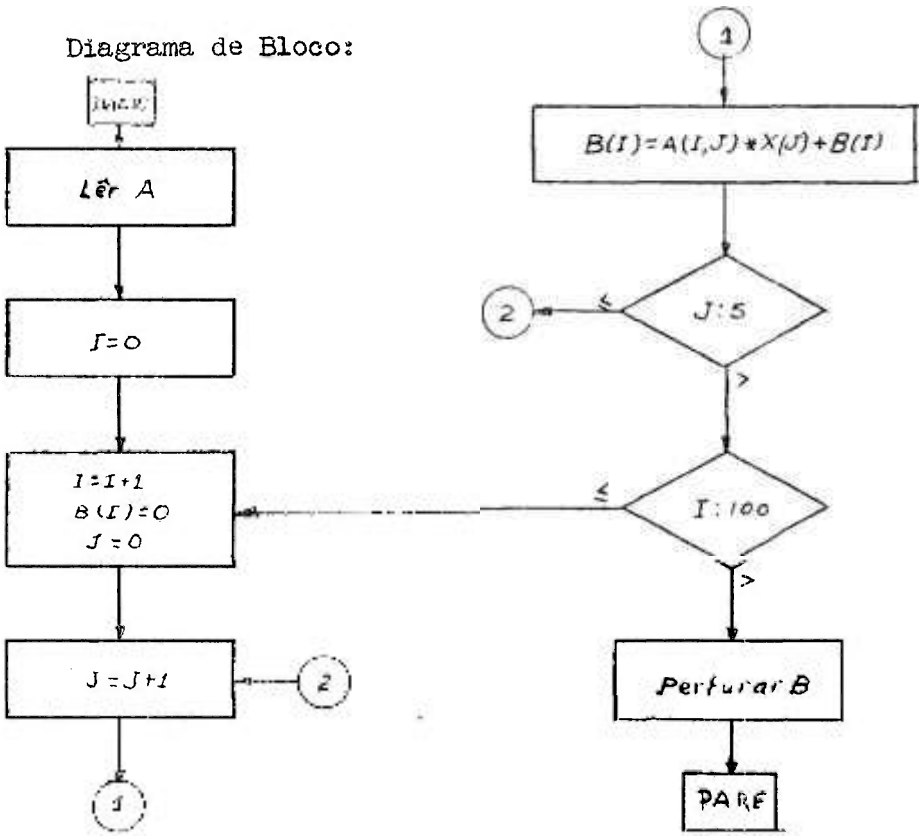
Na programação é necessário ter um conjunto de cartões para a matriz A(I,J), que constará de 100 cartões com 5 valores em cada um. Para a matriz coluna X(J), teremos um cartão separado, também com 5 valores. Obteremos como resultado da multiplicação, a matriz B(I), como um conjunto de 20 cartões com 5 valores em cada um.

Programa FORTRAN

```
      DIMENSION  A(100,5),X(5),B(100)
      I=0
    3 I=I + 1
      READ 10,A(I,1),A(I,2),A(I,3),A(I,4),A(I,5)
   10 FORMAT(5F10.0)
      IF(I-100)3,15,15
```

```
15 READ 10,X(1),X(2),X(3),X(4),X(5)
   I=0
 2 I=I+1
   B(I)=0
   J=0
 6 J=J+1
   B(I)=A(I,J) * X(J) +B(I)
   IF(J-5)6,20,20
20 IF(I-100)2,21,21
21 I=1
27 PUNCH 22,B(I),B(I+1),B(I+2),B(I+3),B(I+4)
22 FORMAT(5E14.8)
   I=I+5
   IF(I-96)27,27,30
30 STOP
   END
```

Diagrama de Bloco:



10.1.5. - Suponhamos que um conjunto denominado DADOS contém 100 elementos (uma dimensão). Queremos determinar a soma de todos os elementos com numeração impar, isto é:

DADOS(1) + DADOS(3) + DADOS(5)+ ... + DADOS(99)

Programa:

```
      DIMENSION DADOS (100)
      SOMA=0.0
      DO 20 I=1,99,2
   20 SOMA=SOMA + DADOS(I)
      . . . . . .
```

10.1.6. - Seja um laboratório no qual se esteja medindo determinado fenômeno. Para cada valor de X da medição corresponde um valor de Y. Os resultados são perfurados aos pares, constituindo conjuntos de cartões que chegam aleatòriamente a um Centro de Processamento de Dados. Deseja-se:

a) ordenar os valores de X, na ordem crescente;
b) proceder a integração aproximada da área;
c) imprimir o valor da área

EXPERIÊNCIA 1	EXPERIÊNCIA 2	. . . . .	EXPERIÊNCIA N
X1 ..... X10	X11 ..... X25	. . . . .	X950 ..... X1000
Y1 ..... Y10	Y11 ..... Y25	. . . . .	Y950 ..... Y1000

Supondo que os intervalos de X sejam constantes, podemos usar a regra parabólica para a integração aproximada, isto é:

$$AREA = \int_{X1}^{X1000} y \, d\, x = \frac{H}{3} (y_1 + 4y_2 + 2y_3 + 4y_4 + \ldots + 4y_{998} + 2y_{999} + y_{1000})$$

Teremos a distribuição abaixo:

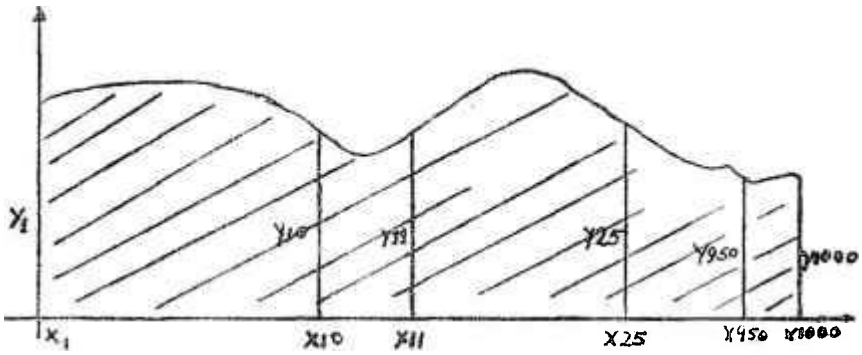
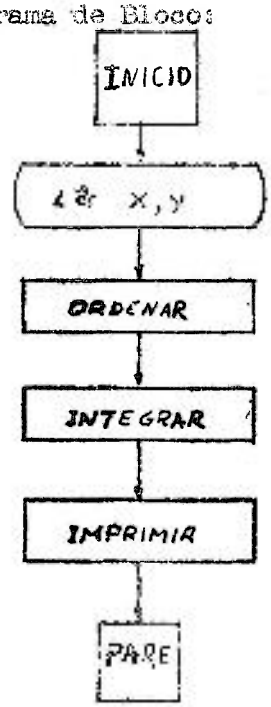


Diagrama de Bloco:



Programa FORTRAN:

```
      DIMENSION X(1000),Y(1000)
      DO 10 I=1,1000
   10 READ 5,X(I),Y(I)
    5 FORMAT(2F10.0)
    7 K=0
      DO 30 I=1,999
      IF(X(I)-X(I + 1)) 30,30,20
   20 AUXI=X(I)
```

```
      X(I)= X(I + 1)
      X(I + 1)= AUXI
      AUXI=Y(I)
      Y(I)=Y(I + 1)
      Y(I+1)= AUXI
      K=1
   30 CONTINUE
      IF(K)31,35,7
   31 STOP
   35 XIMP=0.0
      PAR=0.0
      DO 47 I=2,998,2
   47 PAR=PAR + Y(I)
      DO 48 I=3,999,2
   48 XIMP=XIMP + Y(I)
      XH=(X(2)-X(1))/3.
      AREA=XH*(Y(1) + 4.* PAR + 2.* XIMP + Y(1000))
      PRINT 50, AREA
      FORMAT (1HO,E14.8)
      STOP
      END
```

10.1.7. - Um maço de cartões é dado para ser lido no formato (I3.F10.2); um cartão para cada pessoa de uma cidade. O número do primeiro campo é a idade da pessoa e o número do segundo campo é seu salário para 1964. Seguindo êste maço de cartões, está um cartão com 1 no primeiro campo; esta informação será usada para testar o fim do maço.

