Program Product

APL\360-OS (5734-XM6) and APL\360-DOS (5736-XM6) General Information Manual

IBM

Program Product

APL\360-OS (5734-XM6) and APL\360-DOS (5736-XM6) General Information Manual

This publication provides a general introduction to the use, operation, and installation of APL\360. APL\360 is a conversational time-shared terminal system utilizing APL (A Programming Language), a concise notation which makes use of well known mathematical symbols, plus a number of symbols that are used for designating other simple and powerful functions. Because APL is closely related to mathematical notation, and its functions operate on sets of information as well as individual data items, the user can write effectively in this language with a minimum of instruction.

APL\360 is designed to operate under either (a) the IBM System/360 Operating System (OS/360), Multiprogramming with a Fixed Number of Tasks (MFT with subtasking) or Multiprogramming with a Variable Number of Tasks (MVT) or (b) the IBM System/360 Disk Operating System (DOS/360).

The manual is in three parts:

Part I describes APL\360 as viewed by a user at a typewriter-like terminal. The statements and commands used to describe the operations to be performed by the APL\360 System are summarized.

Part II describes APL\360 as viewed from the central data processing system location. The system features which facilitate the management and operation of an APL\360 System are summarized.

Part III describes the items to be considered in planning for the installation of an APL\360 System. Minimum machine requirements and storage requirements are provided.

IBM

Second Edition (December 1970)

This edition applies to Version 1, Modification Level 0, of the program products APL\360-OS (5734-XM6) and APL\360-DOS (5736-XM6) and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters.

Changes are continually made to the information herein. Therefore, before using this publication, consult the latest System/360 SRL Newsletter (GN20-0360) for the editions that are applicable and current.

Copies of this and other IBM publications can be obtained through IBM branch offices.

A form has been provided at the back of this publication for readers' comments. If this form has been removed, address comments to: IBM Corporation, Technical Publications Department, 1133 Westchester Avenue, White Plains, New York 10604.

CONTENTS

INTRODUCTION 1
APL\360: A Programming Language 1
Some Characteristics of APL. 1
Concise Notation 1
Easy Interpretation of APL Statements. 2
Handling of Numbers and Tables 2
User-Defined Functions 3
Wide Spectrum of Applications. 3
The Implementation of APL\360. 4
Characteristics of the APL\360 Program Products. 4

PART I: THE USER's VIEW OF APL\360. 6
The APL\360 System 6
APL\360 Organization 6
Locks, Keys, and Security. 8
APL Primitive Functions. 9
Execution Mode and User Defined Functions. 18
Using APL\360. 23
Entries from the Keyboard. 23
The APL Character Set. 23
User Terminal Equipment. 24
A Brief Description of the APL\360 Functions 24
Standard Scalar Functions. 24
Generalized Matrix Operations. 26
Generalized Reduction. 26
Compression and Expansion. 26
Other Functions. 27
Symbols Having Special Uses. 28
A Brief Description of APL\360 29
Classification of Commands 29
System and Program Information 32

PART II: OPERATING APL\360. 34
Introduction 34
The Recording Terminal 34
Privileged System Commands 35
Operator's Workspace 35
APL\360 Offline Utility Program. 37

PART III: INSTALLATION PLANNING 39
Introduction 39
System Configuration 39
Minimum System Requirements--APL\360-OS. 40
Minimum System Requirements--APL\360-DOS 40
Terminals. 41

Storage Estimates. 45
Parameters 45
Estimating Core. 46
Estimating Swap Extent 46
Estimating Minimum Library Disk Storage. 47
Functionally Restricting APL\360 to Require Less Than The Minimum Specified Partition/Region Size 47
Priorities in a Multiprogramming Environment 48
Performance. 48

APPENDIX A: SAMPLE TERMINAL SESSION 50

BIBLIOGRAPHY 64

INTRODUCTION

APL\360: A PROGRAMMING LANGUAGE

The APL Program Products presented here consist of computer programs to interpret and execute procedures written in APL. Thus, there are two separate aspects which will be distinguished throughout this manual:

- The formal language: APL
- An implementation: APL\360

SOME CHARACTERISTICS OF APL

APL is a developing language based on a notation defined by K. E. Iverson and published in 1962 in his book entitled A Programming Language. The objective was to take a fresh start in defining procedures in both data processing and computation from the problem definition point of view, rather than basing it on a current state of computer technology.

The result is a very concise language which makes generous use of symbols for specifying operations, and which allows complex procedures to be defined with a minimum number of characters or keystrokes.

CONCISE NOTATION

The concise notation promotes ease of learning and exploits the use of well known mathematical symbols:

ADD	+
SUBTRACT	-
MULTIPLY	×
DIVIDE	÷

and extends these to logical functions and relations:

AND	∧
OR	∨
EQUAL TO	=
NOT EQUAL TO	≠

LESS THAN <
GREATER THAN >

It also defines a number of special symbols used in other functions. One of these, the left arrow (←), is used for assigning a value. For example,

```
A ← 5
```

means assign 5 to A. Another special symbol, the right arrow (→), is used to effect branching to a statement. For example,

```
→ B
```

means branch to the statement labelled B.

