PHILIPS

P|1000

# FORTRAN B
# Instruction Manual

Data Systems

LUMO-ZET N.V.
Amst. West. Ind. Centrum
VAN HALLSTRAAT 183
AMSTERDAM-W.
Tel. 0.20-16 48 48 (3 lijnen)

LUMO-ZET N.V.
Amst. West. Ind. Centrum
VAN HALLSTRAAT 183
AMSTERDAM-W.
Tel. 0.20-16 48 48 (3 lijnen)

# P|1000

# FORTRAN B
## Instruction Manual

LUMO-ZET N.V.
Amst. West. Ind. Centrum
VAN HALLSTRAAT 183
AMSTERDAM-W.
Tel. 0.20-16 48 48 (3 lijnen)

A publication of
Market Development Department
NV Philips-Electrologica
Computer Systems Division.

Pub. No. 5122 991 07651

July, 1969.

Great care has been taken to ensure that the information contained in this handbook is accurate and complete. Should any errors or omissions be discovered, however, or should any user wish to make a suggestion for improving this handbook, he is invited to send the relevant details to:

NV Philips-Electrologica,
Computer Systems Division,
P.O. Box 245, Apeldoorn,
The Netherlands.

# Preface

This handbook is a training manual. It should be made available to all trainees attending courses on FORTRAN B. It contains a description of the way FORTRAN should be programmed when used for one of the P1000 series systems. Before reading this book the reader should be familiar with the basics of electronic data processing. The user may use the FORTRAN B handbook as a reference manual.
This handbook is intended for programmers.

# Contents

# 1 Introduction

Originally IBM designed a language to formulate mathematical problems for the IBM 704. The system language was called FORTRAN, the language name being derived from the words FORmula TRANslation.
At first only formulae were translated, while, after a time, the language became adapted to the requirements of the programmer.
The two best known FORTRAN systems are FORTRAN II and FORTRAN IV. The last-named system is used for FORTRAN B and is defined according to 'American Standard Association FORTRAN'. FORTRAN was evolved in order to write and test programs quickly in a simple language.
The language includes expressions which link up with the common mathematical scripts. Hence knowledge of the operation codes and other characteristics of the machine has become superfluous.

# 2 General

## 2.1 FORTRAN CHARACTER SET

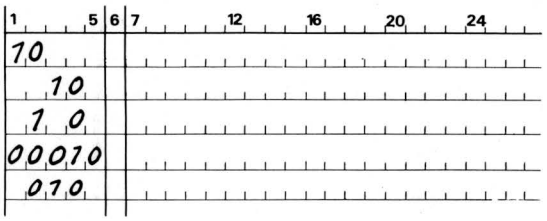
| | | |
|---|---|---|
| Letters: | A B C D E F G H I J K L M N Ø P Q R S T U V W X Y Z | |
| Digits: | 0 1 2 3 4 5 6 7 8 9 | |
| Special signs: | decimal point | . |
| | comma | , |
| | plus | + |
| | minus | – |
| | slash | / |
| | asterisk | * |
| | equals | = |
| | left parenthesis | ( |
| | right parenthesis | ) |
| | apostrophe | ' |
| | currency symbol | $ |
| | blank | |

## 2.2 FORTRAN CODING FORM

The coding form has 80 columns.

Column 1 through 5: statement number.
A statement number consists of a maximum of 5 digits.
Blanks or insignificant zeros are ignored.
Only accepted: blanks and digits 0 to 9 inclusive.
Statement numbers can be in arbitrary sequence.

Example:



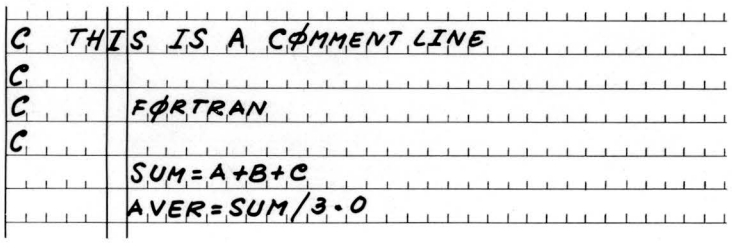

These statement numbers are all equal, namely 10.

Column 7 through 72: FORTRAN statement.

Column 73 through 80: Identification.
In FORTRAN this field is not processed.

In order to keep the program clear and comprehensible we use a comment line. This has no effect on the program.

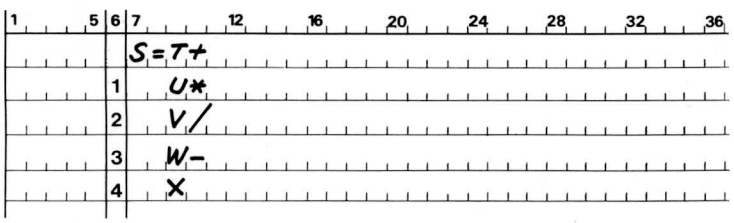
Column 1: A comment line contains the letter C in column 1. Comments may then be written in columns 2 through 80, which, alternatively, may comprise blanks only.
Example:



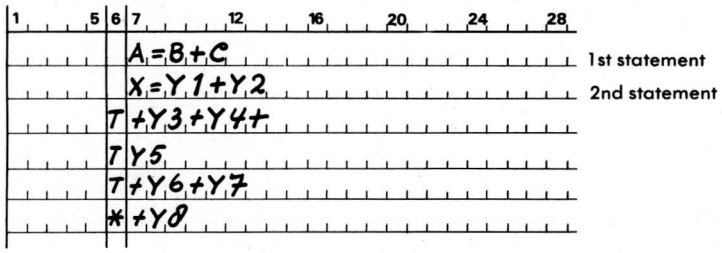

If the statement is too long for one line or for some reason is written on more than one line, we use column 6.

Column 6: line continuation.
The first line of a statement always contains a blank or a 0 in column 6; the following lines of the statement must contain in column 6 a character other than blank or 0.

The formula $s = t + \frac{uv}{w} - x$ may be written on several lines
Example:



The formulae a = b + c and x = y1 + y2 + y3 + y4 + y5 + y6 + y7 + y8 can also be written as follows

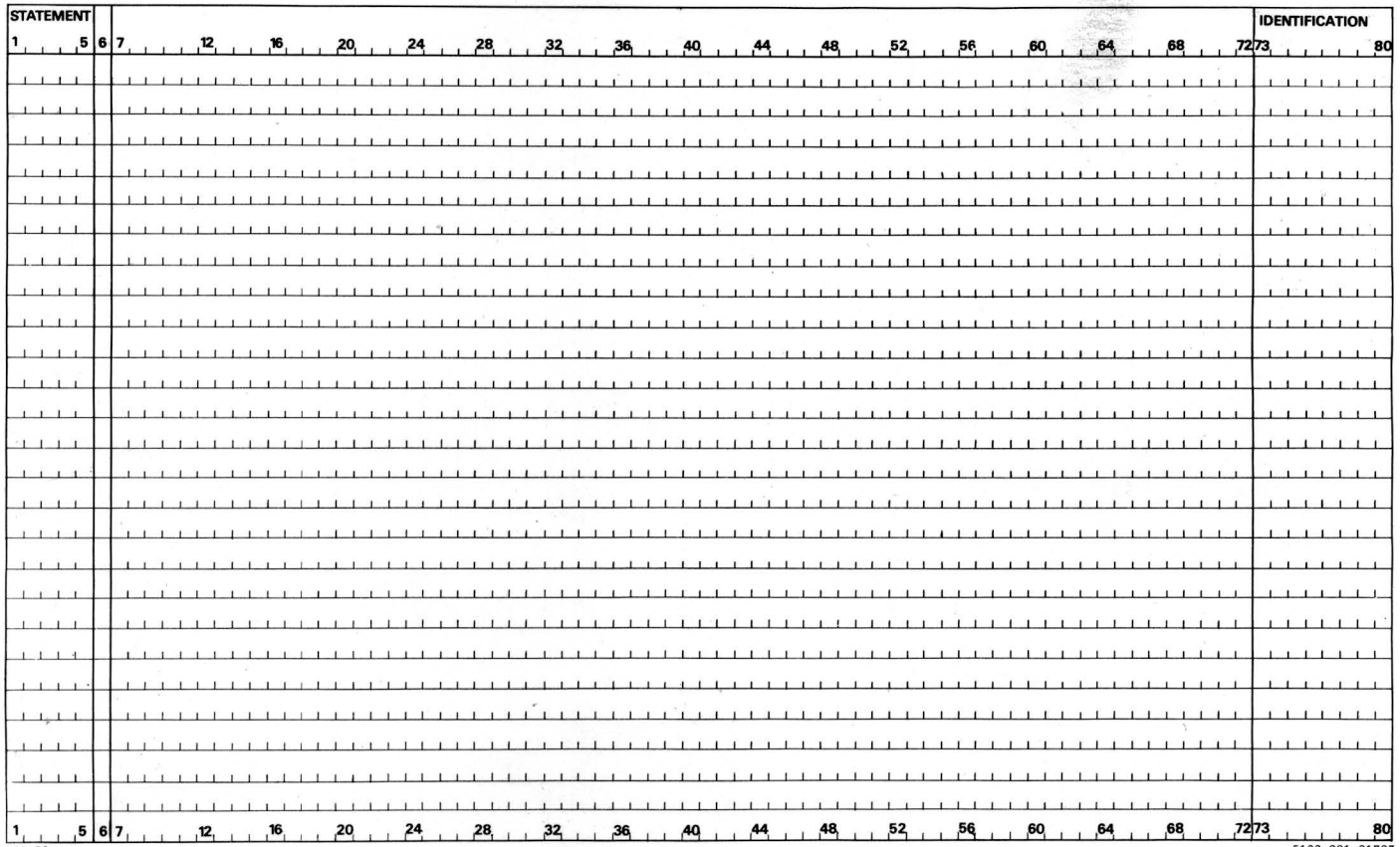


**PHILIPS**

data systems

## FORTRAN

Problem _____ Programmer _____ Date _____ Page _____ of _____




380.00

5122 991 91793

# 3 Constants

## 3.1 INTEGER CONSTANT

This constant is an integer number written without a decimal point.
FORTRAN B limits: $-2^{31} \leqslant$ integer constant $\leqslant 2^{31}-1$,
or $-2147483648 \leqslant$ integer constant $\leqslant 2147483647$.
If no sign is stated the number is positive.
Blanks are ignored.

Examples:

| right | wrong | |
|---|---|---|
| 0 | 0. 0 | contains decimal point |
| 91 | 27. | contains decimal point |
| 264 | 3147483647 | too large |
| 2 64 | | |
| −2127777 | 5, 3 | contains comma |
| +123 | | |

## 3.2 REAL CONSTANT

A basic real constant is a number that is written as an integer part, decimal point, and decimal fraction in that order. The integer and decimal parts are strings, either of which may be empty but not both.
The use of a sign for both the basic real constant and the decimal exponent is mandatory if negative, and optional if positive.
The decimal exponent is written as the letter E, which is followed by an integer constant consisting of a maximum of 2 digits. A real constant is indicated by writing either a basic real constant, a basic real constant followed by a decimal exponent, or an integer constant followed by a decimal exponent.

FORTRAN B limits: $5.4 * 10^{-79} <$ real constant $< 7.2 * 10^{75}$
General form: a E b means $a * 10^{b}$.