Calcule o salário médio para grupos de pessoas em cada 5 anos de idade, isto é, 0 - 4,5-9,10-14,...,95-99. Imprima o limite maior de idade para cada grupo, isto é, 0,5,10,...,95, o salário médio para êste grupo, e o número de pessoas em cada grupo. Precauções deverão ser tomadas para se evitar divisão por zero.

Programa FORTRAN

```
C     PROBLEMA DA POPULAÇÃO E SALÁRIO
  3   FORMAT(I3,F10.2,F7.0)
100   DIMENSION P(20),S(20)
101   DO 9 N=1,20
102   P(N)=0.0
  9   S(N)=0.0
  1   READ 3,K,SAL
103   IF(K * 1)8,4,2
  2   N=K/5 * 1
104   P(N)=P(N) * 1.0
105   S(N)=S(N) * SAL
106   GO TO 1
  4   DO 7 N=1,20
107   IF(P(N)) 8,6,5
  5   S(N)=S(N)/P(N)
  6   K=5 * N-5
  7   PRINT 33,K,S(N),P(N)
 33   FORMAT (1H ,I3, F10.2, F7.0)
  8   STOP
      END   (Diagrama de Bloco no final do trabalho)
```

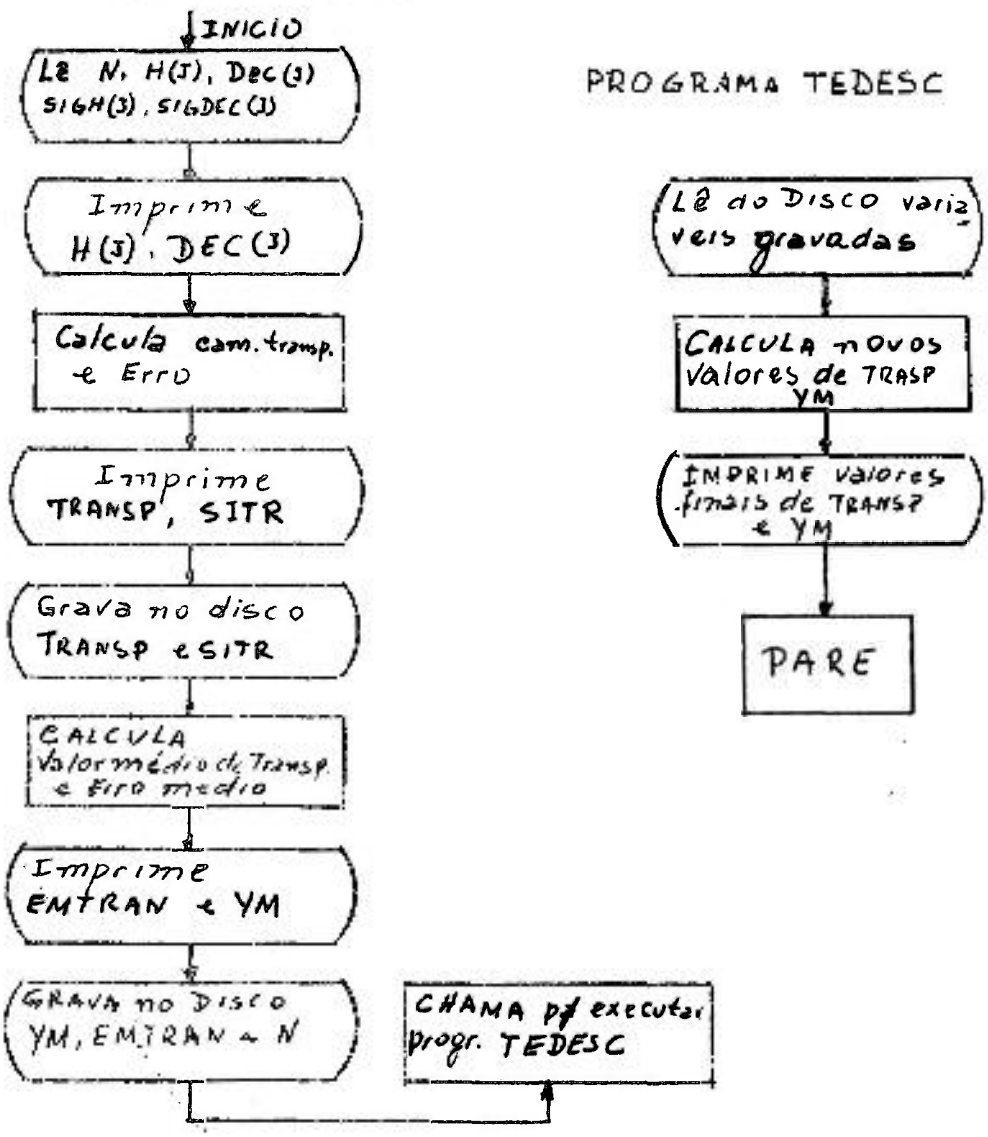
10.1.8. - Desejamos calcular o caminho livre médio de transporte (para neutrons térmicos) de meios moderadores, pelo método de fonte pulsada.

Como é um programa longo e não temos certeza de que caiba na memória do Computador, utilizaremos a facilidade do disco para dividir o programa em duas partes, fazendo a segunda parte como um programa encadeado com o programa principal. Daremos ao programa encadeado o nome de TEDESC, para ser carregado no disco em forma permanente. O método matemático utilizado será a aplicação direta da formula:

$$2.84\left[\left(\frac{H_1 + H_2}{H_1 - H_2}\right)^2 - 1\right] \chi^2 tr - (H_1 + H_2)\lambda_{tr} + \frac{3(\lambda_2 - \lambda_1)}{v\,\pi^2}\,\frac{(H_1 \cdot H_2)^2}{H_1 - H_2} = 0$$

desprezando os têrmos de gráu maior que 2, resultantes do desenvolvimento em série da expressão que traduz em têrmos matemáticos as considerações físicas do problema. Desta equação de 2º gráu utilizada, ùnicamente tem significado físico a raiz positiva; sendo assim, a raiz negativa é desprezada no próprio programa.

Diagrama de Bloco:



Programa FORTRAN (Principal):

```
C       MEAN FREE PATH
C       ALTURAS ENTRAM EM ORDEM DECRESCENTE
C       H(J,K) = ALTURA (CM)
C       DEC(J,K)=CONST. DE DECAIMENTO (1/SEG)
C       VELOCIDADE ENTRA EM CM/SEG
        DEFINE DISK (10,100)
        DIMENSION H(20),DEC(20),SIGH(20),SIGDEC(20),TRANSP(20,20),
     1  SITR(20,220)
     1  READ 19,N
    19  FORMAT(I4)
     2  READ 10,(H(J),DEC(J),J=1,N)
        READ 10,(SIGH(J),SIGDEC(J),J=1,N)
    10  FORMAT (2E14.8)
        TYPE 60
    60  FORMAT (5X,4HH(J), 11X,6HDEC(J))
        PIQ=9.86965
        TYPE 10,(H(J),DEC(J),J=1,N)
        TYPE 411
   411  FORMAT(//)
        VELOC=.2489*10.**6
        W=N
        L=N-1
        Y=0.
        EY=0.
        I=10
        S=0.
        DO 30 J=1,L
        M=J + 1
        DO 30 K=M.N
        A=2.84*(((H(J)+ H(K))**2/(H(J)*H(K)))-1.1
        B=H(J)+H(K)
        C1=(3.*(DEC(K)-DEC(J))*(H(J)*H(K))**2)
```

```
      C=C1/(VELOC*PIQ*(H(J)-H(K)))
      DELTA1=(B**2-4.*A*C)
      IF(DELTA1) 3,3,4
    3 TYPE 100
  100 FORMAT (2X,16HDELTA=ABANDONADO)
      S=S+2.
      TRANSP(J,K)=0.
      GO TO 30
    4 DELTA=DELTA1**0.5
      TRANSP(J,K)=(B-DELTA)/(2.*A)
      ALFA=A
      BETA=B/2.
      GAMA=C
      DEN1=1./(BETA**2-ALFA*GAMA)**0.5
      DALFA=(GAMA*DEN1-2.*TRANSP(J,K))/2.*ALFA
      DBETA=-TRANSP(J,K)*DEN1
      DGAMA=1.*DEN1/2.
      DALFAQ=DALFA**2
      DBETAQ=DBETA**2
      DGAMAQ=DGAMA**2
      SIGH1Q=SIGH(J)**2
      SIGH2Q=SIGH(K)**2
      SIQDE1=SIGDEC(J)**2
      SIQDE2=SIGDEC(K)**2
      R1=(SIGH1Q/H(J)**2+SIGH2Q/H(K)**2)
      R2=(2.84*(H(J)**2-H(K)**2)/(H(J)*H(K)))**2
      SIQALF=R1*R2
      SIQBET=0.25*(SIGH1Q+SIGH2Q)
      R11=(3.*(DEC(K)-DEC(J))/(VELOC*PIQ))**2
      R12=(H(J)*H(K))**4/(H(K)-H(J))**2
      R13=R11*R12
      R17=R13/(DEC(K)-DEC(J))**2
      R18=R17*(SIQDE1+SIQDE2)
```

```
      CA=3.*(DEC(K)-DEC(J))/(VELOC*PIQ)
      R19=CA**2*(H(J)*H(K))**2/(H(J)-H(K))**4
      R14=(H(K)**2)*(2.*H(K)-H(J))**2*SIGH1Q
      R15=(H(J)**2)*(2.*H(J)-H(K))**2*SIGH2Q
      SIQGAM=R19*(R14+R15)+R18
      SIQTR=DALFAQ*SIQALF+DBETAQ*SIQBET+DGAMAQ*SIQGAM
      SITR(J,K)= SIQTR**0.5
      TYPE 3000,J,K,J,K
3000  FORMAT(2X,7HTRANSP(,13,13,1H),10X,5HSITR(,13,13,1H))
      TYPE 3001, TRANSP(J,K),SITR(J,K)
3001  FORMAT(3X,E14.8,10X,E14.8)
      Y=Y+TRANSP(J,K)
      EY=EY+SIQTR
      RECORD (I) TRANSP(J,K),SITR(J,K)
 30   CONTINUE
      TYPE 412
 412  FORMAT(//)
      YM=Y*2./((W*(W-1.))-S)
      EMTRAN=(EY**0.5)*2./((W-1.1)-S)
 328  TYPE 4000
4000  FORMAT(4X,5HMEDIA,10X,4HERRO)
      TYPE 5000,YM,EMTRAN
5000  FORMAT (2E14.8)
      I=1
      RECORD(I) YM,EMTRAN,L,N
      CALL LINK(TEDESC)
      END
```

Programa FORTRAN (Encadeado):

```
C     PROGRAMA LINK TEDESCO
      DEFINE DISK(10,100)
      DIMENSION TRANSP(20,20),SITR(20,20)
      I=1
```

```
      FETCH(I) YM,EMTRAN,L,N
      P=0
      CAL=3.*EMTRAN
      I=10
      EY=0.
      Y=0
      DO 101 J=1,L
      M=J+1
      DO 101 K=M,N
      FIND(I)
      FETCH (I) TRANSP(J,K),SITR(J,K),SITR(J,K)
      PS=YM+CAL
      PI=YM-CAL
      IF(YM-TRANSP(J,K))15,1,3
   15 IF(PS-TRANSP(J,K))2,1,1
    3 IF(PI-TRANSP(J,K))1,1,2
    1 P=P+1.
      Y=Y+TRANSP(J,K)
      EY=EY+SITR(J,K)**2
      GO TO 101
    2 TYPE 1010,J,K
 1010 FORMAT(5HVALOR,1X,10HABANDONADO,3X,7HTRANSP(,13,13,1H))
  101 CONTINUE
      YM=Y/P
      EMTRAN=EY**0.5/P
      TYPE 327
  327 FORMAT(10X,14HVALORES FINAIS)
      TYPE 4000
 4000 FORMAT(4X,5HMEDIA,10X,4HERRO)
      TYPE 5000,YM,EMTRAN
 5000 FORMAT (2E14.8)
      CALL EXIT
      END
```

10.2. - Exercícios a resolver

10.2.1. - Determine qual o tipo de cada uma das quantidades abaixo e verifique se é válida ou inválida. Se inválida, por que?

a) MAX$8
b) QUANT(3 * B)
c) -3.78.
d) IOVR7
e) AII
f) 427
g) A(* M)
h) 71E-8.
i) ABLE(3*N-5)
j) I71
k) RATE(3*I,2-J)
l) *17.2
m) JOBNO(I,3,L)
n) MIN(A)
o) AAA
p) GABLE
q) 5E*9
r) I(-K)
s) 12.E15
t) JACK(-3*K)

10.2.2. - Pode uma variável à esquerda de um sinal de igual num comando aritmético, ser subscrita da?

10.2.3. - Quando uma quantidade inteira pode aparecer numa expressão de ponto flutuante?

10.2.4. - Qual será o valor de cada um dos comandos aritméticos abaixo, onde
A=3.,B=2.,C=1.,I=3,J=2,K=1

a) L=I/J

b) M=J*K

c) N=K-1

d) D=C*A

e) E=B/C

f) F=C-A

g) A=I/J

h) M=A/B

i) I=2*I

j) C=C + 1.

10.2.5. - Quais dos seguintes comandos aritméticos são válidos no FORTRAN II? Por que?

a) A(I,3)=M2 + J

b) X(I + 2,J)=-3.* D + (E-F)

c) Y=I**A

d) A(B)=I+2

e) A+3.=B*C

f) M=(A+B)

g) Z=A**R

h) I(J)=K(J)/J

i) I(A)=H + 14.

j) W=I+(-B)

10.2.6. - Quais dos seguintes comandos FORTRAN II são válidos? Por que?

a) DO 10 FEW = 1,10,4

b) DO 51 K=1,K-1,3

c) GO TO M

d) GO TO (1,2,3),A

e) GO TO 187

f) IF(A-I)1,2,2

g) GO TO 3,4

h) DO 1,J=1,50,1

i) GO TO (4,7,3)K

j) GO TO N-12

10.2.7. - Escrever o comando DIMENSION e o comando necessário para transmitir uma matriz A de 10 x 10, para o computador, e como estariam os dados nos cartões?

10.2.8. - Escrever o comando necessário para transmitir ao computador os seguintes dados:

A(1),A(2),A(3),A(4),A(5),BJOB,NEXT,DELTA(2),E(3),E(5),E(7),E(9).

10.2.9. - Escrever um comando para furar um cartão com o seguinte resultado:

F(2,2),G(1,4),G(2,4),G(3,4)

10.2.10.- Escrever o comando necessário para perfurar um cartão com as palavras:

THE FOLLOWING ARE PAYROLL CARDS

10.2.11. - Quais dos seguintes comandos são válidos?

a) FORMAT(I3,E12.8,5F10.2)

b) PRINT 2,A,B(I),C(I)

c) FORMAT(3I2)

d) TYPE A

e) PRINT 10245,A,I,B,J

f) FORMAT(3F10.4)

g) PRINT 4,A,B,3.2,D

h) FORMAT(2F10.4,I4)

i) FORMAT(2OHGObTObNEXTbJOB,E10.6,I3)

j) FORMAT(I100,F18.9,E14.7)

k) TYPE 35H3,A,B,C

10.2.12. - São dados dois conjuntos A e B, cada um contendo 30 elementos. Programar o cálculo de:

$$D = \left[\sum_{i=1}^{30} (Ai-Bi)^2 \right]^{1/2}$$

Cada uma das quantidades dos conjuntos acima está perfurada em um cartão (60 cartões ao todo), e deseja-se o valor D impresso pela máquina de escrever.