Because of APL's close correspondence to mathematical notation, a user will be able to write in this language with a minimum of instruction.

EASY INTERPRETATION OF APL STATEMENTS

The rules of reading an APL statement are simple. All the functions have the same priority -- there are no complex hierarchy rules. Special sequences of functions can be handled with parentheses in an obvious way. Thus, the value of C in

```
C←5 + (2×4) + 4
```

is 17. Functions are executed from right to left, except as indicated by the parentheses. Non-programmers adapt very quickly to the rules of APL. Moreover, experienced programmers soon overcome their early confusion when they can lay aside the hierachy rules of other languages.

HANDLING OF NUMBERS AND TABLES

APL functions handle sets of numbers and tables (matrices) as well as single numbers. Thus,

```
3 + 12 7 4 15
```

results in the set of numbers

```
15 10 7 18
```

Consider another example:

```
PAYMENT ← QUANTITY × PRICE
```

where *QUANTITY* could consist of the quantities of hundreds of items, *PRICE* the hundreds of corresponding prices. The result is a set of hundreds of numbers for the corresponding payments. APL therefore avoids many of the programmed "loops" of other languages which handle one number at a time.

USER-DEFINED FUNCTIONS

Whenever a program is written in APL, it becomes a defined function so that its name can be used in the same way in which the basic symbols such as + - × ÷ are used. For example:

```
A ← 3 × (5×4÷2)
```

means that A will take on the value of 30, which is 3 times the area of a triangle (one half the height of 5 times the base length of 4). Another way to write this is:

```
A ← 3 × (5 TRIAREA 4)
```

where *TRIAREA* is the name of a program that calculates the area of a triangle. In this example A would be set to 30, which would be 3 times the area of a triangle which has a height of 5 and a base length of 4.

Thus, in APL each program extends the set of functions available in the language for that user. This has powerful implications for interactive computing and for implementing problem oriented languages.

WIDE SPECTRUM OF APPLICATIONS

Although APL is based on a mathematical notation, it can also be used to describe data processing procedures. In fact, it demonstrates how mathematics, including logical facility, can encompass all data processing including "commercial applications". Much of this data processing consists of handling groups of numbers or tables of data, which is precisely where APL excels.

APL can also handle sets of characters in the same way it processes sets of numbers. A set of characters is otherwise known as alphabetic text. Hence there are many popular applications of APL in information retrieval, text editing, budget control, financial analysis, forecasting, and computer assisted instruction.

THE IMPLEMENTATION OF APL\360

The language described above could be implemented in many different ways ranging from batch processing to time-sharing. These program products are an implementation based on interactive use at multiple terminals connected to a single central processing unit.

CHARACTERISTICS OF THE APL\360 PROGRAM PRODUCTS

Each user can regard his terminal as an APL computer with its own memory. This memory is called the active workspace and contains both programs and data. This active workspace can be stored in the libraries in the central computer. Similarly, a workspace can be loaded from the libraries into the active workspace.

Each user controls the operation of libraries and workspaces with simple commands. APL is supplemented by commands for such operations as signing on and off, loading and saving workspaces, dropping workspaces from the library, or erasing programs and data in the active workspace.

There are only two modes of operation:

- The normal or execution mode in which statements or expressions are entered and immediately executed. This implicitly provides a "desk calculator mode" in which the full power of the APL language is available, including any predefined programs in the active workspace.

- Program definition mode in which programs are written statement-by-statement and provided with program names. This mode includes a powerful editing facility for adding, deleting, or changing statements.

These program products implement the concept of "visual fidelity". This means that each line entered is interpreted as it appears on the printed page regardless of the sequence in which the individual characters may have been entered. This feature helps to build confidence, makes reconstructing events easy, and is particularly useful in tutorial situations.

There are two APL\360 program products. One is for installations using DOS/360 and the other for installations using OS/360. The DOS version can also be set up in a dedicated mode or run under Control Program-67 (CP-67) on a System/360 Model 67. However, the terminal user is oblivious to which version he is connected to; the APL commands and functions he uses are the same.

The security protection in the APL\360 System is an important consideration, and is implemented on four levels:

- Each user is given an account number and can add his own password to his assigned number. He can change this password any time he chooses. The identification number as well as the current password is necessary in order to sign on.

- Only the user who stored a given workspace in the libraries can modify or drop that workspace.

- A password may be assigned to each workspace stored in the libraries, and it may be changed at any time. No one can access a workspace which has a password without entering the current password along with the workspace name.

- Programs within a workspace may be "locked". This means that anyone who can access that workspace can execute the locked programs but cannot list the statements that constitute the program. This facility is useful in computer assisted instruction and whenever an author wishes to keep his code proprietary. Once a program is locked, it cannot be unlocked.

PART I: THE USER'S VIEW OF APL\360

THE APL\360 SYSTEM

APL\360 ORGANIZATION

Workspaces

The APL\360 System executes system commands and APL statements entered by a user on a terminal typewriter. Commands allow the user to control the general environment. APL statements are used to perform computation and other work (see Example 1).