Examples:

+0.
−999.9999
0.0
123.4
7.0E+1 (i.e. $7.0 * 10^1 = 70.0$)
12.E+3
12E3
12.0E 3
12.0E+3
12.0E03
12.0E+03
(i.e. $12.0 * 10^3 = 12000.0$)
12.0E−3 (i.e. $12.0 * 10^{-3} = 0.012$)
314E−2 (i.e. $314. * 10^{-2} = 3.14$)

Precision: with real numbers entities to 6 decimal digits can be accurately stated.

### 3.3 DOUBLE-PRECISION CONSTANT

A double-precision exponent is written and interpreted identically to a decimal exponent in a real constant, except that the letter D is used in the exponent instead of E.
A double-precision constant is indicated by writing either a basic real constant or an integer constant, followed by the double-precision exponent.

General form: a D b signifies $a * 10^b$.

Examples:
+145.34D0
+7.9D+3

Precision: entities up to 16 decimal digits.

### 3.4 COMPLEX CONSTANT

This comprises an ordered pair of real constants separated by a comma and included in parentheses.

Example: (135.E2, 12.0) signifies 13500. + 12.0i in which $i^2 = -1$.

### 3.5 LOGICAL CONSTANT

The logical constants true and false are written as .TRUE. and .FALSE. respectively.

### 3.6 HOLLERITH CONSTANT

| | |
|---|---|
| General form: | nHt<br>where t is a string of n characters in which blanks are counted. |
| Examples: | 4HNAME<br>5HbbJ3b (b signifies blank here) |
| FORTRAN B restriction: | $1 \leqslant n \leqslant 249$ |

# 4 Variables

## 4.1 DEFINITION OF VARIABLE

A variable is an entity which attains a value during execution. It is represented by a symbolic name consisting of a maximum of 6 (in FORTRAN B: 8) alphanumeric characters of which the first must be an alphabetic character; special characters are forbidden.

Examples:

| right | wrong | |
|---|---|---|
| ABCD | 2AB | must begin with letter |
| INTERVAL | MAX$ | no special character |
| X1234567 | IDENTIFIER | more than 8 characters |

In FORTRAN there are two basic types of variable, namely integer and real, which may be used in accordance with the above definition.

## 4.2 INTEGER VARIABLE

This must begin with one of the letters:
I, J, K, L, M, N (INteger, i.e. I and N indicate the range of letters).

Examples: I
MAX

## 4.3 REAL VARIABLE

If the name of a variable begins with a letter other than those named above for the integer variable then it is a real variable.

Examples: X
ARTICLE

## 4.4 TYPE STATEMENT

If we deviate from these definitions (refer to 4.1) we use a type statement.

General form: t $v_1$, $v_2$, $v_3$, ....., $v_n$
where t is INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL; and where v is a variable name, an array name, a function name, or array declarative (see also 6.2.1).

Examples:

```
INTEGER SAX, ARTICLE
REAL I,KILØ,LØT,J8
```

# 5 Expressions

Three types of expression can be distinguished, as detailed below:
- arithmetic expressions;
- relational expressions;
- logical expressions.

## 5.1 ARITHMETIC EXPRESSION

This comprises a number of constants and variables, one separated from the other by arithmetic operators which may be combined by parentheses to partial expressions.

Arithmetic operators are:

| | |
|---|---|
| + | addition |
| − | subtraction |
| * | multiplication |
| / | division |
| * * | exponentiation |

The hierarchy is:

| | |
|---|---|
| * * | highest priority |
| * and / | |
| + and − | lowest priority |

Given the same hierarchy processing is from left to right.
A * * B * * C is not defined; this must be either (A * * B) * * C or A * * (B * * C) depending on what is meant.
Two operators must not follow in direct succession, e.g. A * −B is wrong.
To deviate from the normal hierarchy parentheses must be used.

Examples:

| | |
|---|---|
| X + Y ** 3 | means $x+y^3$ |
| (X + Y) ** 3 | means $(x+y)^3$ |
| A + B / C + D | means $a+\frac{b}{c}+d$ |
| A + B / (C + D) | means $a+\frac{b}{c+d}$ |
| (A + B) / (C + D) | means $\frac{a+b}{c+d}$ |

Mixed expressions are arithmetic expressions in which the constants and variables are not of the same type, e.g. A + I in which A is real and I is an integer.
The tables below indicate the mixed expressions which are acceptable. Constants used in the tables are abbreviated as follows:

- I is integer;
- R is real;
- DP is double precision;
- C is complex;
- — is not permissible.

Table for + − * /

| | I | R | DP | C |
|---|---|---|---|---|
| I | I | R | DP | C |
| R | R | R | DP | C |
| DP | DP | DP | DP | C |
| C | C | C | C | C |

} Type of the result

N.B. In the division of integers the result is truncated.

Examples:

```
5 / 3 = 1
2 * 5 / 2 = 5
2 * (5 / 2) = 4
5 / 2 * 2 = 4
```

Table for * *

| base \ exponent | I | R | DP | C |
|---|---|---|---|---|
| I | I | R | DP | – |
| R | R | R | DP | – |
| DP | DP | DP | DP | – |
| C | C | – | – | – |

} Type of the result

## 5.2 RELATIONAL EXPRESSION

This is an expression which contains two arithmetic expressions separated by a relational operator.
The expression can have the value .TRUE. or .FALSE.

The arithmetic expressions must be one of the following:
- both integer
- both real
- both double precision
- the one real and the other double precision.

The relational operators are:
- .LT. means $<$ (less than)
- .LE. means $\leqslant$ (less than or equal to)
- .EQ. means $=$ (equal to)
- .NE. means $\neq$ (not equal to)
- .GT. means $>$ (greater than)
- .GE. means $\geqslant$ (greater than or equal to)

Examples:

```
a  (I+3)/5.EQ.10
b  R .NE. 0.4
c  D .GE. DF
```

In a relational expression more than one relational operator must not occur.

## 5.3 LOGICAL EXPRESSION

This is a combination of relational expressions, logical constants and logical variables separated by logical operators.
A logical expression has the value .TRUE. or .FALSE.
Logical operators: .NOT.
.AND.
.OR.

.NOT. is followed by a relational expression or a logical expression or a logical constant.
.AND. and .OR. are preceded and followed by a relational expression or a logical expression or a logical constant.
.AND. and .OR. may be followed by .NOT. expression.

Examples:

a .NOT. X .GE. 6.4
b (A .AND. B) .OR. .NOT. (Z .LE. T)

Truth tables for .AND. and .OR.

| .AND. | true | false |
|---|---|---|
| true | true | false |
| false | false | false |

Type of the result

| .OR. | true | false |
|---|---|---|
| true | true | true |
| false | true | false |

Type of the result

Hierarchy: .NOT. highest priority
.AND.
.OR. lowest priority

## 5.4 HIERARCHY

The hierarchy is as follows:
** highest priority
* /
+ –
relational operators
.NOT.
.AND.
.OR. lowest priority

# 6 Array

## 6.1 DEFINITION OF ARRAY

An array is a number of entities identified by one symbolic name. In order to be able to specify an array, one, two or three subscripts are added to the variables. These subscripts must be of the integer type only.
The minimum value which a subscript may assume is +1.
A subscript can be written in one of the following ways:

c * v + k
c * v − k
c * v
v + k
v − k
v
k

c and k are integer constants,
v is an integer variable.

We distinguish between 1-dimensional array, e.g. A(1); A(J)
2-dimensional array, e.g. A(4,7); A(I,J)
3-dimensional array, e.g. A(2,3,4); S(K,L,M)

Examples of subscripted variables:

```
A(I,J)
B(I+2,J+3,2*K+1)
Q(14)
P(KLM,JLM+5)
S(J-6)
XYZ(1,2,3)
Z(J,3,5*K)
```

Examples of subscripted expressions:

```
(A-B(I,J+5))
9*C(J)+4.7/(Z(I+2,3*K)*P(K,L,M+3))
-C+D(I,J)*13.67
```

## 6.2 DECLARATION OF ARRAY

The declaration of an array must precede the statement in which this array occurs for the first time.
An array must be declared with the aid of either a type, a DIMENSION, or a COMMON statement.

### 6.2.1 Type statement

The general form is defined in section 4.4.

Example:

```
REAL MATRIX(20,20)
```

### 6.2.2 DIMENSION statement

General form: DIMENSION $v_1$ ($i_1$), $v_2$ ($i_2$) ---- $v_n$ ($i_n$). v is the symbolic name of the array. i is the array subscript and represents the total number of array elements; it may comprise one, two or three positive non-zero integer constants, or dummy variables (separated by commas), each of which represents the maximum value of the particular dimension.

Example:

```
DIMENSION VECTOR (15), MATRIX (20,20)
```

*Note:* VECTOR is a real and MATRIX an integer array.

### 6.2.3 COMMON statement

This statement is dealt with in topic 11.1.3 later in this manual.

# 7 Standard functions

In the FORTRAN compiler and in the library there are a number of standard functions, which are immediately available for the programmer's use.
The names reserved for these must not be used for other entities. There are two types of these functions, namely: (a) intrinsic, and (b) basic external.
The two types are to be found in Appendices A and B. The type of function is indicated by the first letter of the function name, or by a type statement.

Examples:

```
ABS(X)
MAXO(ABCD,XYZ,13)
SIN(A)
SQRT(B**2-4.*A*C)
```

*Note:* arguments may be expressions.

# 8 Assignment and control statements

## 8.1 ASSIGNMENT STATEMENTS

### 8.1.1 Arithmetic assignment statement

General form: v = e
where v is a variable or indexed variable and e is an arithmetic expression.

Rules for the assignment of e to v are detailed below. The abbreviations used in stating these rules are:

I is integer;
R is real;
DP is double precision;
C is complex;

| | |
|---|---|
| I = I | normal assignment; |
| I = R | the integer part of R is assigned to I (leaving out the decimal part); |
| I = DP | the item part of DP is assigned to I (leaving out the decimal part); |
| I = C | the integer part of the real part of C is assigned to I; |
| R = I | conversion of integer to real; |
| R = R | normal assignment; |
| R = DP | DP rounded off to real and assigned to R; |
| R = C | the real part of C is assigned to R; |
| DP = I | conversion of integer to double precision; |
| DP = R | R is extended to double precision and assigned to DP; |
| DP = DP | normal assignment; |
| DP = C | the real part of C is extended to double precision and assigned to DP; |
| C = I | integer is converted to real and assigned to the real part of C, the imaginary part becomes zero; |
| C = R | R is assigned to the real part of C, the imaginary part becomes zero; |
| C = DP | DP is rounded off to real and assigned to the real part of C, the imaginary part becomes zero; |
| C = C | normal assignment. |

Examples:

```
DISCR=B**2-4.*A*C
A=I*J
I=5/2
R=5/2
I=5./2
A=(1.6,3.2)
```

The real value of the expression is assigned to the real variable DISCR.

Integer product is converted to real and assigned to A.

I becomes 2

R becomes 2.

I becomes 2

A becomes 1.6

*Note:* A = (X, Y) is forbidden because X and Y are **variables** (see definition of complex constants).

### 8.1.2 Logical assignment statement

General form: v = e
where v is a logical variable or a logical array element and e is a logical or a relational expression.

Example:

```
G=.TRUE.
H(K)=.NOT.G
G(1,2)=3 .GT.I
```

If 3 is greater than I, G(1, 2) attains the value .TRUE., otherwise .FALSE.