10.2.13. - Um conjunto de uma dimensão chamado X contém 50 elementos. Calcule os 49 elementos de outro conjunto chamado DX de acôrdo com:

$$DX(I) = X(I+1) - X(I) \; ; \quad I = 1,2,3,\ldots,49$$

10.2.14. - Dados dois conjuntos de uma dimensão chamados A e B, cada um contendo sete elementos. Os sete elementos de A estão perfurados num primeiro cartão e os sete de B num segundo cartão. Escreva um programa que leia os cartões acima. Calcule ANORM, definido abaixo, e perfure seu valor em um cartão e também imprima pela máquina de escrever e pela impressora.

$$ANORM = \sqrt{\sum_{i=s}^{7} a_i \; b_i}$$

10.2.15. - Sejam duas matrizes A(10,10) e B(10,10). Fazer um programa para calcular a matriz D, definida por:

$$D = A + B$$

10.2.16. - Escrever um programa para inverter uma matriz n x n. Supôr os valores desta matriz entrando por cartões.

10.2.17. - Calcular: $x = \dfrac{-b \pm \sqrt{b^2-4ac}}{2a}$ para um conjunto de valores a, b, c, de forma F7.3. Cada conjunto de a, b, c, perfurado em um cartão.

10.2.18. - Fazer a programação do parágrafo 10.4. apli-

cando o comando DO.

10.2.19. - Fazer um programa geral para integração de funções, utilizando a regra parabólica (Simpson).

10.2.20. - Escrever um programa para calcular a capacidade calorífica dos sólidos (CCS) definida por

$$CCS = \int_0^{2.5} \frac{x^3}{e^x - 1}\, dx$$

com $\Delta X = 0.25$

10.2.21. - Calcular a expressão complexa:

$$Y = \left(\omega\alpha^2 + \beta\right)^2 + i\left(\omega^2\gamma^3 + \delta\right)$$

para valores de α e β fixos. Atribuir valores a γ e δ. Comparar com $|Y|^3 \geq 500$

ω varia de 0 à 500; $\Delta\omega$ a escolher.

10.2.22. - Calcular a eficiência de um motor DIESEL, definida por:

$$\eta_{ff} = 1 - \frac{1}{R^{\gamma-1}} \left| \frac{\left(\frac{V_3}{V_2}\right)^{\gamma} - 1}{\gamma\left(\frac{V_3}{V_2} - 1\right)} \right|$$

onde:

R = razão de compressão;

γ = razão de calores específicos

$\frac{V_3}{V_2}$ = razão de carga

Dados: $V = \frac{V_3}{V_2} = 0.163$

$R = 8.0 \text{ ——— } 12.0$

$\Delta R = 0.5$

$\gamma = 1.3 \text{ ——— } 1.6$

$\Delta\gamma = 0.05$

Calcular e imprimir E_{ff} para tôdas as combinações possíveis de R e γ .

10.2.23. - Escrever um programa para resolver um sistema de n equações a n incógnitas pelo método iterativo.

10.2.24. - Escrever um programa para resolver sistema de equações algébricas de primeiro gráu simultâneas, de duas a 40 equações e o mesmo número de incógnitas. Satisfazer a condição:

$$|A_{ii}| > \sum_{j \neq i} |A_{ij}|, \quad i = 1, 2, \ldots, N$$

N = número de equações

Usar método de iteração de Seidel. Recomenda-se utilizar Comandos de disco.

CAPÍTULO XI

Procedimento de Programação

11.1. - Lista dos Comandos FORTRAN II-D

A tabela a seguir esquematiza os diversos comandos da linguagem automática FORTRAN II-D e seus respectivos tipos:

Comandos	Aritméticos	E/S	Controle	Especificação	Sub-pro
A=B	x				
ACCEPT n, lista		x			
*ACCEPT TAPE n, lista		x			
CALL Nome $(a_1,a_2,\ldots a_n)$					x
CALL EXIT		x			
CALL LINK (Nome)					x
COMMON A,B,...				x	
CONTINUE			x		
DEFINE DISK(N_1,N_2)				x	
DIMENSION V,V,...V				x	
DO n i=m_1,m_2,m_3			x		
END			x		
EQUIVALENCE(a,b,c,...),(d,e,f,...)				x	
FIND (I)		x			
FORMAT(S_1,S_2,...S_n)				x	
FETCH(I) Lista		x			
FUNCTION Nome(a_1,a_2,a_n)					x
GO TO n			x		
GO TO(n_1,n_2,...n_m),i			x		
IF(a) n_1,n_2,n_3			x		
IF(SENSE SWITCH i)n_1,n_2			x		
PAUSE			x		
PRINT n, lista		x			
*PUNCH TAPE n, lista		x			
PUNCH n, lista		x			
RECORD(I) lista		x			
READ n, lista		x			
RETURN					x
STOP			x		
SUBROUTINE Nome(a_1,a_2,...a_n)					x
TYPE n, lista		x			

* Os comandos ACCEPT TAPE e PUNCH TAPE não foram considerados no texto porque o equipamento do I.E.A. não dispõe de leitora e perfuradora de fita de papel.

11.2. - A programação

Para a realização de um programa para resolver um problema no computador é necessário considerar 5 etapas. Elas não são rígidas, e a ordem a seguir dependerá do próprio problema considerado.

1. - Definição do problema
2. - Seleção do método para resolvê-lo
3. - Análise do problema
4. - Confecção do programa
5. - Revisão do programa

11.2.1. - Definição do problema

Antes que um programa, que resolverá um determinado problema, seja escrito, o problema deve ser claramente definido, analisadas tôdas as possíveis alterações e condições.

Como um programa para computador deve ser explícito em todos os aspectos, e como o computador não pode tomar decisões por si mesmo, o problema deve ser estabelecido com a maior precisão possível, assim como a forma e a natureza dos dados a serem processados, e a forma mais adequada de fornecer êsses dados. Deve-se conhecer a precisão que se deseja obter na saída, e a forma como o programa deve fornecer os resultados.

11.2.2. - Seleção do método

Na seleção do método para resolver o problema, são possíveis geralmente vários caminhos, dependendo principalmente da complexidade do problema. Embora não seja possível dar certas regras rígidas para a seleção do método, aplicáveis a todos os problemas, daremos algumas indicações que, supomos, facilitarão a escolha do caminho mais certo:

a) Evitar que o método gere quantidade extraordinàriamente grande de resultados intermediários. Deve-se levar em conta a capacidade da memória do computador.

b) Os métodos iterativos são convenientes porque economizam comandos e instruções na programação e na execução.

c) Sempre que possível, utilizar a precisão "standard" do Computador; precisões maiores ocupam mais espaço na memória, e a solução é menos rápida.

d) É altamente vantajoso se o método permite segmentação do programa em sub-programas ou programas ligados (Link). Isto possibilita um melhor contrôle dos possíveis êrros.

e) O método deve possibilitar a utilização das sub-rotinas de biblioteca disponíveis com o sistema do computador e com a linguagem utilizada.

Uma consideração, às vêzes paralela à seleção do método, se o centro de cálculo dispõe de mais de um computador, é a escolha do computador onde será processado o programa, tendo em conta a capacidade e velocidade de cálculo de cada computador.