APL\360 SYSTEM COMMANDS	APL\360 STATEMENTS (with answers shown)
)*OFF*)*ERASE A*)*SAVE*	20+40÷8 25 4×¯2 ¯8

Example 1. Commands and statements

The APL\360 System is built on the notion of a workspace. When the terminal user signs on the system, he is given a workspace, part of which is set aside to serve the internal workings of the system; the remainder is used, as required, for storing data, programs, and transient information generated in the course of a computation.

When in use, a workspace is said to be active, and it occupies a block of working storage in the central computer. The size of the block, which determines the capacity of each workspace in the system, is set at a fixed value by the central installation.

Inactive workspaces are stored in libraries, where they are identified by names supplied by the user. They occupy space in secondary storage facilities of the central computer and

cannot be used directly. When required, a copy of a stored workspace in a library can be made active, or selected information may be copied from them into an active workspace.

Libraries

Libraries in APL\360 are either private or public. Private libraries are associated with individual users of the system, and are identified by the user's account number. As will be seen later, private libraries are protected against unauthorized usage. Public libraries are not associated with an individual user and may be accessed by any user of the system.

COMMAND	MEANING
)SAVE AVERS	Naming and saving a workspace.
)LOAD AVERS	Loading a workspace from the user's library.
)LOAD 1 NEWS	Loading a workspace from public library #1.

Example 2. Library commands

Control of Workspaces and Libraries

The APL\360 System commands are used to control these workspaces and libraries. In addition to the commands involving signing on and off, there are commands to clear workspaces, to load workspaces from the library, to copy parts of one workspace into another, to erase parts of or entire workspaces, and to save and lock workspaces. Another group of commands is used to inquire into the state of workspaces: to list the names of workspaces in libraries, to list names of defined functions and variables, and names of groups of variables and functions in the workspaces. Messages may be sent from one user to another, or to the computer operator.

In general, APL\360 commands fall into five categories:

- Signing on and off the system
- Controlling the state of the active workspace
- Saving or dropping workspaces in the library
- Asking for reports on the contents of workspaces and libraries
- Sending messages to the computer operator or to other terminal users

A complete list of APL\360 system commands is given at the end of Part I.

LOCKS, KEYS, AND SECURITY

The need for adequate data and program protection is increased in the time-sharing environment where multiple users are simultaneously using the system. Each user's programs and data must be protected against accidental destruction by other users. Confidential data must be safeguarded against unauthorized access.

Stored workspaces and the information they hold can be protected against unauthorized use by associating a lock, comprising a colon and a password of the user's choice, with the name of the workspace when the workspace is saved. In order to activate a locked workspace or to copy any information it contains, a colon and the password must again be used, as a key, in conjunction with the workspace name. Since listings of workspace names, including those in public libraries, never reveal the keys and do not overtly indicate the existence of a lock, added protection is ensured.

Account numbers can be similarly protected by locks and keys, thus maintaining the security of a user's private library and avoiding unauthorized charges against his account.

Passwords for locks and keys may be formed of any sequence of alphabetic and numeric characters without blanks. Characters beyond the eighth are ignored. In use as either a lock or key, a password follows the number or name it is protecting, from which it is set off by a colon.

The APL\360 System offers four levels of security to the user:

- The user account number to which the user can attach his own password.
- User control over his own libraries: only he can modify or drop a workspace from his own library.
- A password, which may be assigned to each workspace as it is saved into the library. No one can access this workspace without knowing the associated password.
- Locking of individual function definitions within a workspace. This means that the program can be used, but cannot be listed or modified.

APL PRIMITIVE FUNCTIONS

APL is easy to start using since it makes use of much standard mathematical notation. Using the system in execution mode, so that calculations are performed immediately and answers printed, is simple; however, very powerful APL functions can be used. For example, to sort a string of numbers into descending sequence, the string is entered and then sorted using the GRADE DOWN function of APL:

```
      N←7 2 5 9 3 4 6 8 12
      N[⍒N]
12  9  8  7  6  5  4  3  2
```

or, if the user wants the numbers in ascending sequence, GRADE UP may be used:

```
      N[⍋N]
2  3  4  5  6  7  8  9  12
```

Many of the APL functions will be familiar to users:

+ PLUS

- MINUS

× TIMES

÷ DIVIDE

< LESS THAN

≤ LESS THAN OR EQUAL TO

= EQUAL TO

≥ GREATER THAN OR EQUAL TO

> GREATER THAN

≠ NOT EQUAL TO

In addition, there are many that are easy to use, even for those who have no mathematical background. For example:

⌈ MAXIMUM

⌊ MINIMUM

There are, in addition to the functions listed above, many other powerful APL functions. For example, there are the log and natural log, trigonometric functions and their inverses, combinatorial, factorial, logic operators, and many others. APL functions are described under the heading "A Brief Description of the APL Functions".

Manipulation of Numbers

APL functions may be used with numbers:

```
      5+4
9
      8⌈7
8
```

or with sets of numbers where the operation is performed on each element in the set. For example:

```
      5 + 4 2 7 3      (Add 5 to each element)
9  7  12  8

      1 2 × 3 4        (Multiply corresponding elements)
3  8

      2 7 5 = 7 7 7    (Indicate equal corresponding ele-
0  1  0                 ments by a one)
```

One of APL's most powerful features is this ability to handle lists of numbers (vectors) or tables (matrices and arrays) with many of the same functions used with single numbers.