### 8.1.3 GOTO assignment statement

General form: ASSIGN i TO m
where i is a statement number and m is a non-subscripted integer variable.

The statement assigns to m the value i (see 8.2.1.3 ).

## 8.2 CONTROL STATEMENTS

### 8.2.1 GOTO statements

#### 8.2.1.1 *Unconditional GOTO statement*

General form: GOTO k
where k is a statement number.

After executing the GOTO statement the program is continued with the statement that has k as statement number.

Example:

```
      GO TO 25
10    A=B+C
25    C=E**2
```

After execution of statement 50 the program is continued with statement 25.

**8.2.1.2** *Computed GOTO statement*

General form: GOTO $(k_1, k_2, k_3, \ldots, k_n)$, i
where $k_1, k_2, \ldots, k_n$ are statement numbers;
and where i is an integer variable, while $1 \leqslant i \leqslant n$.

After the execution of the GOTO statement the program is continued with the statement having $k_i$ as statement number.

Example:

```
      GO TO (25,10,50,7),ITEM

50    A=B+C

7     C=E**2+A

10    L=C

25    B=27.3E02
```

If ITEM has the value one the program goes on with statement 25; if ITEM has the value two the program is continued with statement 10; etc.

**8.2.1.3** *Assigned GOTO statement*

General form: GOTO m $(k_1, k_2, \ldots\ldots, k_n)$

Considering the GOTO assignment statement (topic 8.1.3 above) together with the foregoing assigned GOTO statement we could have:
ASSIGN i TO m
–
–
GOTO m, $(k_1, k_2, \ldots\ldots, k_n)$

In this case the value i must be one of the following statement numbers: $k_1, k_2, \ldots\ldots, k_n$. After the execution of the GOTO statement the program is continued with the statement identified by the statement label having the value i assigned to m.

Example 1:

```
      ASSIGN 50 TØ NUMBER
10    GØ TØ NUMBER ,(35,50,20,72)

50    A=B+C
```

After statement 10 statement 50 is executed.

Example 2:

```
      ASSIGN 10 TØ ITEM

13    GØ TØ ITEM,(8,12,25,50,70)

8     A=B+C

10    B=C+D
      ASSIGN 25 TØ ITEM
      GØ TØ 13

25    C=E**2
```

In the example shown at this side, statement 10 is executed after statement 13; then 25 is assigned to ITEM and there is a jump to statement 13; then statement 25 is executed.

### 8.2.2 IF statement

#### 8.2.2.1 *Arithmetic IF statement*

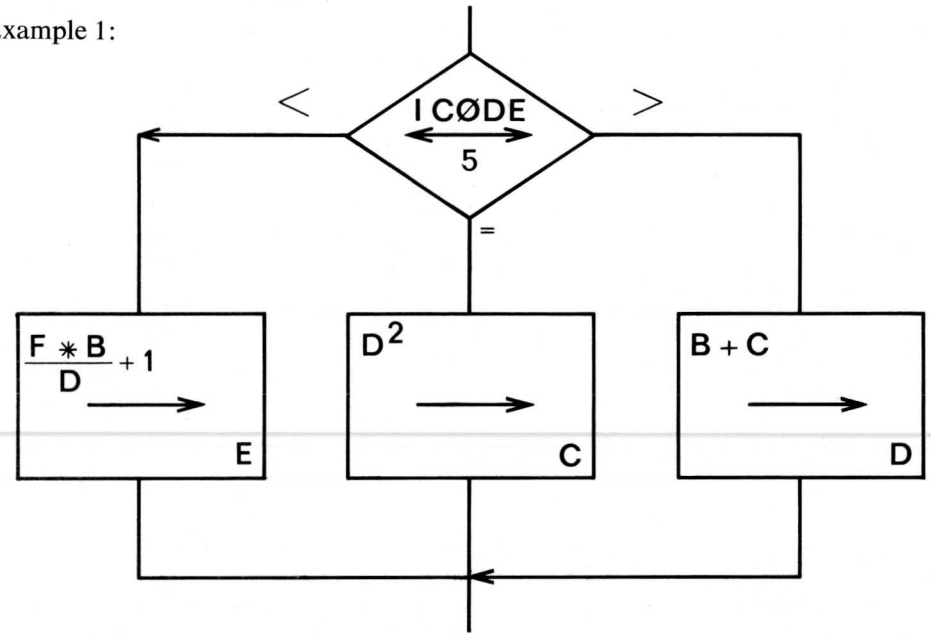
General form: IF (e) $k_1, k_2, k_3$

where e is an arithmetic expression (integer, real or double precision);

and where $k_1, k_2, k_3$ are statement numbers.

The expression e is worked out.
If the value of e is less than zero, the program is continued with statement $k_1$.
If the value of e is equal to zero, the program is continued with statement $k_2$.
If the value of e is greater than zero, the program is continued with statement $k_3$.

Example 1:




```
      IF (ICØDE-5) 70,71,72
70    E=F*B/D+7.
      GØ TØ 700
72    D=B+C
      GØ TØ 700
71    C=D**2
700
```

Example 2:

```
C     ADDS A(1) THRØUGH A(50)
      SUM=0.0
      I=1
700   SUM=SUM+A(I)
      I=I+1
      IF(I-50) 700,700,200
200
```

**8.2.2.2** *Logical IF statement*

General form: IF ( e ) s
where e is a logical or relational expression;
and where s is an (executable) statement, but not a DO statement (see further) or a new logical IF statement.

The expression e is worked out.
If the value of e is true, the statement s is executed, if the value of e is false, the program is continued with the next statement in the program text.

*Note:* Note the difference from the arithmetical IF, in which after 'IF (e)' three statement numbers have to be given, after one of which a jump has to be made, subject to the state of e.
In the case of the logical IF, the statement which is either to be executed or not must be filled in immediately after 'IF (e)'.

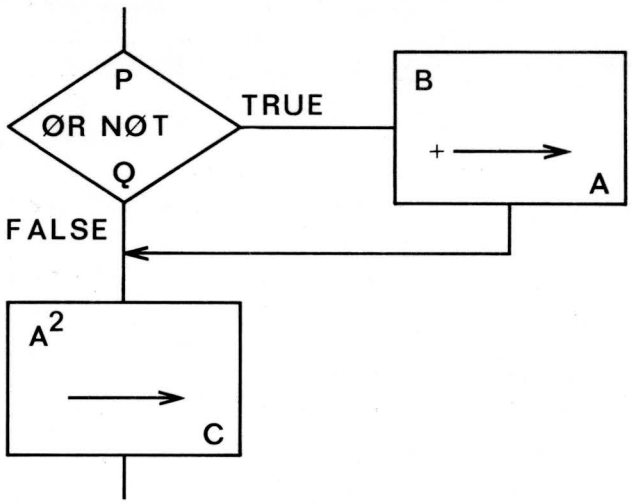
Example:

```
    5  IF (A.LE.0.0) GØ TØ 25
   10  C=D*E
   15  IF (A.EQ.B) ANSW=2.0*A/C
    9  F=G/H

   25  V=X+Y*Z
```

If the expression in statement 5 has the value true, a jump is made to statement 25 (GOTO 25); if the value is false the program is continued with statement 10.
If the expression in statement 15 has the value true, the value of 2.0 * A/C is assigned to ANSW and then the program is continued with statement 9; if the value is false, statement 9 immediately follows on.

Example: P and Q are logical variables




```
    5  IF (P.ØR..NØT.Q) A=A+B
   10  C=A**2
```

If the expression in statement 5 is true, A is increased by B and statement 10 is executed.
If the expression is false, statement 10 is executed.

### 8.2.3 DO statement

General form: DO n i=$m_1$, $m_2$, $m_3$

where:
n is a statement number;
i is an integer variable;
$m_1$, $m_2$, $m_3$ are integer constants or integer variables;
$m_1$ is the initial parameter;
$m_2$ is the terminal parameter;
$m_3$ is the incrementation parameter.

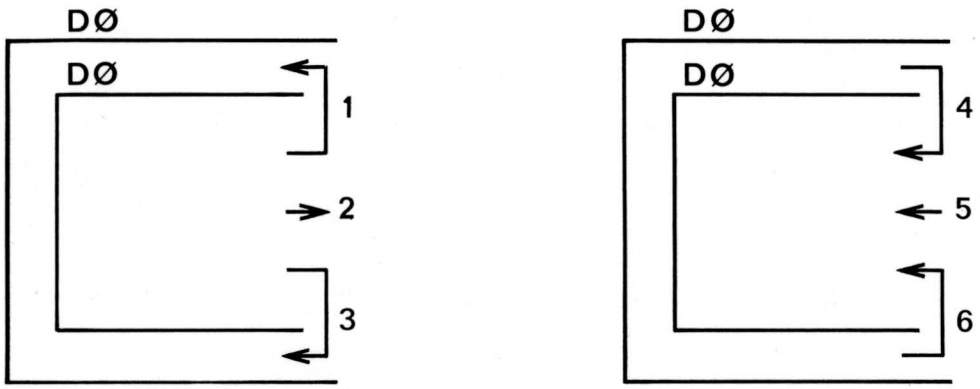
The statements up to and including the statement with statement number n are executed with the value i=$m_1$; then the value of i is increased by the value of $m_3$; so long as the value of i is less than or equal to $m_2$ the statements following the DO statement up to and including the statement with the label n are executed, after which the value of i is again increased by the value of $m_3$.
As soon as the value of i is greater than $m_2$ the program is continued with the statement following the statement with label n.

Example:

```
C     ADDS    A(1) THRØUGH A(50)
      SUM=0.0
      DØ 10 I=1,50
   10 SUM=SUM+A(I)
```

*Note:*

1. Within the DO loop no changes must be made to the values of i, $m_1$, $m_2$ and $m_3$.
2. If $m_1$, $m_2$ and $m_3$ are variables these values remain fixed before a start is made to the execution of the DO statement.
3. It is always permitted to jump **out** of a DO loop, but it is not permitted to jump **into** a DO loop, that is: without executing the DO statement.
4. The values of $m_1$, $m_2$ and $m_3$ must be greater than zero.
5. If $m_3$=1, $m_3$ may be left out of the specification of the DO statement.
6. In a DO statement the statement with label n may not be one of the following statements:
   a. GOTO statement
   b. arithmetic IF statement
   c. DO statement
   d. a RETURN, STOP or PAUSE statement
      (these are still to be dealt with)
   e. a logical IF statement, which contains one of the statements named under a, b, c, d.




Jumps 1, 2, 3 are permissible; 4, 5, 6 are not permissible.