11.2.3. - Análise do problema

Depois de selecionado um método para a solução, é necessário analisar o problema para identificar as operações componentes e as relações entre elas. O processo de análise é muito variável. Pode ir desde uma análise puramente mental, a uma análise cujos passos precisam ser escritos em forma matemática, e, finalmente, ser até necessário esquematizar por meio de um desenho que indicará as partes do problema e a sua seqüência.

O gráu de análise depende tanto do problema quanto do programador. Pode ser que o programador deseje mais detalhes na sua análise, ou que o programa, pela sua complexidade, necessite de muitos passos ou cálculos intermediários.

Em geral um problema requer primeiramente uma análise sem levar em conta o computador, de modo que o resultado da análise

pode ser aplicado a qualquer computador. Numa segunda etapa, já leva em conta o computador, exigindo uma análise mais detalhada. Nesta análise deve-se considerar as facilidades disponíveis no computador e combiná-las com os passos matemáticos do problema.

O desenho de um diagrama bloco torna-se então necessário e útil. O diagrama indicará a sequência das operações a serem realizadas, o caminho e desvios possíveis durante o processamento.

O valor do diagrama bloco está principalmente na sua natureza, simples desenho, permitindo uma visualização de todo o conjunto. Se o diagrama bloco é suficientemente detalhado,os passos da programação tornar-se-ão muito simples, resultando quase que exclusivamente numa tradução à linguagem de programação. Porém, às vêzes, diagramas muito detalhados escurecem a estrutura do problema investigado.

Existe outra razão para o uso do diagrama bloco: Se o programador reexamina um programa depois de um certo tempo, ou outra pessoa deve estudar o mesmo programa, é muito mais fácil analisar o diagrama bloco que os comandos do próprio programa.

Durante a análise do problema pode-se usar tipos de diagramas bloco. Um tipo trata especialmente com a estrutura do problema e constitui o diagrama bloco do problema. Um segundo tipo de diagrama, é uma modificação do primeiro mas é desenhado considerando-se as características do computador e constitui o diagrama bloco do computador.

No capítulo anterior já foram indicados os símbolos a utilizar no diagrama bloco.

11.2.4. - Elaboração do programa.

Quando a análise é completada, passa-se a etapa de escrever o programa.

Utiliza-se o formulário correspondente à linguagem e ao computador selecionados.

Cada comando será escrito seguindo exatamente a forma geral indicada na descrição de cada comando. Quando se escreve o programa em FORTRAN II-D para entrada por cartões, convém escrever levando em conta tôdas as normas de boa programação:

Os números de comandos serão escritos da coluna dois a cinco, sempre, e colocando os dígitos unidade na coluna 5, os dígitos dezena na coluna 4, etc. A coluna 6 será utilizada ùnica - mente quando fôr necessário indicar cartão continuação, e deve--se seguir uma sequência numérica ou alfabética, para facilitar a identificação da ordem dos cartões sucessivos.

Os comandos deverão começar na coluna 7, embora para o computador seja indiferente que tenham começo em qualquer coluna depois da 6.

Cada letra, número ou símbolo especial será colocado numa coluna do formulário. Note-se que os espaços em branco não são considerados pelo computador, de mod. que podem ser evitados,mas seu uso dá mais clareza ao programa. Sempre que possível,deve-se colocar a identificação do cartão nas colunas 73 a 80, combinando letras e números de tal forma que a letra identifique o programa ou partes do programa (por exemplo as duas primeiras letras do nome do programa), e o número, a ordem dos comandos dentro do programa. Isto é tanto mais conveniente quanto mais longo e quanto mais dividido em partes seja o programa.

Para a entrada de dados, utilizar-se-á o formulário correspondente, onde serão escritos os dados de entrada exatamente como estão especificados nos respectivos formatos indicados no programa. Os dados de entrada são escritos a partir da coluna 1 até a 80.

Para uma saída correta, também convém utilizar o mesmo

formulário, onde se esboçarão com símbolos os formatos de saída especificados, para ter uma visão de como o computador fornecerá os resultados.

Quando o programa deva entrar pela máquina de escrever (caso não muito frequente e que deve ser evitado quando possível), não é necessário seguir o mesmo formato que para cartões, mas é conveniente colocar uma marca de registro ao final de cada comando. Se se usar um ou mais registros de comentário, deve-se deixar pelo menos dois espaços entre o C e o que se fôr escrever como comentário.

Recomenda-se utilizar, no formulário de programação, as letras com os símbolos indicados abaixo, para evitar possíveis confusões:

Letras	
O	∅
I	I
Z	Z
S	S

Em anexo, incluem-se os formulários utilizados no SCAD do I.E.A., para a programação FORTRAN II-D.

11.2.5. - Revisão do Programa

Terminada a escrita do programa, vem um estágio muito importante, que é a revisão do programa. Uma boa revisão economizará tempo, evitando sejam processados comandos em que apareçam erros de sintaxe ou erros de lógica. Uma primeira revisão será feita na fôlha de programação. Só depois se poderá passar à fase de perfuração de cartões. Perfurados os cartões, será feita uma revisão para retirar aquêles que possuam erros de perfuração, e proceder à sua duplicação correta antes de passar ao computador.

Antes do processo definitivo, é necessário fazer uma prova do programa com dados simples, mas que mostre a correção do caminho seguido.

No caso de programas longos que necessitem ser divididos em várias partes, cada parte será provada individualmente antes de sua incorporação ao programa total.

CAPÍTULO XII

COMPILAÇÃO

12.1. - Um programa escrito em linguagem automática, neste caso o FORTRAN II-D, não pode ser interpretado pelo computador digital, pois êle só processa [illegible]. Portanto, o programa fonte deve ser transformado em programa objeto, (linguagem de máquina).

Esta "tradução" da linguagem automática para linguagem absoluta da IBM 1620 é feita pelo processador ou Compilador FORTRAN II-D.

Todos os programas são compilados num formato que permite modificar os endereços relativos das instruções (relocatable format), e que chamaremos formato "recolocável". Estas modificações de endereços são feitas antes de começar a execução da instrução. O compilador opera sob o contrôle do programa Supervisor do Monitor I, podendo ser chamado à operação pelo uso de um registro de Contrôle do Monitor (FOR ou FORX). O Sistema Monitor permite as seguintes operações Fortran:

1. - Compilação do programa fonte FORTRAN
2. - Compilação e imediata execução do programa fonte FORTRAN. Do ponto de vista do programador, isto equivale a entrar na máquina o programa fonte como se fôsse o programa objeto.

3. - Os programas objeto podem ser gravados no disco depois de compilados e/ou podem ser perfurados em cartões ou fita de papel.

4. - Execução de programas objeto prèviamente gravados no disco ou perfurados em cartões ou fita.

5. - Execução de programas encadeados. Êste procedimento é utilizado quando o programa total é muito longo e não cabe todo na memória; êle é, então, dividido em partes e cada parte é uma secção do programa total.

12.2. - Processo Geral de Compilação

Embora o processo de compilação de um programa seja uma única operação, existem duas fases através das quais os comandos fonte se convertem em instruções do programa objeto compilado.

O programador seleciona a unidade de entrada do programa fonte por meio dos registros de contrôle do Monitor, e o programa objeto pode sair perfurado em cartões ou fita, ou ser gravado em forma permanente no disco.

Durante a fase I os comandos fonte são analisados e transformados numa série de elementos de instruções de 4 dígitos que são gravados no disco e que serão utilizados durante a fase II para criar as instruções em formato "recolocável".

Se é detetado algum êrro durante a compilação, o computador escreverá pela máquina o código correspondente (tabela 1). Se a chave 1 estiver ligada(ON), fornecendo assim uma lista dos comandos, a máquina de escrever imprimirá imediatamente embaixo do comando correspondente, a mensagem na seguinte forma:

ERROR n

onde, n indica o número código do êrro listado na tabela 1.