Most of the APL symbols are used with two arguments, a left-hand and a right-hand argument. Note in the examples below that these arguments may be groups of numbers:

```
      7  5  23  ⍳  5
2                          (Where in 7 5 23 is the first
                            occurrence of 5?
                            ans:  in position 2)

      3↑2  5  9  7  4
2  5  9                    (Take the first 3 elements from
                            the vector on right)
```

Many symbols can also be used with one argument:

```
      ⍳5
1  2  3  4  5               (Generate the integers from
                            1 to 5)
```

Manipulation of Character Strings

APL can also work on character strings. Some examples of manipulation of alphabetic data are shown in Examples 3A, 3B, and 3C.

Problem: Enter names and print them in a list. (Note that in our example blanks must be added to make all the names the same length, 8 characters.)

Method 1 Execution mode (See Example 4D)

```
      4 8 ⍴ 'HOPKINS CROCKER FLOOD   STANFORD'
HOPKINS                    (Enter the string of char-
CROCKER                    acters enclosed in quotation
FLOOD                      marks.  Rearrange to print 4
STANFORD                   rows of 8 characters each.)
```

Example 3A. Manipulating alphabetic information

Problem: Sort an alphabetic string

```
      S←'RTODESBA'
      A←'ABCDEFGHIJKLMNOPQRSTUVWXYZ'   (Enter the string of
                                        characters.)

      S[⍋A⍳S]                           (Rearrange each char-
ABDEORST                                acter of S to the or-
                                        der in which it
                                        appears in A.)
```

Example 3B. Using APL with alphabetic data

```
Problem:  How many A's are in ABRACADABRA?

      +/'A'='ABRACADABRA'     (Generate a 1 for every letter
5                              equal to A, then add all the
                               1's.)
```

Example 3C. Using APL with alphabetic data

Manipulation of Arrays

APL gets much of its power and simplicity from the way it handles arrays. Basic operations which apply to single values can be applied with equal ease to the processing of vectors, matrices, and entire arrays. In APL\360 there is no limit to the number of dimensions that an array can have.

In addition to the scalar functions that apply to arrays, element-by-element, there are many other functions that make array handling easy. Some examples of array manipulation are shown below:

```
                                  Entering and printing arrays

      A←2 3 ⍴⍳6                   Generate the numbers 1 to 6
      A                           and rearrange into 2 rows of
1  2  3                           3 columns
4  5  6

      B←2 3⍴4 2 7 1 5 8           Rearrange the numbers on the
      B                           right into 2 rows of 3 col-
4  2  7                          umns
1  5  8

      X←3 4⍴'ABCDEFGHIJKL'        Rearrange the letters enclosed
      X                           in quotation marks into 3 rows
ABCD                              of 4 columns
EFGH
IJKL
```

```
      A
1  2  3
4  5  6

      B
4  2  7
1  5  8

      A+B
5   4  10
5  10  14

      A×B
4   4  21
4  25  48

      A=B
0 1 0
0 1 0

      A>B
0 0 0
1 0 0
```

Scalar functions are applied element-by-element.

```
      +/[1]A
5  7  9

      +/[2]A
6  15

      ⌈/[2]B
7  8

      ×/[1]A
4  10  18

      ×/A
6  120
```

Reduction can be applied to arrays over the dimension specified in the brackets.

Note: If no dimension is specified, the last dimension is assumed. Hence, ×/A is equivalent to ×/[2]A in this example.

```
      ,B
4  2  7  1  5  8

      C←3 2⍴B
      C
4  2
7  1
5  8
```

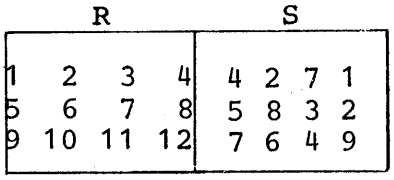
Shapes of arrays may be changed.

Arrays may be catenated.

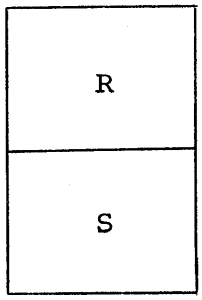
Enter R and S and display.

```
      ⎕←R←3 4⍴⍳12
 1  2  3  4
 5  6  7  8
 9 10 11 12

      ⎕←S←3 4⍴4 2 7 1 5 8 3 2 7 6 4 9
4 2 7 1
5 8 3 2
7 6 4 9
```

```
      R,S
 1  2  3  4  4  2  7  1
 5  6  7  8  5  8  3  2
 9 10 11 12  7  6  4  9
```

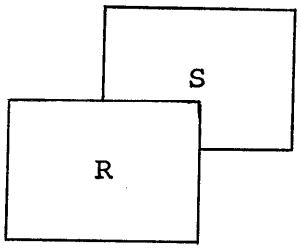



```
      R,[1]S
 1  2  3  4
 5  6  7  8
 9 10 11 12
 4  2  7  1
 5  8  3  2
 7  6  4  9
```

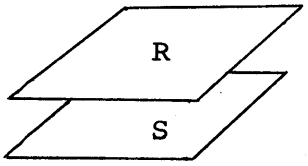


Arrays may be laminated.