Example: Given 2-dimensional array A containing 15 rows and 15 columns;
1-dimensional array X containing 15 elements.
Calculate the 15 elements of 1-dimensional array B according to the formula

$$B_i = \sum_{j=1}^{15} A_{ij} X_j \qquad i = 1, 2, 3, \ldots\ldots, 15$$

```
      DØ 60 I=1,15
      B(I)=0.0
      DØ 60 J=1,15
   60 B(I)=B(I)+A(I,J)*X(J)
```

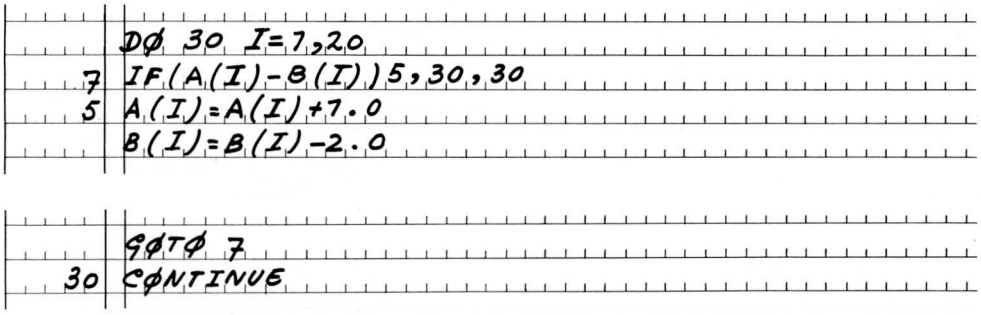
### 8.2.4 CONTINUE statement

In those cases in which one would actually want to use one of the named statements as a last statement in the DO loop, then use is made of the CONTINUE statement.

General form: CONTINUE

The CONTINUE statement is what is known as a dummy statement.

Example: Two rows of numbers A(1) up to and incl. A(20) and
B(1) up to and incl. B(20)
If A(i) < B(i), A(i) must be increased by 1 and
B(i) reduced by 2 and again compared until A(i) $\geqslant$ B(i).




### 8.2.5 STOP statement

General form: STOP
or: STOP n
where n is an octal number consisting of a maximum of 5 digits. (An octal number is formed by any digit in the range 0 through 7.)

The machine terminates the execution of the program. When using STOP n, n indicates which STOP statement ends the execution of the program.

### 8.2.6 PAUSE statement

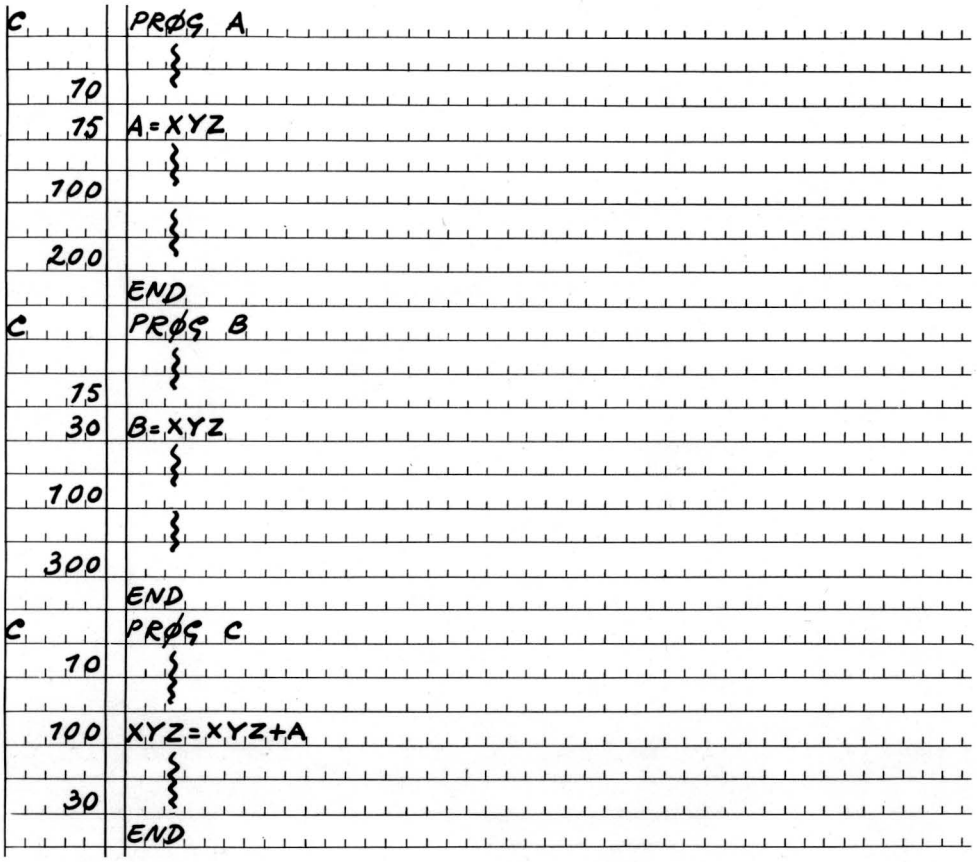
General form: PAUSE
or: PAUSE n
where n is an octal number consisting of a maximum of 5 digits.

The machine halts the execution of the program. In contradistinction to the STOP-statement the program can now be started again, after which the next subsequent instruction is executed.
When using PAUSE n, see note on STOP n.

### 8.2.7 END

Every program unit must end with an END line.
The letters E, N and D must stand in that sequence within columns 7 through 72.
In column 1 through 5 there may be a statement number, but this is ignored.
An END line brings to the notice of the compiler that the end of the part of the program has been reached.
In every program part terminated by an END the statement numbers and the symbolic names are local.
Every program part terminated with END is compiled separately.

Example: prog. A, prog. B and prog. C are compiled separately.




All statement numbers differ.

All variables differ.

# 9 Input/output and FORMAT statements (Part I)

In this chapter we briefly survey the simplest forms of INPUT/OUTPUT statements and FORMAT statements.
In chapter 12 we shall go into this in greater detail.

## 9.1 READ STATEMENT

General form: READ (u, f) k
where:
u is the unit number;
f is the statement number of a FORMAT statement;
k is a list of names of variables in the sequence in which the read-in values must be assigned.

In FORTRAN B the unit number for the card reader is 1. This instruction enables data to be read in. The data is assigned to the appropriate variables.

Example:

```
      READ (1,700) A,B,C
```

Three real numbers are read by the card reader and assigned to A, B and C.

*Note:* The types of the numbers to be read in must be in agreement with those of the corresponding variables.

## 9.2 FORMAT STATEMENT

Conversion and editing information between the internal representation and external character strings are given in the FORMAT statement.
The statement is not bound to a given place in the program.
We distinguish provisionally between two types: integer and real.

Example: A card contains the following data

col. 1

↓

−643016E−0423+12.98014

In the program we have:

```
      READ (7,50) A,K,B
50    FORMAT (E11.5,I2,F9.0)
```

The first field specification, E11.5, belongs to the first field on the card; the second field specification, I2, to the second field and F9.0 to the third field.

The first field specification indicates 11 columns ('11' in E11.5).
The number is interpreted really and has an exponent (E-04).
The '5' in E11.5 signifies that the number must be considered as if there were 5 digits after the (assumed) decimal point.
The number occurring in A is thus $-6.43016E-04 = -6.43016 \times 10^{-4} = -0.000643016$.
The second field specification is simpler.
'I' signifies an integer and '2' signifies two columns.
I2 is thus an integer consisting of two columns (without a sign in this example).

*Note:* The integer −23 must be specified by I3.

The third field consists of nine columns, indicated by '9' in F9.0.
The number is interpreted as real but without an exponent.
The '0' signifies no digits after the decimal point.
This last applies only if no decimal point has been punched.
In this example a decimal point has indeed been punched and is retained, thus in B we have +12.98014.
If +1298014 were punched with field specification F8.3, +1298.014 would have been assigned to the appropriate variable.

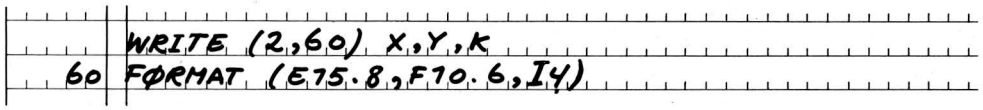
## 9.3 WRITE STATEMENT

General form: WRITE (u, f) k

For u, f, and k, see details under 9.1 (READ statement).

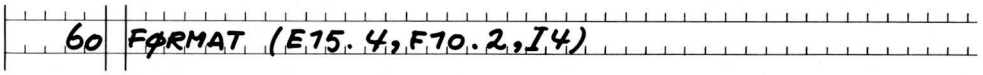
In FORTRAN B the unit number for the card punch is 2 and for the line printer is 3.
With these instructions data can be implemented.

Example:



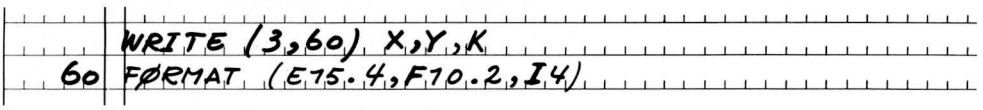

*Note:*
Plus signs are not punched or printed.
X contains −410.46128
Y contains +12.614078
K contains +649
The following is punched:
col. 1
↓
−0.41046128Eb03b12.614078b649 b is blank

If the format were written as




the following would be punched:
col. 1
↓
bbbb−0.4105Eb03bbbbb12.61b649 b is blank

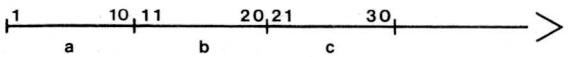
If the result has to be printed with the line printer we write the following:




## 9.4 EXAMPLE

Calculate the square roots of the quadratic equation $ax^2+bx+c = 0$, whose coefficients are read in with the card reader.
The square roots X1 and X2 must be printed on the line printer.

Field division of card.




```
STATEMENT
1    5 6 7   12   16   20   24   28   32   36   40   44   48
C      CALCULATIØN ØF THE SQUARE RØØTS
C      ØF A QUADRATIC EQUATIØN
C
       READ (1,20) A,B,C
       IF (A) 11,12,11
11     D=B**2-4.*A*C
       IF (D) 21,22,23
23     D=SQRT(D)
22     X1=(-B+D)/(2.*A)
       X2=(-B-D)/(2.*A)
       WRITE (3,30) X1,X2
15       }
         }
12     X1=-C/B
       WRITE (3,30) X1
       STØP
21     STØP 1
C      STØP 1  DISCR. NEGATIVE
20     FØRMAT (F10.0,F10.0,F10.0)
30     FØRMAT (E20.8,E20.8)
```

*Note:* Field specifications may be preceded by a repetition factor. In the example the last two statements should be written as:

```
20     FØRMAT (3F10.0)
30     FØRMAT (2E20.8)
```

# 10 Functions

## 10.1 STATEMENT FUNCTION

A statement function is defined by an arithmetic or logical statement. The statement function must be defined at the beginning of the program part (but after the specification statements, see 11.1).

General form: $f(a_1,a_2,\ldots\ldots,a_n) = e$
in which:
f is a function name;
e is an expression;
$a_1,a_2,\ldots\ldots, a_n$ (formal parameters) are names of (unsubscripted) variables.

Example:

```
      FORM(A,B)=3.*A+B**2+X+Y+Z
        }
   10 D=FORM(S,T)
```

Statement with statement number 10 is equivalent to

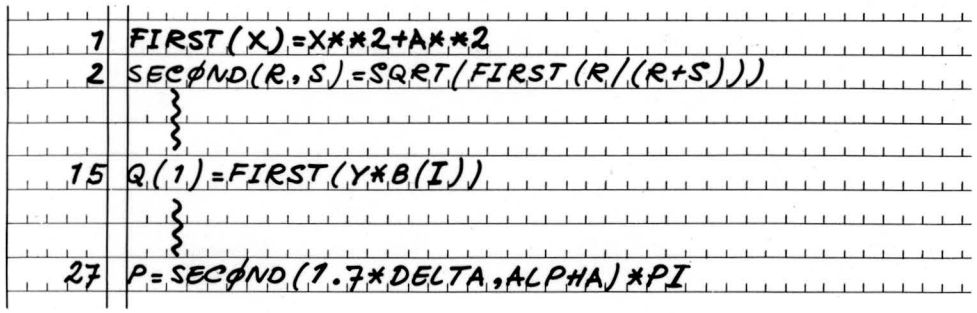
```
   10 D=3.*S+T**2+X+Y+Z
```

*Note:* the formal parameters may occur in more than one statement function definition (they do not exist as variables).