Se a chave 1 estiver desligada (OFF), aparecerá a mensagem na seguinte forma:

SSSS + CCCC ERROR n

onde, SSSS é o último número de comando encontrado antes da detecção do êrro, e CCCC é o número de comandos que seguem a SSSS, sendo SSSS + CCCC o comando com êrro.

Exemplo:

0509 + 0012 ERROR 01

significa que o décimo segundo comando depois do comando numerado 509 é incorreto. Se o êrro é detetado antes de haver algum comando numerado, SSSS será 0000. Na contagem de comandos não serão considerados os cartões comentários, cartões em branco nem cartões continuação.

Os erros podem ser do tipo I ou do tipo II. Se durante a fase I da compilação é achado qualquer êrro do tipo I, o processo não continua na fase II, mas se são encontrados erros do tipo II, o processo passa à fase II, porém a execução não será realizada.

12.3. - Registros de contrôle

O Sistema Monitor 1 controla a realização da compilação e a execução do programa-fonte por meio de uns poucos registros de contrôle. Uma vez que o computador esteja sob o contrôle do Sistema Monitor 1, pode ser carregado o compilador FORTRAN-II-D utilizando um dos registros seguintes:

1) FOR êste registro é utilizado quando se deseja ùnicamente a compilação, sem execução do programa fonte.
2) FORX registro utilizado para a compilação e imediata execução do programa fonte.
3) XEQS registro utilizado para a execução imediata de um programa já compilado e que já esteja gravado no disco com um nome determinado ou perfurado em cartões ou fita. Êste registro deve especificar como serão carregadas as sub-rotinas e sub-programas que o programa principal irá utilizar, colocando-se um número na coluna 28. Dêste número código dependerá também

onde começará a carga do programa objeto e sub-rotinas associadas, no nosso caso pode ser a partir da posição 7500 ou 14000.

O Compilador FORTRAN pode utilizar outros registros de contrôle que podem estar em qualquer ordem, mas entre os registros FOR ou FORX e os comandos fonte.

Êstes registros podem ser fàcilmente identificados, pois levam um asterisco na coluna 1. Êles são:

1) FANDK - Êste registro serve para modificar o comprimento das palavras que, normalmente, são processadas com comprimento 10 para ponto flutuante (ff) e 04 para ponto fixo (kk). O formato dêste registro é:

Colunas	1	-	6	*FANDK
	7	-	8	ff
	9	-	10	kk
	11	-	80	não usadas

ff indica o comprimento da mantissa, e pode variar de 02 a 28;
kk indica o comprimento da palavra de ponto fixo, podendo variar de 04 a 10.

2) PSTSN - Êste registro serve para perfurar a tabela de símbolos e os endereços dos números dos comandos.
O formato é o seguinte:

Colunas	1	-	6	*PSTSN
			7	n
	8	-	80	não usadas

onde, n é 2 para perfurar fita de papel e 4 para perfurar cartões.

3) POBJP- Êste registro serve para perfurar o programa objeto uma vez terminada a compilação. O formato é o seguinte:

Colunas	1	-	6	*POBJP
			7	n
	8	-	80	não usadas

n é 2 para fita e 4 para cartões.

4) LDISK - Êste registro serve para mover o programa objeto da área de trabalho do disco, onde foi colocado durante a compilação, a uma área permanente, imediatamente depois de terminar a compilação.

O formato é:

Colunas	1	- 6	*LDISK
	7	- 12	nome (opcional)
	13	- 16	número (opcional)
	17	- 80	não usadas

onde nome é o nome que identificará o programa. Será o mesmo usado no registro XEQS;
número é um número de 4 dígitos, que será a entrada na tabela de identificação DIM. Se êste número não é especificado, êle será criado pelo programa de Utilidade de Disco (DUP), e colocado na tabela DIM. O nome do programa será colocado, também pelo DUP, na tabela de equivalência. Quando se grava um sub-programa FUNCTION ou SUBROUTINE, não é necessário colocar o nome no registro LDISK, porque automàticamente será usado o nome do sub-programa que está no respectivo comando FUNCTION ou SUBROUTINE.

12.4. - Como entrar o programa fonte

O programa fonte pode entrar no computador por meio da leitura de cartões perfurados ou fita perfurada, ou escrevendo os comandos na máquina de escrever do console do computador. O meio de entrada é especificado nos registros de contrôle do Monitor, FOR ou FORX.

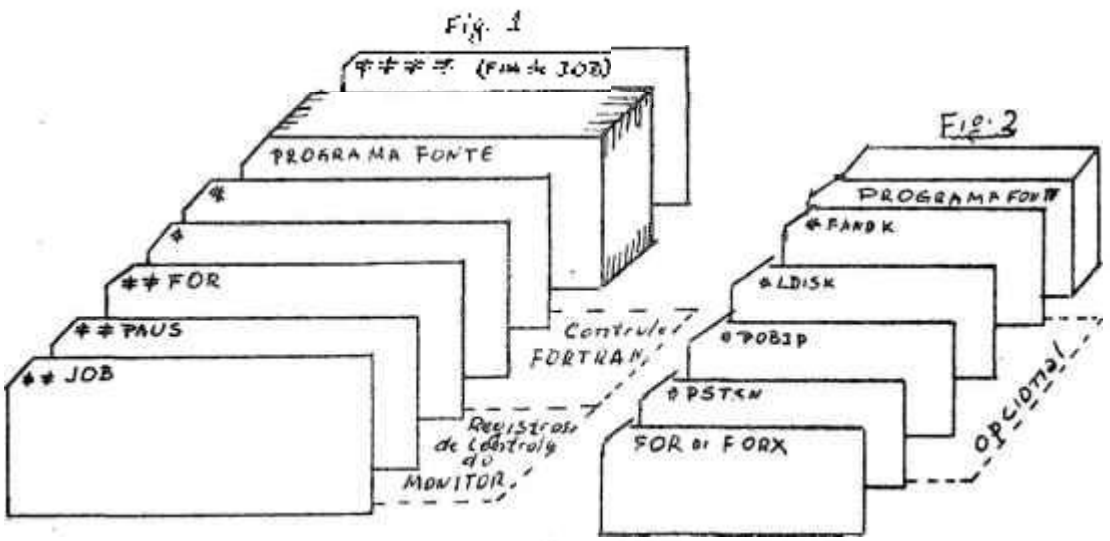
Quando o programa fonte entra por meio de cartões perfurados, os seguintes passos serão dados:

1 - Colocar tôdas as chaves de programa para operação de compilação desejada, conforme especificado na tabela II.

2 - Colocar tôdas as chaves de provas em PROGRAM.
3 - Se é necessário obter cartões ou fitas perfuradas durante a compilação, colocar cartões ou fitas em branco, nos respectivos alimentadores e acionar a chave PUNCH START.
4 - Colocar um registro FOR ou FORX na unidade de entrada.
5 - Colocar qualquer registro de contrôle do FORTRAN II-D desejado na unidade de entrada.
6 - Colocar os comandos fonte na unidade de entrada.
7 - Colocar o registro de contrôle LOCAL segundo o caso na unidade de entrada.
8 - Colocar os dados na unidade de entrada.
9 - Apertar as teclas necessárias para iniciar as operações.

Quando os comandos fonte devem entrar pela máquina de escrever, cada um dos comandos deve terminar com uma marca de registro (≠), e em seguida ser acionada a tecla R/S para processar o comando. Uma vez processado o comando, o carro da máquina de escrever retorna, esperando a entrada de um nôvo comando. Um comando pode ter até 330 caracteres.

As figuras 1 e 2 mostram esquemàticamente o arranjo dos cartões de entrada do programa fonte.