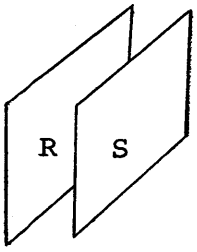
```
      R,[.5]S
 1    2    3    4
 5    6    7    8
 9   10   11   12

 4    2    7    1
 5    8    3    2
 7    6    4    9
```




```
      R,[1.5]S
 1    2    3    4
 4    2    7    1

 5    6    7    8
 5    8    3    2

 9   10   11   12
 7    6    4    9
```




```
      R,[2.5]S
 1    4
 2    2
 3    7
 4    1

 5    5
 6    8
 7    3
 8    2

 9    7
10    6
11    4
12    9
```



Note: The .5 in these three examples has no special significance; the same results would be obtained if the 5 were any digit from 1 to 9.

```
      N←1 2 3 4 5
      N∘.×N
 1  2  3  4  5
 2  4  6  8 10
 3  6  9 12 15
 4  8 12 16 20
 5 10 15 20 25
```

Multiplication tables may be produced using the outer product.

(Any of the other scalar operators that use two arguments can be used in place of the ×).

```
      A
1 2 3
4 5 6

      C
4 2
7 1
5 8

      A+.×C
33 28
81 61
```

The inner product can be used to find the familiar matrix product.

(Any of the other scalar operators that use two arguments can be used in place of the + and ×).

```
      1 0 1/A
1 3
4 6

      1 0 1 1\A
1 0 2 3
4 0 5 6

      1 0 1\[1]A
1 2 3
0 0 0
4 5 6


      1 0 1 1 1\X
A BCD
E FGH
I JKL
```

Arrays may be compressed or expanded.

```
      X[1;2]
B
      B[2;3]
8
      A[;1]
1  4
      A[1;]
1  2  3
```

Specific parts of arrays may be selected.

```
      ⌽X
DCBA
HGFE
LKJI
      1⌽X
BCDA
FGHE
JKLI
      2⌽[1]X
IJKL
ABCD
EFGH
```

Arrays may be reversed or rotated.

```
      ⍉A
1  4
2  5
3  6
```

Arrays may be transposed.

```
      M←2 2⍴1 2 3 ¯1
      V←8 3
      V⌹M
2  3
```

Simultaneous linear equations:

$$x+2y=8$$
$$3x-y=3$$

may be solved by using the matrix of coefficients and vector of constants.

EXECUTION MODE AND USER-DEFINED FUNCTIONS

The APL\360 System operates in either of two user-selected modes; execution mode where statements are executed when they are entered (Example 4A) and definition mode where statements are entered and stored in the active workspace as part of a user-defined function.

APL STATEMENT	MEANING
`2 × 3` `6`	2 times 3

`A←2`
`B←3`
`A*B`
`8`
	2 raised to the 3rd power

`25*.5` `5`	square root of 25
`10⍟100` `2`	log of 100 to the base 10
`+/3 5 2 4` `14`	the sum of 3, 5, 2, and 4

Example 4A. Sample APL statements in execution mode

Example 4B illustrates function definition. The defined function (or program, as it is sometimes called) starts with a ∇ and is given a name, GRADES in this case. The APL statements for solving the problem are written, and the function definition is ended by typing the ∇. The defined function is not executed until the user signals the system by typing the name of the function. Notice that the function can be used over and over again to solve for different groups of values.

Problem: Find the average of a group of grades.
Arrange the grades in descending sequence.

Function definition:

Enter		Meaning
	∇ *GRADES*	Open definition and name it
[1]	(+/*G*)÷ρ*G*	Sum of all values in G divided by number of values in G
[2]	*G*[⍒*G*]	Arrange G in descending order of G
[3]	∇	Close definition

Execute

G←100 80 90 70 90 *GRADES*	user enters
86 100 90 90 80 70	APL responds
G←17 30 40 *GRADES*	user enters
29 40 30 17	APL responds

Example 4B. Function definition and repeated execution

Example 4C is another example of a defined function. In order to describe the answers printed during execution, descriptive information is included in the function definition. This information is identified by quotation marks. Notice that when the function is executed, the text prints without the quotation marks. The computations done in Example 4C for Team 1 (computations for Team 2 are done similarly) are discussed below:

+/*T*1	Add all the scores for Team 1
*R*1←⌈/*T*1	⌈/*T*1 means find the maximum value (maximum number of runs in any one inning) for T1

```
T1 ⍳ R1
```

Which position in the scores for Team 1 (which inning) did the maximum number of runs occur?

Problem: BASEBALL

Find the final score and biggest inning for each team.