Example 1:

```
      SUM(A,B,C)=A+B+C
      DIFFERENCE (A,B) = A-B
```

Example 2:




*Note:* The actual parameter may be an expression (see statements number 15 and 27).

## 10.2 FUNCTION STATEMENT

In chapter 7 we have already encountered what is known as intrinsic and basic external functions.
There are situations in which it is desirable to define own functions.
This function, however, consists of more than one statement.

One function is defined by a statement of the form

t FUNCTION f($a_1$, $a_2$, ....., $a_n$)

where:
t is INTEGER, REAL, DOUBLE PRECISION, COMPLEX, or nothing;
f is the symbolic name of the function;
$a_1$, $a_2$, ....., $a_n$ are the formal parameters.

## 10.3 RETURN STATEMENT

The group of statements which define this function begins with the FUNCTION statement and must finish with END.
A function must contain RETURN statement as a logical end.
The symbolic name of the function (f) must occur in the function subprogram as the name of a variable; a value must be assigned to this variable.
In a function subprogram no function statement may occur (nesting).
The form of the statement is as stated below.

General form: RETURN

This statement ends the FUNCTION program and returns the control to the program in which the FUNCTION was called.

Example:

```
C     SUM TWO ARRAYS (A AND B)
      FUNCTION SUM(A,B,N)
      DIMENSION A(N),B(N)
      SUM=A(1)+B(1)
      DO 10 J=2,N
   10 SUM=SUM+A(J)+B(J)
      RETURN
      END
```

In the main program, for instance there is the statement

```
      X=SUM(VEC1,VEC2,50)
```

The formal parameters are A, B, N.
The actual parameters are VEC1, VEC2, 50.

*Note:* Agreement (number, location, type) between actual and formal parameters is obligatory.

## 10.4 SUBROUTINES

The subroutine is defined by a statement of the form:

SUBROUTINE s ($a_1$, $a_2$, ....., $a_n$)

or SUBROUTINE s

in which:
s is the symbolic name of the subroutine;
$a_1$, $a_2$, ....., $a_n$ are the formal parameters.

A subroutine begins with one of the above-mentioned statements and ends with END.
A subroutine has a RETURN statement as a logical end.
The subroutine is called with a CALL statement.
The subroutine names must not occur as the symbolic name of a variable in the subroutine.
In a subroutine another subroutine may be called, but no subroutine statement, function statement or statement function may occur in it.
The CALL statement is of the following form:

CALL s ($a_1$, $a_2$, ....., $a_n$)

or CALL s

depending on the subroutine definition.

Example: Copy an array in another array.
main program:

```
      DIMENSION X(100),Y(100)
        .
        .
        .
      CALL COPY (X,Y,50)
```

X, Y and 50 are actual parameters.

subroutine:

```
      SUBROUTINE COPY(A,B,N)
      DIMENSION A(100),B(100)
      DO 10 I=1,N
   10 B(I)=A(I)
      RETURN
      END
```

A, B, N are formal parameters.

*Note:* In a function we must assign a value to the symbolic name of the function; a subroutine name may not occur in a subroutine. The parameters may be assigned a value in a function as well as in a subroutine.

# 11 Specification and DATA/ BLOCK DATA statements

## 11.1 SPECIFICATION STATEMENTS

There are 5 types of specification statements.

a. Type statements (see 4.4);
b. DIMENSION statements (see 6.2.2);
c. EXTERNAL statements;
d. EQUIVALENCE statements;
e. COMMON statements.

### 11.1.1 EXTERNAL statement

We may use the name of a subroutine or of a function as a parameter. To be able to do this, the name must be declared in an EXTERNAL statement.

General form: EXTERNAL a, b, c...

Example:

```
      EXTERNAL MULT
      ~
      CALL SUB(J,MULT,C)
      ~
```

subroutine:

```
      SUBROUTINE SUB(K,N,Z)
      IF (K) 4,6,6
    4 D=N(K,Z**2)
      ~
    6 RETURN
      END
```

function:

```
FUNCTION MULT(I,B)
MULT=I**2+5*B
RETURN
END
```

MULT is used as a parameter in the subroutine SUB.
MULT is assigned to N.
In the subroutine the program part MULT is executed in the statement:
either

```
D=MULT(K,Z**2)
```

or

```
D=J**2+5*C**2
```

### 11.1.2 EQUIVALENCE statement

General form: EQUIVALENCE $(k_1), (k_2), \ldots\ldots, (k_n)$
in which each k is a list of the form
$a_1, a_2, \ldots\ldots, a_m$

By means of this statement we can assign two or more names of variables to one memory location.

Example 1: After having written a large program we realize that we have used various names for the variable X, namely X, X1, RST. In order to avoid having to change all the corresponding names we use the EQUIVALENCE statement.

```
EQUIVALENCE (X,X1,RST)
```

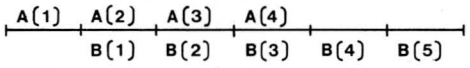
Example 2: If A and B(1) represent the same field
and C, D and E also represent one field, we write:

```
EQUIVALENCE (A,B(1)),(C,D,E)
```

This statement can also be used to obtain a subdivision in our labelled range of storage units.

Example:

```
      DIMENSION A(4),B(5)
      EQUIVALENCE (A(3),B(2))
```




### 11.1.3 COMMON statement

General form: COMMON $/x_1/a_1...../x_n/a_n$
in which every a is a list of names of variables which may be subscripted, and every x is a block name (symbolic name).

The block name is always found between slashes (/.../); the names of the variables are separated by commas.
In the following example variables appearing after the two consecutive slashes, namely RS(2,3), are in blank or unlabelled common.
Alternatively entities may be expressed as in blank common if (a) no block name appears, or (b) they occur between the beginning of a statement and the appearance of a block name.
The size of a common block in a program unit is the amount of the storage required for the elements introduced through COMMON and EQUIVALENCE statements.
It should be noted that the sizes of labelled common blocks with the same label within one executable program must be the same; the sizes of blank common within one executable program do not need to be the same.

Example:

```
      COMMON /ABLOCK/A,B,C/BBLOCK/X(5)//RS(2,3)
```

memory subdivision:

| | |
|---|---|
| ABLOCK | A |
| | B |
| | C |
| BBLOCK | X(1) |
| | X(2) |
| | X(3) |
| | X(4) |
| | X(5) |
| no block name | R |
| | S(1,1) |
| | S(1,2) |
| | S(1,3) |
| | S(2,1) |
| | S(2,2) |
| | S(2,3) |

Example: main program:

```
      DIMENSION A(6),B(10),C(10,10)
      COMMON /ABLOCK/A,B,C
        ~
        ~
      CALL SR(M,N)
        ~
```

We make use of the memory locations called A, B and C in the main program.

subroutine:

```
      SUBROUTINE SR(K,L)
      DIMENSION X(6),Y(10),Z(10,10)
      COMMON/ABLOCK/X,Y,Z
        ~
        ~
      RETURN
      END
```

In the subroutine we use the same locations but, however, these locations are called X, Y and Z.

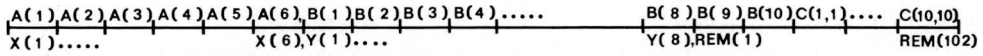
*Note:* The main program and the subroutine use the same range with the block name ABLOCK.

| ABLOCK | main prog. | subr. |
|---|---|---|
| | A's | X's |
| | B's | Y's |
| | C's | Z's |

If we do not desire to use the whole memory locations called A, B and C in the subroutine, we should have to write the following, for example, in the subroutine:

```
SUBROUTINE SR(K,L)
DIMENSION X(6),Y(8),REM(102)
COMMON /ABLOCK/ X,Y,REM
```

The memory location would then be divided as follows:




REM is not used in the subroutine, but, however, must be named.

*Note:*

```
DIMENSION A(6),B(10),C(10,10)
COMMON A,B,C
```

may be substituted by

```
COMMON A(6),B(10),C(10,10)
```

### 11.2 DATA/BLOCK DATA statements

#### 11.2.1 DATA statement

The DATA statement is used to define initial values of variables, arrays or elements. It takes the general form:

DATA $k_1/d_1/k_2/d_2/.....k_n/d_n/$
in which:
k is a list of names of variables;

d is a list of constants in which every d can be preceded by j*;
j is an integer constant.

Example:

```
      DATA A,B,C/5.0,6.1,7.3/,D/25*1.0/
```

The values 5.0, 6.1 and 7.3 are assigned to A, B and C respectively, and the value 1.0 to the first 25 elements of D.

Example: We want to make 10 counters zero. This can be done as follows:

```
      DIMENSION COUNT (10)
      DATA COUNT/10*0.0/
```

### 11.2.2 BLOCK DATA statement

General form: BLOCK DATA

This statement may only occur as a first statement in a subprogram in which specification statements occur. This subprogram exclusively contains the type statements, EQUIVALENCE, DATA, DIMENSION and COMMON statements, and must be terminated with END.

# 12 Input/output and FORMAT statements (Part II)

In chapter 9 we have already considered the simplest input/output and format statements.
In this chapter we shall discuss a few more FORMAT statements and applications of READ and WRITE.

## 12.1 SUBSCRIPTING LIST FOR INPUT/OUTPUT STATEMENTS

If 5 data items are punched on 1 card which we assign to A(1), A(2), A(3), A(4) and A(5) we use the statement:

```
READ (7,20) A(1),A(2),A(3),A(4),A(5)
```

or shorter:

```
READ (7,20) (A(I),I=7,5)
```

(This construction means: implied DO loop.)
This manner of writing recalls the DO statement:

```
      DØ 72 I=7,5
72    READ (7,20) A(I)
```

After execution of the last two statements, the result is that 5 separate cards are read. When subscripting the list we can also use an increment.

```
READ (7,20) (A(I),I=7,10,2)
```

Now the first field is assigned to A(1);
third field is assigned to A(3) etc.;
ninth field is assigned to A(9).

If we write for this list:
((C(I,J), D(I,J), I = 1,5), J = 1,4)
this is the same as:
C(1,1), D(1,1), C(2,1), D(2,1), ..., D(5,1), C(1,2) ... etc.

*Note:* If the card contains less data than is described in the list, the following cards are read until all the data specified in the list is assigned.
The same quantity of data is read or printed as stated in the list.

If A is an array and we write:

```
READ (1,100) A
```

this means that all the elements of the array are read in succession.

Example: a 10 × 10 matrix A has to be read.

```
DIMENSION A(10,10)
READ (1,20) A
```

is the same as

```
READ (1,20) ((A(I,J),I=1,10),J=1,10)
```

which means: read columnwise.

If we wish to read one row after the other, we have to write:

```
READ (1,20) ((A(I,J),J=1,10),I=1,10)
```

## 12.2 FORMAT

We can indicate in the format whether we wish to read a new card or print on a new line, by means of a slash (/).