12.5. - Divisão de um programa

Existem várias razões pelas quais convém dividir um programa; uma delas é permitir corrigir os errros que se produzam numa seção de um programa longo, para evitar a recompilação do programa completo. Também é interessante dividir o programa em partes quando diferentes pessoas têm que efetuar a programação, porém isto apresenta certos problemas, como a coordenação dos nomes das variáveis, a impossibilidade de compilação das partes antes de se juntarem as outras partes, etc., uma terceira vantagem é que os sub-programas usados constantemente podem fàcilmente ser incorporados a outros programas.

O programa se divide quase que arbitràriamente, sendo um dêles um programa principal curto e uma série de breves sub-programas SUBROUTINE, usando os mesmos nomes das variáveis em tôdas elas. Os nomes dos sub-programas nos comandos SUBROUTINE e CALL escrevem-se sem argumentos. Depois de haver terminado o programa escreve-se um longo comando COMMON que contém os nomes de tôdas as variáveis usadas e que se inclui tanto no programa principal quanto nos sub-programas.

Com isto fica como se fôsse um único programa.

Outra forma de dividir o programa é fazendo também um curto programa principal e vários outros programas encadeados, cada um com um nome determinado. Gravam-se os valores das variáveis que serão usadas no programa encadeado seguinte, e chama-se à ação êste programa com o comando CALL LINK. O programa chamado, posteriormente chama o outro programa e assim por diante. Cada nôvo programa lê do disco, usando o comando FETCH, as variáveis que utilizará, e que foram gravadas durante a execução do programa anterior, com um comando RECORD. Recomenda-se o uso de áreas comuns para facilitar a identificação das variáveis úteis. Tanto no programa principal quanto nos programas encadeados usar-se-ão os mesmos valores dos parâmetros do comando DEFINE DISK.

12.6. - Como seguir o curso de execução do programa

Sob contrôle das chaves de programação, os comandos do programa fonte podem ser compilados de tal forma a permitir ao operador, quando é requerido pelo programador, seguir o curso da execução do programa, dando na saída o valor da variável do lado esquerdo dos comandos aritméticos, e/ou o valor da expressão calculada num comando IF.

Para isto, durante a compilação deve estar ligada (ON) a chave 2 para os comandos aritméticos, e a chave 3 ligada (ON) para os comandos IF.

Durante a execução, deve estar ligada a chave 4. Isto facilitará a análise do programa especialmente nos casos em que ocorreram erros durante a execução.

12.7. - Programa objeto

12.7.1. - Para executar um programa FORTRAN prèviamente compilado, colocar-se-ão os cartões na seguinte ordem:

1 - Registro de contrôle JOB
2 - Registro de contrôle XEQS
3 - Registro de contrôle LOCAL, quando necessário
4 - Programa principal, se não foi carregado prèviamente no disco
5 - Sub-programas (se necessários, e não prèviamente carregados no disco)
6 - Registro de contrôle DATA. Êste registro deve estar sempre presente se os dados foram carregados prèviamente no disco.
7 - Dados de entrada (se não foram carregados prèviamente no disco)
8 - Registro de contrôle de fim de trabalho (≠ ≠ ≠ ≠)

A figura 3 é um esquema do conjunto de cartões de entrada para a execução do programa prèviamente compilado.

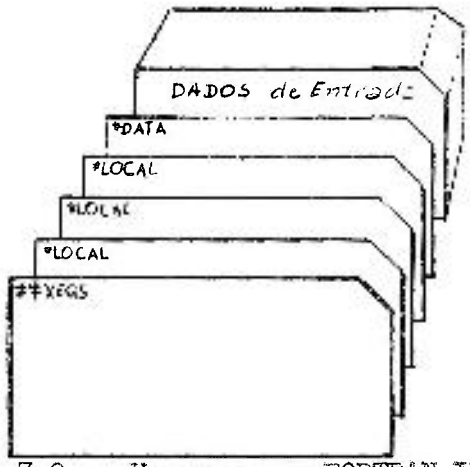


FIG. 3

12.7.2. - Um programa FORTRAN II-D para a IBM 1620, em geral consiste em 3 partes: um programa principal, um grupo de sub-programas, e as sub-rotinas de biblioteca utilizadas pelo programa principal e sub-programas.

O programa principal exerce contrôle sôbre tôdas as operações que estão sendo realizadas. Além da execução normal das instruções, êle tem a possibilidade de chamar sub-programas e sub-rotinas. Os sub-programas poderão ser carregados em continuação ao programa principal, ou poderão estar no início da área de trabalho do disco, e passar à memória de núcleos só quando sejam chamados. Esta segunda forma é definida usando um registro de contrôle LOCAL, no qual pode figurar ou não o nome do programa principal, mas devem figurar os nomes de todos os sub-programas que irão ser carregados de acôrdo com a segunda modalidade mencionada atrás.

O formato do registro de contrôle é:

Colunas	1 - 6	*LOCAL
	7 - 80	Nome do programa principal, nome do sub-programa 1, nome do sub-programa 2, etc.

Durante a execução do programa objeto, o computador fornece mensagens, avisando a ocorrência de qualquer êrro codificado nas tabelas III e IV em qualquer instante.

O computador assume um determinado comportamento quanto à deteção dêsses erros, e na maioria dos caso continua o processamento.

TABELA 1

Erros tipo I durante a compilação

1 - Comando mal redigido.
2 - Êrro de sintaxe em comando não aritmético (exceção: comando DO).
3 - Variável dimensionada usada impròpriamente, i.é. sem subscrito, ou subscrito aparece numa variável não prèviamente dimensionada.
4 - Tabela de símbolos completa (O processamento não pode continuar).
5 - Subscrito incorreto.
6 - Mais de um comando com o mesmo número.
7 - Contrôle de transferência para um comando FORMAT.
8 - Nome de variável com mais de 6 caracteres.
9 - Nome de variável usado como variável não dimensionada e como nome dentro de comando SUBROUTINE ou FUNCTION.
10 - Variável inválida dentro de um comando EQUIVALENCE.
11 - Nome de uma SUBROUTINE ou FUNCTION, ou nome de uma variável muda, usado num comando EQUIVALENCE (só para sub-programas).
12 - k não é igual a f + 2, para equivalência de variáveis de ponto fixo e ponto flutuante.
13 - Dentro de uma lista de EQUIVALENCE, a colocação de duas variáveis prèviamente num COMMON, ou uma variável prèviamente "equivalenciada" e outra prèviamente equivalenciada ou colocada num COMMON.
14 - Falta o número da chave num comando IF(SENSE SWITCH).
15 - Num comando de transferência, faltam ou não estão separados por vírgulas os números de comandos.
16 - Falta de um índice num GO TO COMPUTADO, ou índice inválido, ou não precedido por vírgula.

17 - Número de ponto fixo maior que k dígitos.

18 - Número de ponto flutuante inválido.

19 - Subscrito incorreto dentro de um comando DIMENSION.

20 - O primeiro caráter de um nome não é alfabético.

21 - Variável dentro de um comando DIMENSION prèviamente usada como variável não dimensionada, ou prèviamente dimensionada, ou usada como nome de um sub-programa.

22 - Variável dimensionada usada dentro de um comando de FUNÇÃO ARITMÉTICA.

23 - Mais de 4 cartões continuação.

24 - Número de comando num comando DO apareceu prèviamente em outro comando.

25 - Êrro de sintaxe num comando DO.

26 - Número de FORMAT faltando num comando de Entrada ou Saída.

27 - Número de comando num comando de E/S apareceu prèviamente num comando qualquer, que não o FORMAT, ou um número do FORMAT apareceu prèviamente em outro comando que não de E/S.

28 - Êrro de sintaxe em lista de E/S ou elemento inválido.

29 - Êrro de sintaxe num comando CALL ou argumento inválido.

30 - Num sub-programa, o comando FUNCTION ou SUBROUTINE não é o primeiro.

31 - Êrro de sintaxe ou parâmetro inválido num comando de sub-programa.

32 - Êrro de sintaxe ou variável inválida num comando COMMON.