Team 1	0	1	0	2	0	3	2	5	0
Team 2	0	0	0	1	2	1	3	0	0

Function Definition

```
      ∇ BSEBL
[1]   T1←⍞
[2]   T2←⍞
[3]   'FINAL SCORE:    ';+/T1;' TO ';+/T2
[4]   'TEAM1:     RUNS IN BIGGEST INNING ';R1←⌈/T1
[5]   '                  INNING NUMBER ';T1⍳R1
[6]   'TEAM2:     RUNS IN BIGGEST INNING ';R2←⌈/T2
[7]   '                  INNING NUMBER ';T2⍳R2
[8]   →1
[9]   ∇
```

Note: Scores are entered at execution time.

```
      BSEBL
⍞:
      0 1 0 2 0 3 2 5 0
⍞:
      0 0 0 1 2 1 3 0 0
FINAL SCORE:    13 TO 7
TEAM1:     RUNS IN BIGGEST INNING 5
                  INNING NUMBER 8
TEAM2:     RUNS IN BIGGEST INNING 3
                  INNING NUMBER 7
⍞:

      →
```

Answers print with messages.

APL waits for next set of team scores.

Enter right arrow to end execution.

Example 4C. Identifying answers

Example 4D is another example of a function definition in which the manipulation of characters takes place, rather than computations with numbers. The example shows one method of entering variable length alphabetic information. In this case, the length of names entered is restricted to eight characters or fewer. After execution, the names can be arranged to print in a list.

Problem: Enter names and list them. (Note that in this example blanks must be added to make all the names the same length, eight characters.)

Method: Use the TAKE function (↑) to get blanks added automatically.

```
      ∇ NAMES                          Meaning
[1]   N←⍳0                             Make N an empty vector.
[2]   IN←8↑⍞                           Take 8 characters. Pad
                                       with blanks or truncate
[3]   N←N,IN                           if necessary.
                                       Add into string N.
[4]   →2                               Go to step 2 and repeat.
[5]   ∇

      NAMES
HOPKINS
CROCKER
FLOOD
STANFORD
⍬                                      Type O backspace U
                                       backspace T to end
      N                                execution.
HOPKINS CROCKER FLOOD   STANFORD        Display the string.

      4 8 ⍴N                           Rearrange to print 4
HOPKINS                                rows of 8 characters
CROCKER                                each.
FLOOD
STANFORD
```

Example 4D. Entering alphabetic information

One of the powers of APL is its concise notation. This is illustrated in example 4E, which shows how the solution in 4D can be entered in a more concise form without altering any logic:

```
      ∇ NAMES
[1]   N←⍳0
[2]   N←N,8↑⍞∇
[3]   →2∇
```

Example 4E. Entering alphabetic information

When an APL statement is executed, it may employ primitive functions (for example, + - × ÷) or defined functions to process data. When a defined function is thus referenced, each statement in that function is executed and may, in turn, employ primitive functions and any defined functions in the active workspace (Example 5).

Problem: Print average and test scores in descending order. Change scores to percent, average again, and print in descending order.

Function definition		Meaning
	`∇ PRCNT`	
[1]	`GRADES`	Use function GRADES (Example 4B) to compute average and rearrange. Change to percent: Divide score by total possible and multiply by 100. Use GRADES to compute average and rearrange.
[2]	`G←100×G÷T`	
[3]	`GRADES`	
[4]	`∇`	
	`G←40 30 50 35 45 38 42`	Enter scores, total possible, and PRCNT.
	`T←50`	
	`PRCNT`	
40		GRADES prints average and rearranges scores.
50 45 42 40 38 35 30		
80		GRADES prints percent average and rearranges percent scores.
100 90 84 80 76 70 60		

Example 5. An APL statement employing a defined function.

USING APL\360

ENTRIES FROM THE KEYBOARD

After a connection is established, normal communication between a terminal and the central computer is carried on by means of entries from the typewriter keyboard, which alternately locks and unlocks as each entry is made and the computer completes its work. The general procedure is to type an instruction or command, then strike the carrier return to indicate the end of the line. (On some terminals a transmission signal must also be sent.)

Errors in typing a line can be corrected before the carrier return is struck.

In addition to correcting errors as they occur, APL allows the user to display and to make corrections, changes, and additions to existing function definitions. Lines can be inserted into or added to the end of the function. Whole lines can be replaced, or specific characters within a line can be changed or inserted.

THE APL CHARACTER SET

The APL\360 keyboard is shown in Figure 1. The numerals, alphabetic characters, and the comma, colon, semi-colon, and period appear in the standard typewriter keyboard positions. Alphabetic characters are always entered when the keyboard is in the lower case position, i.e. not shifted, although they print as upper case italics. Upper case is used to enter most APL special characters.

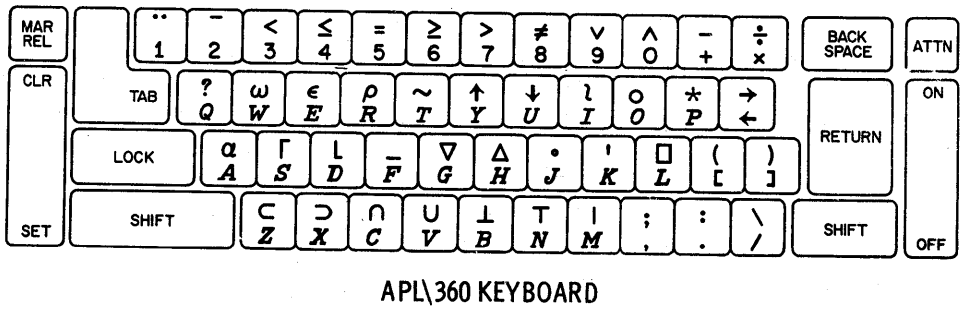


Figure 1. APL keyboard

The part numbers of APL printing elements are given in Part III. However, any printing element could be used with the APL system, since the encoded characters generated by the keyboard and transmitted to the computer are independent of the particular element mounted on the terminal. The transmitted information will be interpreted according to the APL keyboard characters.