Example:

```
      WRITE (3,20) A,B,C
20    FORMAT (F10.0/F10.0/F10.0)
```

output: 1st line number A
2nd line number B
3rd line number C

If we wish to skip one line we indicate this with two slashes (//).

```
20 FØRMAT (F10.0//F10.0///F10.0)
                ↑          ↑
          skip 1 line   skip 2 lines
```

If the slashes are at the beginning or the end of the format the same number of lines are skipped as slashes are written.

Output with

```
30 FØRMAT (3F9.2,2F10.4/2I3)
```

Result:

Line 1 (F9.2) (F9.2) (F9.2) (F10.4) (F10.4)
Line 2 (I3) (I3)
Line 3 as line 1
Line 4 as line 2
etc.

Output with

```
40 FØRMAT (I2,3E12.4/2F10.3,3F9.4/(10F12.4))
```

Result:

Line 1 according to I2, 3 E12.4
Line 2 according to 2F10.3, 3F9.4
All following lines according to 10F12.4

### 12.3 FIELD SPECIFICATION I

Integer

Iw
where w is field length.

Input:

Blank is read as zero.

Output:

If the number is less than indicated by the field specification the positions at the left-hand side are filled with blanks.

Examples:

field specification I3

Input

| external | internal | |
|---|---|---|
| 721 | 721 | |
| −72 | −72 | |
| 7bb | 700 | (b is blank) |
| bb1 | 001 | (b is blank) |

Output

| internal | external | |
|---|---|---|
| 721 | 721 | |
| −721 | 721 | (wrong) |
| −12 | −12 | |
| 9 | bb9 | (b is blank) |
| −5 | b−5 | (b is blank) |

## 12.4 FIELD SPECIFICATION F

Real — Fw.d (without E-power notation)
where:
d is the number of digits after the decimal point (fraction);
w is total field length w $\geqslant$ d + 3 (including sign and decimal point).

Input: If a decimal point is punched it is retained.
If no decimal point is punched it is located as indicated in the field specification.

Output: If the fraction is greater than stated in the field specification the fraction is rounded off. If the fraction is less than indicated in the field specification the fraction is extended to the right with zeros.
The rest of the field (the integer part) is dealt with as with integers.

Examples: F5.2

Input

| internal | external |
|---|---|
| 12345 | 123.45 |
| 12.45 | 12.45 |
| 123.4 | 123.4 |

Output

| internal | external | |
|---|---|---|
| 12.17 | 12.17 | |
| −41.16 | 41.16 | (wrong) |
| −.2 | −0.20 | |
| 7.3542 | b7.35 | (rounded off)(b is blank) |
| −1. | −1.00 | |
| 9.03 | b9.03 | (b is blank) |
| 187.64 | 87.64 | (wrong) |

## 12.5 FIELD SPECIFICATION E

Real — Ew.d (E-power notation)
where:
d is the number of digits after decimal point;
w is the field length such that w $\geqslant$ d + 7 (including sign and four positions for E-power including the E and the sign).

Example: E10.3

Output

| internal | external |
|---|---|
| 238. | b0.238Eb03 |
| −.002 | −0.200E−02 |
| .00000000004 | b0.400E−10 |
| −21.0057 | −0.210Eb02 |

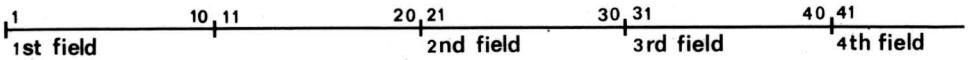
## 12.6 FIELD SPECIFICATION X

Blanks — n X
where n is the number of characters skipped (input)
or the number of blanks inserted (output).

The field specifications describe the fields beginning in column (position) 1, the other fields are combined.
If we wish to insert blanks in the case of output or unread characters in the case of input, we use the X-field specification.

Example: Input:

```
 30 FØRMAT (I10,10X,3I10)
```




Output:
If, for example, we wish to leave the first 8 positions blank, then we shall have to use the format statement:

```
 20 FØRMAT (8X,~~~~~~~~~)
```

The comma after an X-field specification may be deleted.

## 12.7 FIELD SPECIFICATION H

Hollerith nH
where n is the number of characters after the H.

Example:

```
101 FØRMAT (18H THIS IS HØLLERITH)
```

Input: n characters are read and replaced by characters which are specified.

Output: The n characters which are specified (whether already replaced or not by a READ) are output.

Example 1:

```
    WRITE (3,100)
100 FØRMAT (5H GØØD)
```

Output: bGOOD (b is blank)

Example 2: A card is read with a

```
    READ (1,20) K
 20 FØRMAT (4H YES,I5)
```

In the card we read:
column 1
↓
bbNObb234 (b is blank)

From the statement

```
      WRITE (3,20) K
```

we have an output:
pos. 1
↓
bbNObb234 (b is blank)

## 12.8 FIELD SPECIFICATION A

Alphanumeric — rAw
in which:
w is the number of characters read (input) or printed out (output);
r is the repetition factor.

Example:

```
10    FØRMAT (20A4)
        ≀
      READ (7,10) (ALPHA(I),I=7,20)
        ≀
      WRITE (3,10) (ALPHA(I),I=7,20)
```

20 groups of 4 characters are read and assigned to ALPHA(1) to ALPHA(20) after which they are printed.

Example: Check whether character A is found in card column 31; if so, continue at statement number 30.

```
100   FØRMAT (3F10.2,A1)
      DATA ALPHA/1HA/
      READ (7,100) (B(I),I=7,3),C
      IF (C.EQ.ALPHA) GØ TØ 30
```

## 12.9 CARRIAGE CONTROL

Vertical spacing before printing with the aid of H

```
      1  FORMAT (1H )          1 line
      2  FORMAT (1H0)          2 lines
      3  FORMAT (1H1)          to first line of next page
      4  FORMAT (1H+)          no advance
```

The first character of a line is never printed.
This character is used for control.

Example:

```
          K=123
          WRITE (3,100) K
    100   FORMAT (I3)
```

Output pos. 1 of the following page

↓

b23 (b is blank)

Thus in place of field specification I3 we should have written I4.

Output pos. 1 of the following line

↓

b123 (b is blank)

## APPENDIX B

BASIC EXTERNAL FUNCTIONS

| Basic External Function | Definition | Number of Arguments | Symbolic Name | Type of: Argument | Type of: Function |
|---|---|---|---|---|---|
| Exponential | $e^a$ | 1 | EXP | Real | Real |
| | | 1 | DEXP | Double | Double |
| | | 1 | CEXP | Complex | Complex |
| Natural Logarithm | loge $(a)$ | 1 | ALOG | Real | Real |
| | | 1 | DLOG | Double | Double |
| | | 1 | CLOG | Complex | Complex |
| Common Logarithm | $\log_{10}(a)$ | 1 | ALOG10 | Real | Real |
| | | | DLOG10 | Double | Double |
| Trigonometric Sine | sin $(a)$ | 1 | SIN | Real | Real |
| | | 1 | DSIN | Double | Double |
| | | 1 | CSIN | Complex | Complex |
| Trigonometric Cosine | cos $(a)$ | 1 | COS | Real | Real |
| | | 1 | DCOS | Double | Double |
| | | 1 | CCOS | Complex | Complex |
| Hyberbolic Tangent | tanh $(a)$ | 1 | TANH | Real | Real |
| Square Root | $(a)^{\frac{1}{2}}$ | 1 | SQRT | Real | Real |
| | | 1 | DSQRT | Double | Double |
| | | 1 | CSQRT | Complex | Complex |
| Arctangent | arctan $(a)$ | 1 | ATAN | Real | Real |
| | | 1 | DATAN | Double | Double |
| | arctan $(a_1/a_2)$ | 2 | ATAN2 | Real | Real |
| | | 2 | DATAN2 | Double | Double |
| Remaindering* | $a_1$ (mod $a_2$) | 2 | DMOD | Double | Double |
| Modulus | | 1 | CABS | Complex | Real |

* The function DMOD $(a_1, a_2)$ is defined as $a_1 - (a_1/a_2)a_2$, where $(x)$ is the integer whose magnitude does not exceed the magnitude of $x$ and whose sign is the same as the sign of $x$.

## APPENDIX C

---

### PROBLEMS

#### Problem F0010

Which of the following names or constants are permitted?
If permitted: which type?
If not permitted: what errors?

a. MAX $8
b. QUANT
c. -3.78.
d. IOVR7
e. AII
f. 427
g. 71E-8.
h. I71
i. 6ABLE
j. INTEGER
k. KOUNT
l. COUNT
m. A+B
n. 5280.
q. REALCONSTANT

**Problem F0020**

For the following formulae write FORTRAN expressions:

a. $X+Y^4$

b. $(X-Y)^{10}$

c. $A+\frac{B}{C}$

d. $\frac{A+B}{C}$

e. $\left(\frac{A+B}{C+D}\right)^2+X^2$

f. $\frac{A+B}{C+\frac{D}{E+F}}$

g. $1+X+\frac{X^2}{2!}+\frac{X^3}{3!}$

h. $\left(\frac{X}{Y}\right)^{G-1}$

i. $\frac{\frac{A}{B}-1}{G\left(\frac{G}{D}-3\right)}$

j. $\left(\frac{X+A+\pi}{2Z}\right)^2$

k. $AX^3+BX^2+CX+D$

l. $\frac{1}{A^2}\left(\frac{R}{10}\right)^A$

$\pi = 3.14159265359$

**Problem F0030**

Which subscripted variables are wrong and why?

a. ARRAY (LIST)
b. NAME (M * N)
c. A (2* I + 1,3)
d. ANSW (K – 25)
e. BLOCK (I + J)
f. MMM (X, Y, Z)
g. HER (3. * K, L)
h. AA (I, J, K, L)
i. XYZ (5 * K, K – 5,5 * K + 5)
j. SAM (4 * L – K)

**Problem F0040**

Which of the following arithmetic assignment statements are permitted and which are not?

a. A (I, 3) = M2 + J
b. X (I + 2, J) = – 3. * D + (E – F) **.6
c. Y = I ** A
d. A (B) = I + 2
e. A + 3. = B * C
f. M = (A + B)
g. Z = A ** R ** S
h. I (J) = K(J) / J
i. I (A) = H + 14.
j. W = I + ( –D)

**Problem F0050**

Write the following arithmetic assignment statements with
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.

**Problem F0060**

Write arithmetic statements for the following formulae.

a. $AREA = 2PR\sin(\pi / P)$

b. $CHØRD = 2R\sin\frac{A}{2}$

c. $ARC = 2\sqrt{Y^2 + (4X^2/3)}$

d. $S = -\frac{\cos^4 X}{4}$

e. $VAL = \frac{1}{\cos X} + \log\left|\operatorname{tg}\frac{X}{2}\right|$

f. $G = \frac{1}{2}\log\frac{1+\sin X}{1-\sin X}$

g. $Q = \left(\frac{2}{\pi X}\right)^{\frac{1}{2}}\sin X$

h. $B = \frac{e^X \sqrt{2}\cos(\sqrt{X/2} + \pi/8)}{2\pi X}$

i. $Y = (2\pi)^{\frac{1}{2}} X^{X+1} e^{-X}$

j. $E = \left|X \operatorname{arctg}\frac{X}{A} - \frac{A}{2}\log(A^2 + X^2)\right|$

**Problem F0070**

a. Make LARGE equal to the greater of the two variables X and Y.

b. Make LARGE 3 equal to the greatest of the three variables X, Y and Z.

c. The variables R and S can be positive or negative.
Make LARGE ABS equal to the greater of the two in absolute value.

d. An angle THETA is given (0 rad. $\leqslant$ THETA $\leqslant$ 30 rad.).
Reduce THETA a number of times by 2 $\pi$ until THETA is less than 2 $\pi$.