33 - Variável numa lista COMMON prèviamente colocada num COMMON ou prèviamente "equivalenciada".

34 - Nome de uma função de biblioteca aparece do lado esquerdo de um sinal = ou num COMMON, EQUIVALENCE, DIMENSION, ou comando de E/S, ou o nome da função não é seguido por parênteses.

35 - Êrro de sintaxe no comando FORMAT ou especificação do formato inválida.

36 - Expressão inválida do lado esquerdo de um sinal igual numa expressão aritmética.

37 - Definição de função aritmética não está antes do primeiro comando executável.

38 - Expressão inválida num comando IF ou CALL, ou expressão inválida do lado direito de um sinal igual, em um comando aritmética.

39 - Parênteses incompletos.

40 - Argumento inválido usado ao "chamar" uma FUNÇÃO ARITMÉTICA , ou Sub-programa.

41 - Êrro de sintaxe em comando de E/S em Disco.

42 - Lista de E/S de Disco faltando.

43 - Lista de E/S de Disco contém tanto variáveis simples como nomes de arranjos.

44 - COMMON excede a capacidade da memória (poderá ocorrer quando se definem arranjos muito grandes).

Erro Tipo II durante a compilação

51 - O alcance do DO termina num comando de transferência.

52 - Falta número de comando a um comando seguinte a um de transferência.

53 - Comando não aritmético terminado impròpriamente.

54 - Falta número a um comando CONTINUE

55 - Número de endereços do COMMON excede a capacidade da memória por causa do EQUIVALENCE.

56 - Número de comando maior que 9999.

57 - Comando RETURN aparece num programa que não é um sub-programa (comando ignorado).

58 - Comando RETURN falta num sub-programa.

59 - Número de comando faltando.

60 - Êrro de sintaxe no comando DEFINE DISK.

CAPÍTULO 13

Chaves de instruções

CHAVE	ON	OFF
1	Comandos fonte são impressos pela máquina de escrever à medida que são processados	Comandos fonte não são listados
	Os erros dos comandos fonte são impressos na forma ERROR n	Os erros dos comandos fonte são impressos na forma SSSS + CCCC ERROR n
	No fim da fase I a tabela de símbolos e os números de comandos são impressos	Tabela de símbolos e números de comandos não são impressos
2	Instruções para seguir o curso da execução do programa são compiladas, mas não são geradas instruções adicionais	Instruções para seguir o curso da execução do programa não são compiladas
3	São compiladas instruções para acompanhar o curso do programa (apenas no que se refere às expressões de comandos IF). Uma instrução adicional é gerada no programa objeto para comando IF	Não são compiladas as instruções para acompanhar o curso do programa (p/comandos IF)
4	Erros feitos quando entrando os comandos fonte pela máquina de escrever poderão ser corrigidos por: a) ligando a chave 4 b) utilizando a tecla R/S	c) desligando a chave 4 d) reescrevendo o comando

Observação - Durante a execução do programa objeto, se na compilação estiveram ligadas as chaves 2 e/ou 3, a chave 4 ligada proporcionará a saída dos valores de expressões aritméticas e/ou de expressões de comandos IF, seguindo o curso do programa.

TABELA III

Erros durante a execução

Cod. de erro	Significado e Razão	Resultado
L1	Registro de contrôle LOCAL inválido: palavra LOCAL mal escrita, mal colocada, ou falta o asterisco	Impressão da mensagem JOB ABANDONED; desvio para MONCAL*
L2	Nome inválido no registro LOCAL; não formado de acôrdo com as regras FORTRAN	Impressão da mensagem JOB ABANDONED; desvio para MONCAL*
L3	Nome múltiplo no registro LOCAL; algum nome de sub-programa aparece mais de uma vez para programa, ou nome de algum programa ou programa Link aparece mais de uma vez.	Impressão da mensagem JOB ABANDONED; desvio para MONCAL*
L4	Tabela de sub-programa LOCAL completa	Impressão da mensagem JOB ABANDONED; desvio para MONCAL
L5	Registro encabeçador inválido	Desvio para MONCAL
L6	F e K desiguais. Programa principal e sub-programa não têm o mesmo F e/ou K	O sub-programa não é carregado
L7	Nova sub-rotina chamada por um sub-programa LOCAL; sub-programa LOCAL não pode chamar sub-rotina que não tenha sido utilizada pelo programa principal.	O sub-programa é carregado, e a sub-rotina não é carregada.
L8	Conjunto de sub-rotinas aritméticas e de E/S inválido. Não definido como 1,2,3 ou 4.	É carregado o conjunto standard
L9	Tabela de sub-programas na memória, completa; não é permitido mais de 50 sub-programas	Sub-programa além de 50, ignorados

Cod. de erro	Significado e Razão	Resultado
L10	Novo sub-programa chamado por sub-programa LOCAL; um sub-programa LOCAL não pode chamar outro sub-programa	Sub-programa LOCAL é carregado; o novo sub-programa não é carregado.
L11	A área de disco de sub-programas LOCAL ultrapassa a área de trabalho reservada no disco.	O sub-programa LOCAL não é carregado.

*MONCAL é o nome simbólico da Rotina Analisadora de Registros de Controle do Monitor.

TABELA IV

Códigos de erros de sub-rotinas FORTRAN

Código de erro	Êrro	Resultado guardado
D1	E/S de disco usado sem o comando DEFINE DISK	
D2	Registro lógico especificado excede N2	
D3	Não é encontrada uma marca de grupo ao final de um arranjo que foi lido do disco	
E1	Overflow em FAD ou FSB	$\bar{9}9.....99\bar{9}9$
E2	Underflow em FAD ou FSB	$\bar{0}0.....0\bar{9}\bar{9}$
E3	Overflow em FMP	$\bar{9}9.....\bar{9}9$
E4	Underflow em FMP	$\bar{0}0.....\bar{9}\bar{9}$
E5	Overflow em FDV ou FDVR	$\bar{9}99.....\bar{9}9$
E6	Underflow em FDVR	$\bar{0}00.....\bar{9}\bar{9}$
E7	Divisão por zero em FDV ou FDVR	$\bar{9}99.....\bar{9}9$
E8	Divisão por zero em FXD ou FXDR	$\bar{9}99....$
E9	Overflow em FIX	$\bar{9}99....$
F1	Perda de tôda significância em FSIN ou FCOS	$\bar{0}00.....\bar{9}\bar{9}$
F2	Argumento zero em FLN	$99.....\bar{9}\bar{9}\bar{9}$
F3	Argumento negativo em FLN	ln /x/
F4	Overflow em FEXP	$\bar{9}9.....\bar{9}9$
F5	Underflow em FEXP	$\bar{0}00.....\bar{9}\bar{9}$
F6	Argumento negativo em FAXB Argumento negativo em FSQR	$/A/^{B}$ SQR/X/

Código de erro	Erro	Resultado gerado
F 7	Dado de entrada em forma incorreta ou fora do alcance permitido	
F 8	Dado de saída fora do alcance permitido	
F 9	Registro de E/S maior que 80 ou 87 caracteres (dependendo do meio de E/S)	
G 1	Zero à uma potência negativa em FIXI	$\bar{9}99\ldots$
G 2	Número de ponto fixo à potência negativa em FIXI	$\bar{0}00\ldots$
G 3	Overflow em FIXI	$\bar{9}99\ldots$
G 4	Zero flutuante à potência negativa em FAXI	$\bar{9}99\ldots\ldots\ldots\bar{9}99$
G 5	Overflow em FAXI	$\bar{9}99\ldots\ldots\ldots\bar{9}9$
G 6	Underflow em FAXI	$\bar{0}00\ldots\ldots\ldots\bar{9}9$
G 7	Zero à potência negativa em FAXB	$\bar{9}99\ldots\ldots\ldots\bar{9}9$

Diagrama de Bloco do exercício nº 10.1.7 da página 95.