The visual interpretation of APL expressions require an APL printing element. However, other printing elements are frequently useful in conjunction with special-purpose APL programs designed to exploit their character sets. Also, any element that matches the keyboard encoding (SELECTRIC ® or PTTC/BCD) of the terminal can be used for straightforward numerical work, since letters and digits print properly with such elements.

USER TERMINAL EQUIPMENT

The terminal used with the APL\360 System must be an IBM 2741 Communications Terminal, an IBM 2740-1 Communications Terminal equipped with the Transmit Control feature, or an IBM 1050 Data Communications System. It may connect to the central computer through the dial-up telephone network, by a leased telephone line, or by private wire.

A BRIEF DESCRIPTION OF THE APL FUNCTIONS

The following descriptions are included to illustrate the variety of functions available in APL. Detailed descriptions are provided in the APL\360 User's Manual.

STANDARD SCALAR FUNCTIONS

The following functions return a scalar result when their arguments are scalars. They may also be applied on an element-by-element basis to arrays of any rank, provided either that where the functions have two arguments, both arguments have the same rank and the same length in every dimension, or that at least one argument has only one element. Any of the scalar functions with two arguments may be used in reduction, or in the generalized inner and outer products.

X+*Y*	X plus Y
+*Y*	Y (no change)
X-*Y*	X minus Y

Note: Functions with two arguments are called dyadic functions. Those with only one argument are called monadic functions.

$-Y$	Minus Y
$X\times Y$	X times Y
$\times Y$	Signum of Y
$X\div Y$	X divided by Y
$\div Y$	Reciprocal of Y
$X*Y$	X to the Yth power
$*Y$	e to the Yth power
$X\lceil Y$	Maximum of X and Y
$\lceil Y$	Ceiling of Y (next integer $\geq$ Y)
$X\lfloor Y$	Minimum of X and Y
$\lfloor Y$	Floor of Y (next integer $\leq$ Y)
$X\mid Y$	X residue of Y
$\mid Y$	Absolute value of Y
$X\circledast Y$	Log of Y to the base X
$\circledast Y$	Natural log of Y
$X\circ Y$	Trigonometric functions and inverse trigonometric functions
$\circ Y$	PI times Y
$X!Y$	Number of combinations of Y things taken X at a time
$!Y$	Y factorial; Gamma of Y-1
$?Y$	Random equi-probable selection of an integer from 1 to Y
$X<Y$	X less than Y
$X\leq Y$	X less than or equal to Y
$X=Y$	X equals Y
$X\geq Y$	X greater than or equal to Y
$X>Y$	X greater than Y

$X \neq Y$ X not equal to Y

$X \wedge Y$ X and Y

$X \vee Y$ X or Y

$X \nor Y$ Neither X nor Y (X NOR Y)

$X \nand Y$ Not both X and Y (X NAND Y)

$\sim Y$ Not Y

GENERALIZED MATRIX OPERATIONS

In the entries below, the symbol ⍟ (not an APL symbol) has been used to indicate any standard scalar dyadic function.

$X +.\times Y$ Ordinary matrix product of X and Y

$X ⍟.⍟ Y$ Generalized inner product of X and Y

$X \circ.⍟ Y$ Generalized outer product of X and Y

GENERALIZED REDUCTION

$⍟/Y$ The reduction along the last dimension of Y

$⍟/[Z]Y$ The reduction along the Zth dimension of Y

$⍟⌿Y$ The reduction along the first dimension of Y

COMPRESSION AND EXPANSION

X/Y X (logical) compressing along the last dimension of Y

$X/[Z]Y$ X (logical) compressing along the Zth dimension of Y

$X⌿Y$ X (logical) compressing along the first dimension of Y

$X\backslash Y$ X (logical) expanding along the last dimension of Y

$X\backslash[Z]Y$ X (logical) expanding along the Zth dimension of Y

$X⍀Y$	X (logical) expanding along the first dimension of Y

OTHER FUNCTIONS

$X⍴Y$	Reshape Y to have dimension X
$⍴Y$	Dimension of Y
$X[Y]$	The elements of X at locations Y
$X⍳Y$	Locations of Y within vector X
$⍳Y$	The first Y consective integers (follow index origin)
$X∈Y$	Which elements of X are members of Y?
$X⊤Y$	Representation of Y in number system X
$X⊥Y$	Value of the representation Y in number system X
$X⌽Y$	Rotation by X along the last dimension of Y
$X⌽[Z]Y$	Rotation by X along the Zth dimension of Y
$X⊖Y$	Rotation by X along the first dimension of Y
$⌽Y$	Reversal along the last dimension of Y
$⌽[Z]Y$	Reversal along the Zth dimension of Y
$⊖Y$	Reversal along the first dimension of Y
$X⍉Y$	Transpose by X of the coordinates of Y
$⍉Y$	Ordinary transpose of Y (transposing last two coordinates only)
X,Y	Y catenated to X
$X,[Z]Y$	Y catenated to X on the Zth coordinate, or laminations of X and Y
$,Y$	Ravel of Y (make Y a vector)
$X↑Y$	Take the first X (or last -X) elements of Y
$X↓Y$	Drop the first X (or last -X) elements of Y