**Problem F0080**

a. XREAL and XIMAG are respectively the real and imaginary part of a complex number.
If the real and the imaginary part are less than 1 in absolute value, jump to statement number 81 or otherwise to statement number 82.

b. Y1, Y2 and Y3 are three points on a curve.
If Y2 $>$ Y1 and Y2 $>$ Y3 then jump to statement number 456, otherwise to statement number 789.

c. If NUMBER is less than $10^{-5}$ in absolute value, jump to statement number 50, otherwise to statement number 51.
   1. Make use of the ABS function.
   2. Make no use of the ABS function.

**Problem F0090**

a. If $0.999 \leqslant X \leqslant 1.001$, jump to statement number 100, otherwise to statement number 101.
   1. Make use of two IF statements without ABS function.
   2. Make use of one IF statement, the expression of which is an ABS function.

b. If I = 1 and R $<$ S, jump to statement number 200.
If I = 1 and R $\geqslant$ S, jump to statement number 201.
If I $\neq$ 1, jump to statement number 202.

c. If N = 1, 2 or 8, jump to statement number 11.
If N = 3 or 7, jump to statement number 10.
If N = 4, 5 or 6, jump to statement number 15.
If N is not equal to one of the values named, jump to statement number 100.

**Problem F0100**

Which of the following FORTRAN statements are permitted?

a. DO 10 FEW = 1, 10, 4
b. DO 51 K = 1,K – 1,3
c. GOTO M
d. GO TO (1, 2, 3), A
e. GO TO 1 87
f. IF (A – I) 1, 2, 2
g. GOTO 3, 4
h. DO 1, J = 1, 7, 1
i. GOTO (4, 5, 6) K
j. GO TO N – 12

**Problem F0110**

a. Write a program part which sums the last 50 elements of a one-dimensional array A (150 elements) and assigns the sum to SUM.

b. Two one-dimensional arrays A and B each contain 30 elements.

Program: $D = \left\{ \sum_{i=1}^{30} (A_i B_i)^2 \right\} \frac{1}{2}$

c. Two one-dimensional arrays A and B each contain a maximum of 18 elements.
N is a positive integer not greater than 18.

Program: $C = \sum_{k=1}^{N} A_k B_k$

**Problem F0120**

a. A one-dimensional array A contains 1000 elements.
Locate the array A in the reverse order in the one-dimensional array B:
B(1000) is substituted by A(1), B(999) by A(2), etc.

b. A one-dimensional array M contains 20 elements.
Substitute every element by the element itself multiplied by its index. In other words, substitute $m_i$ by $i * m_i$ for $i = 1, 2, \ldots\ldots, 20$.

c. A one-dimensional array X contains 50 elements.
Make another array DX of 49 elements according to DX(I) = X(I+1) – X(I) for I = 1, 2, ....., 49.

**Problem F0130**

a. A one-dimensional array B contains 50 elements.
Assign the greatest element to LARGE and the subscript of the greatest element to NUMBER.

b. Two one-dimensional arrays X and Y each contain 50 elements.
The variable XS is equal to one of the elements of array X.
If $XS = X_i$, then assign $Y_i$ to YS.

c. The number of elements of a one-dimensional array K is unknown (max. 100). The array (positive elements) is terminated with a negative number. Find the normal rounded-off average of the array K and assign this to MEAN.

**Problem F0140**

a. Three two-dimensional arrays A, B and C each have 15 rows and 15 columns. Calculate the elements of C according to the following:

$$C_{ij} = \sum_{k=1}^{15} A_{ik} B_{kj} \quad \text{for } i, j = 1, 2, \ldots, 15.$$

b. A two-dimensional array MATRIX has 20 rows and 20 columns.
Calculate the product of the elements which form the main diagonal and assign this to MPROD.

**Problem F0150**

In these problems data is read with the card reader, followed by a calculation in which the results are to be printed by the line printer.

a. read: A, B, C
print: A, B, C, S, AR
calculation: $S = \frac{A+B+X}{2}$ and $AR = \sqrt{S(S-A)(S-B)(S-C)}$

b. read: A, B, C, X
print: A, B, C, X, R
calculation: $R = \frac{B.C}{12}\left\{6X^2\left(1-\frac{X}{A}\right)^2 + B^2\left(1-\frac{X}{A}\right)^4\right\}$

c. read: A, E, H, P
print: A, E, H, P, X
calculation: $X = \frac{E.H.P}{(\sin A)\left(\frac{H^4}{16} + H^2P^2\right)}$

d. read: A, X, S
print: A, X, S, Y, Z
calculation: $Y = \sqrt{X^2 - A^2}$ and $Z = \frac{X.S}{2} - \frac{A^2}{2}\log|X+S|$

**Problem F0160**

Define the arithmetic statement function

$$\text{FØRM}(X) = X^2 + \sqrt{1 + 2X + 3X^2}$$

and then compute:

$$\text{ALPHA} = \frac{6.9 + Y}{Y^2 + \sqrt{1 + 2Y + 3Y^2}}$$

$$\text{BETA} = \frac{2.1.(Z + Z^4)}{Z^2 + \sqrt{1 + 2Z + 3Z^2}}$$

$$\text{GAMMA} = \frac{\sin Y}{Y^4 + \sqrt{1 + 2Y^2 + 3Y^4}}$$

$$\text{DELTA} = \frac{1}{\sin^2 Y + \sqrt{1 + 2\sin Y + 3\sin^2 Y}}$$

**Problem F0170**

Define the arithmetic statement function:

$$\text{FUNLØG}\ (A) = 2.5\ \log\left(A + A^2 + \frac{1}{A}\right)$$

and then compute:

$$R = X + \log X + 2.5\ \log\left(X + X^2 + \frac{1}{X}\right)$$

$$S = \cos X + 2.5\ \log\left(1 + X + (1 + X)^2 + \frac{1}{1 + X}\right)$$

$$T = 5\ \log\left\{(A - B)^3 + (A - B)^6 + \frac{1}{(A - B)^3}\right\}$$

**Problem F0180**

Write a function programme to compute

$$Y(X) = \begin{cases} 1+\sqrt{1+X^2} & \text{for } X < 0 \\ 0 & \text{for } X = 0 \\ 1-\sqrt{1+X^2} & \text{for } X > 0 \end{cases}$$

Then write the statements for the following formulae:

$$F = 2 + Y(A + Z)$$

$$G = \frac{Y\{X(K)\} + Y\{X(K+1)\}}{2}$$

$$H = Y\{\cos(2\pi X)\} + \sqrt{1+Y(2\pi X)}$$

**Problem F0190**

Write the function program to compute:

$$RHØ\ (A,B,N) = \frac{A}{2\pi} \sum_{i=1}^{N} B_i \quad (N \leqslant 50)$$

in which B is a one-dimensional array which contains 50 elements.
Use this function to calculate $\frac{1}{2\pi}$ times the sum of the first 18 elements of a one-dimensional array A and assign this result to X.

**Problem F0200**

a. A is a one-dimensional array with a maximum of 50 elements.
Write a subroutine to compute the average of the first N elements and keep a count of the number of these elements that are zero.

b. Given the variables A, C, B, X and L,
write a subroutine to compute R, S and T from:

$$R = \sqrt{A + BX + CX^L}$$

$$S = \cos(2\pi X + A) \cdot e^{BX}$$

$$T = \left(\frac{A+BX}{2}\right)^{L+1} - \left(\frac{A-BX}{3}\right)^{L-1}$$

**Problem F0210**

a. Five real numbers are given on a punched card.
Between 2 real numbers are 5 blanks.
The real numbers are:
8 digits, 1 decimal place
10 digits, 8 decimal places
6 digits, 2 decimal places
7 digits, no decimal place
2 digits, 1 decimal place.

Write the format statement.

b. The allocation on a punched card is:

| col. | | name of the variable | |
|---|---|---|---|
| 1- 3 | ± dd | LGA | |
| 4- 6 | ddd | IGL | |
| 7-20 | ± d.ddddddE±dd | BAL | d = digit |
| 21-34 | ± d.ddddddE±dd | ATL | |

Write the statements to read this card.

c. DATA is a one-dimensional array of 10 elements, consisting of 7 digits.
The value of n is punched in columns 1-2 of a card; the columns 3-80 contain a variable number of elements of DATA.The number of elements is given by n. Every number has a decimal point and no exponent.

Write the statement to read such a card.

**Problem F0220**

a. 10 real numbers, consisting of 6 digits and a decimal point (4 digits after the decimal point), without E notation, are punched on one card.
Write the statement to read such a card, and assign the read-in values to the elements 11 through 20 of the array X.

b. Write the statements:
1. To print a heading line:
RESULTS OF THE CALCULATION;
2. followed by a blank line;
3. followed by 10 lines containing on each line:
1 value of the array ANSW.
Each element of ANSW consists of 20 positions with 7 digits after the decimal point (E notation).

## APPENDIX D

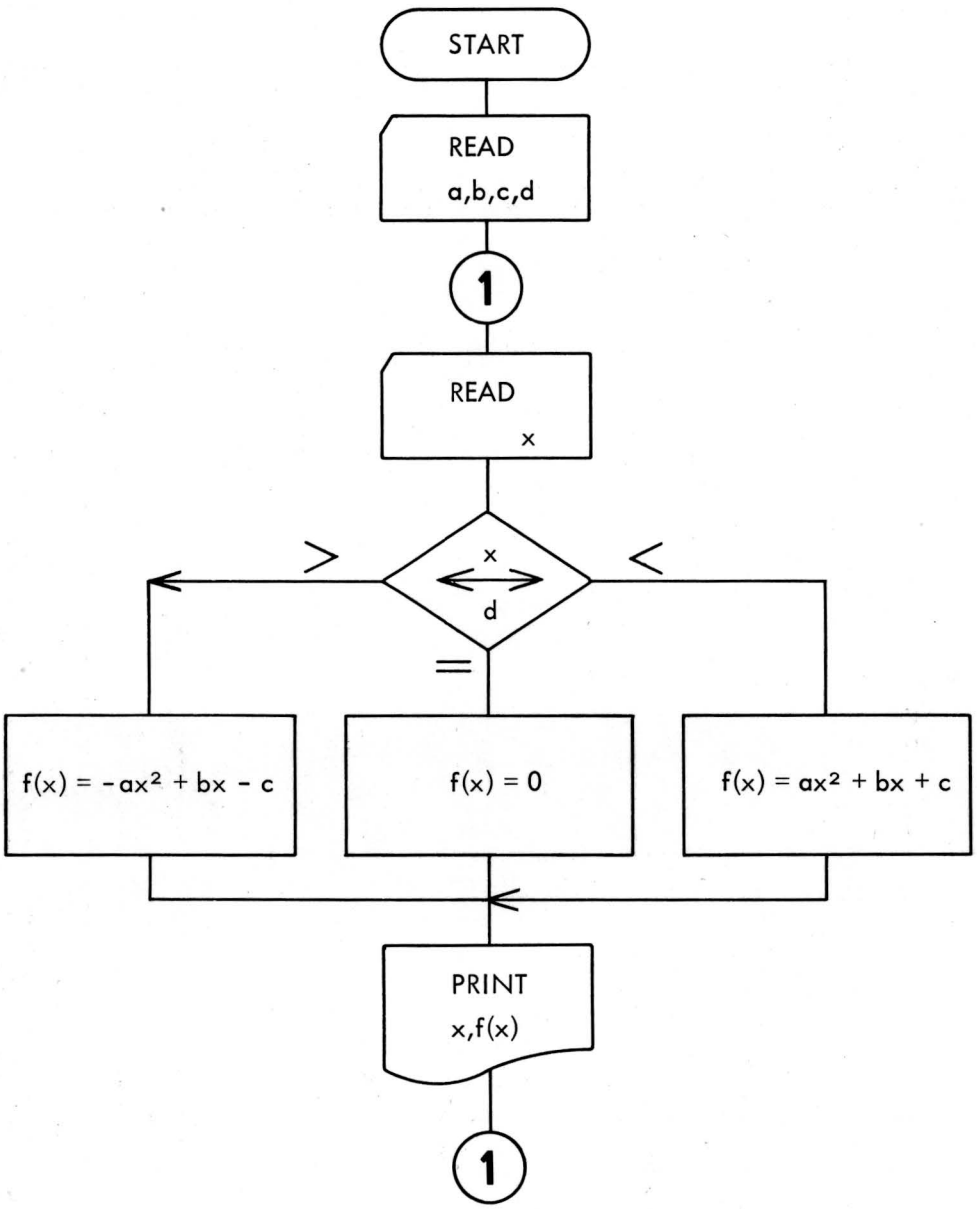
## EXAMPLES

**Example 1: Problem**
The values a, b, c and d are given on a punched card.
The values of the variable x are on the following cards (1 per card)

$$\text{Compute the function } f(x) = \left\{ \begin{array}{ll} ax^2+bx+c & \text{for } x < d \\ 0 & \text{for } x = d \\ -ax^2+bx-c & \text{for } x > d \end{array} \right\}$$

for every value of x and print the value of x and f(x).

**Example 1: Flow diagram**



PHILIPS

data systems

## FORTRAN

Problem _____ Programmer _____ Date _____ Page _____ of _____

| STATEMENT 1 5 | 6 | 7 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 | IDENTIFICATION 73 80 |
|---|---|---|---|
| C | | | |
| | | READ (1,10) A,B,C,D | |
| 1 | | READ (1,70) X | |
| | | IF (X-D) 2,3,4 | |
| 2 | | FX=A*X**2+B*X+C | |
| | | GØTØ 5 | |
| 3 | | FX=0.0 | |
| | | GØTØ 5 | |
| 4 | | FX=-A*X**2+B*X-C | |
| 5 | | WRITE (3,70) X,FX | |
| | | GØTØ 7 | |
| 70 | | FØRMAT (4F14.5) | |
| | | END | |

1 5 6 7 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 73 80

380.00

5122 991 91793

**Example 2: Problem**

The population of a town is divided into 20 age groups.
The group numbering is 0, 5, 10, ....., 95 respectively for the ages 0-4 yrs., 5-9 yrs., 10-14 yrs., ....., 95-99 yrs.
The information is on punched cards (1 card per person).
Card division: columns 1-3 age (I3)
4-13 income (F10.2)

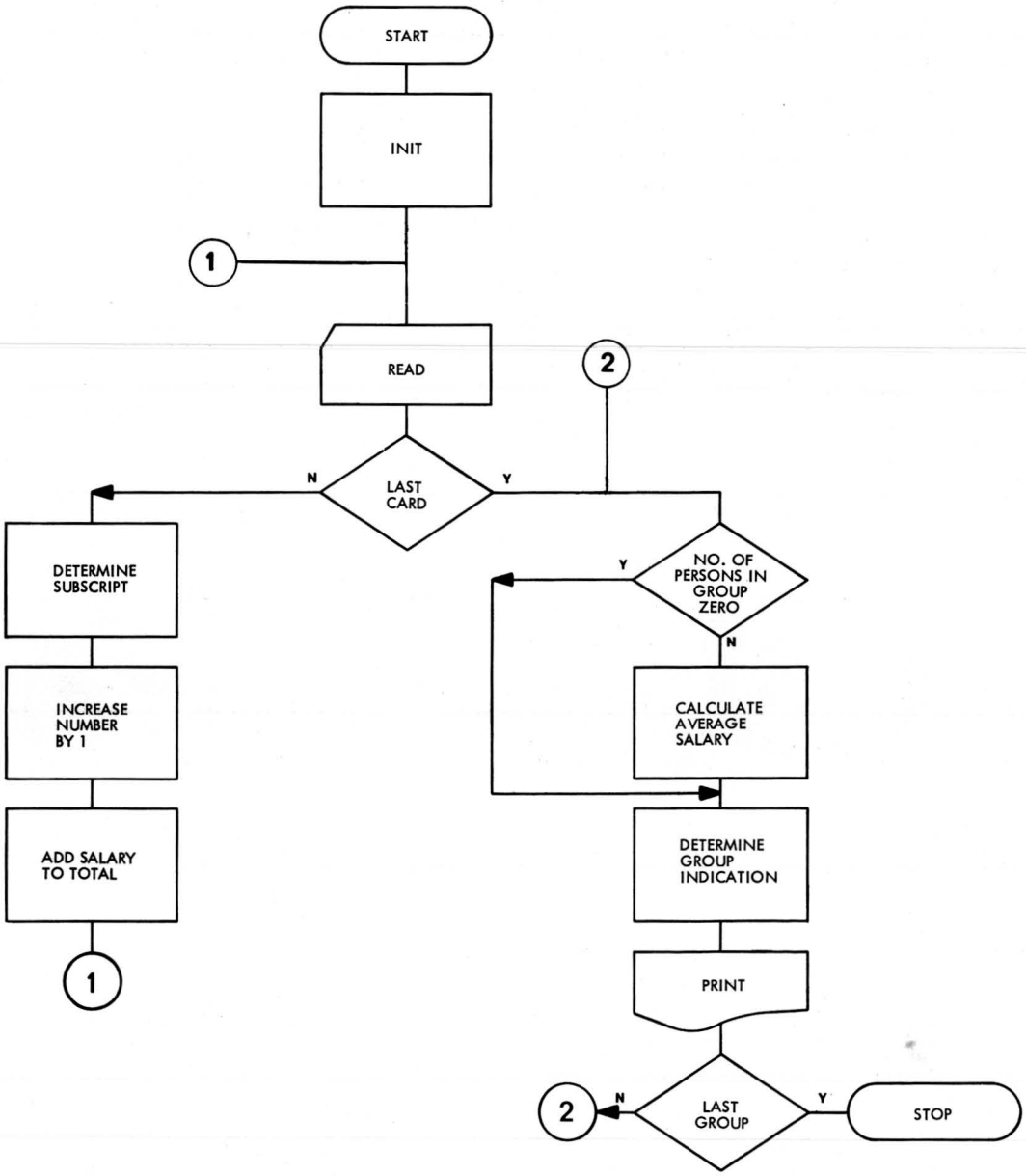
Last card: age is $-1$.
Compute the average income per age group.
Print for each age group the group number and the average income.

*Note:* consider the possibility that the number of persons in an age group may be zero.



CDT. PA 03224 A

**Example 2: Flow diagram**



**PHILIPS**

data systems

## FORTRAN

Problem _____ Programmer _____ Date _____ Page _____ of _____

| STATEMENT 1 5 | 6 | 7 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 | IDENTIFICATION 73 80 |
|---|---|---|---|
| C | | CALCULATING AVERAGE SALARY PER AGE GROUP | |
| | | INTEGER A(20) | |
| | | DIMENSION S(20) | |
| 100 | | FORMAT (I3,F10.2,I7) | |
| | | DO 101 N=1,20 | |
| | | A(N)=0 | |
| 101 | | S(N)=0.0 | |
| 1 | | READ (1,100) K,SAL | |
| | | IF (K+1) 8,4,2 | |
| 2 | | N=K/5+1 | |
| | | A(N)=A(N)+1 | |
| | | S(N)=S(N)+SAL | |
| | | GO TO 1 | |
| 4 | | DO 7 N=1,20 | |
| | | IF (A(N)) 8,6,5 | |
| 5 | | S(N)=S(N)/A(N) | |
| 6 | | K=5*N-5 | |
| 7 | | WRITE (3,100) K,S(N),A(N) | |
| 8 | | STOP | |
| | | END | |



CDT.PA03225A

380.00 5122 991 91793

## APPENDIX A

INTRINSIC FUNCTIONS

| Intrinsic Function | Definition | Number of Arguments | Symbolic Name | Type of: Argument | Type of: Function |
|---|---|---|---|---|---|
| Absolute Value | $\|a\|$ | 1 | ABS | Real | Real |
| | | | IABS | Integer | Integer |
| | | | DABS | Double | Double |
| Truncation | Sign of $a$ times largest integer $\leqslant \|a\|$ | 1 | AINT | Real | Real |
| | | | INT | Real | Integer |
| | | | IDINT | Double | Integer |
| Remaindering* (see note below) | $a_1 \pmod{a_2}$ | 2 | AMOD | Real | Real |
| | | | MOD | Integer | Integer |
| | | 2 | DMOD | Double | Double |
| Choosing Largest Value | Max $(a_1, a_2, \ldots)$ | $\geqslant 2$ | AMAX0 | Integer | Real |
| | | | AMAX1 | Real | Real |
| | | | MAX0 | Integer | Integer |
| | | | MAX1 | Real | Integer |
| | | | DMAX1 | Double | Double |
| Choosing Smallest Value | Min $(a_1, a_2, \ldots)$ | $\geqslant 2$ | AMIN0 | Integer | Real |
| | | | AMIN1 | Real | Real |
| | | | MIN0 | Integer | Integer |
| | | | MIN1 | Real | Integer |
| | | | DMIN1 | Double | Double |
| Float | Conversion from integer to real | 1 | FLOAT | Integer | Real |
| Fix | Conversion from real to integer | 1 | IFIX | Real | Integer |
| Transfer of Sign | Sign of $a_2$ times $\|a\|$ | 2 | SIGN | Real | Real |
| | | | ISIGN | Integer | Integer |
| | | | DSIGN | Double | Double |
| Positive Difference | $a_1 - \text{Min}(a_1, a_2)$ | 2 | DIM | Real | Real |
| | | | IDIM | Integer | Integer |
| Obtain Most Significant Part of Double-Precision Argument | | 1 | SNGL | Double | Real |
| Obtain Real Part of Complex Argument | | 1 | REAL | Complex | Real |

* The functions MOD, AMOD and DMOD are defined as $a_1 - (a_1/a_2)a_2$, where $(x)$ is the integer whose magnitude does not exceed the magnitude of $x$ and whose sign is the same as the sign of $x$.

| Intrinsic Function | Definition | Number of Arguments | Symbolic Name | Type of: Argument | Type of: Function |
|---|---|---|---|---|---|
| Obtain Imaginary Part of Complex Argument | | 1 | AIMAG | Complex | Real |
| Express Single-Precision Argument in Double-Precision Form | | 1 | DBLE | Real | Double |
| Express Two Real Arguments in Complex Form | $a_1 + a_2\sqrt{-1}$ | 2 | CMPLX | Real | Complex |
| Obtain Conjugate of a Complex Argument | | 1 | CONJG | Complex | Complex |


data systems