X←*Y* X specified by Y: the name X receives the value of Y

X?*Y* X integers taken without replacement from Y

⍋*X* Grade up of X

⍒*X* Grade down of X

⌹*A* Inversion of Matrix A

B⌹*A* Solution of simultaneous linear equations where A is the coefficient matrix and B is the vector of constants

SYMBOLS HAVING SPECIAL USES

The following symbols are not functions, but may be used in APL with the sense indicated below:

( ) Parentheses. Expression within them is to be evaluated before being used as the argument of an operator or defined function.

→*X* Branch to X. When X is a scalar, branch to X; when X is a vector, branch to the first element of X; when X is an empty vector, go to the next line in sequence (do not branch); when X is 0 (or any invalid line number) exit the function.

⎕←*X* Print the value of X.

X←⎕ Request input. Value of X is the resulting value after expression entered is evaluated.

X←⍞ Request input. Value of X is entire input text as literal characters, up to but not including carrier return.

'*XYZ*' The literal characters XYZ.

⍝ Comment. Precedes an unexecuted line of comments.

⍀ Illegal character having the special property of halting a request for literal input. Formed by typing *O* backspace *U* backspace *T*, carriage return.

∇ Used to open and close function definition.

⍫ Used to lock a function definition.

[□] Used to print function when in definition mode.

A BRIEF DESCRIPTION OF APL\360 SYSTEM COMMANDS

CLASSIFICATION OF COMMANDS

System commands and APL operations are distinguished functionally by the fact that system commands can be called for only by individual entries from the keyboard, and cannot be executed dynamically as part of a defined function. They are distinguished in form by the requirement that system commands be prefixed by a right parenthesis.

It may be desirable to exert some system control dynamically, and some items of system information can be profitably used during the execution of a program. For these purposes, APL\360 provides functions which can be used like other APL operations. These functions are described at the end of this section.

All system commands can be executed when the terminal is in the execution mode, in which APL operations are executed immediately upon entry. However, in definition mode, in which sequences of operations are being composed into functions for later execution, commands which call for storing a copy of the workspace, or which might otherwise interfere with the definition process itself, are not permitted.

System commands are conveniently grouped into five classes with regard to their effect upon the state of the system:

- Terminal control commands affect the relation of a terminal to the system.
- Workspace control commands affect the state of the active workspace.
- Library control commands affect the state of the libraries.
- Inquiry commands provide information without affecting the state of the system.
- Communication commands affect the transmission of messages among the terminals.

Notes on the following commands:

- Items in brackets are optional.

- key or lock: a password.
- wsname: library number and workspace name, or workspace name alone, as required.

Terminal Control

)number [:key]	Sign on designated user and start a work session.
)*OFF* [:lock]	End work session.
)*OFF HOLD* [:lock]	End work session and hold dial-up connection.
)*CONTINUE* [:lock]	End work session and store active workspaces.
)*CONTINUE HOLD* [:lock]	End work session, store active workspace, and hold dial-up connection.

Workspace Control

)*CLEAR*	Activate a clear workspace.
)*LOAD* wsname [:key]	Activate a copy of a stored workspace.
)*COPY* wsname [:key] name	Copy a global object from a stored workspace.
)*COPY* wsname [:key]	Copy all global objects from a stored workspace.
)*PCOPY* wsname [:key] name	Copy a global object from a stored workspace, protecting active workspace.
)*PCOPY* wsname [:key]	Copy all global objects from a stored workspace, protecting active workspace.
)*GROUP* names	Gather objects into a group.
)*ERASE* names	Erase global objects.
)*ORIGIN* 0 or 1	Set index origin for array operations.

)*DIGITS* 1 to 16	Set maximum for significant digits in output.
)*WIDTH* 30 to 130	Set maximum for an output line.
)*WSID* wsname	Change active workspace identification.

Library Control

)*SAVE*	Save a copy of active workspace using its current name.
)*SAVE* wsname [:lock]	Save a copy of the active workspace assigning a new name.
)*DROP* wsname	Erase a stored workspace.

Inquiry

)*FNS* [letter]	List names of defined functions.
)*VARS* [letter]	List names of global variables.
)*GRPS* [letter]	List names of groups.
)*GRP* name	List membership of designated group.
)*SI*	List halted functions (state indicator).
)*SIV*	List halted functions and associated local variables (augmented state indicator).
)*WSID*	Give identification of active workspace
)*LIB* [number]	List names of workspaces in the specified library.
)*PORTS*	List ports in use and codes of connected users.
)*PORTS* code	List port numbers associated with designated user code.

Communications

)*MSGN* port [text]	Address text to designated port. No reply expected.
)*MSG* port [text]	Address text to designated port and lock sender's keyboard. Reply expected.
)*OPRN* [text]	Address text to recording terminal (APL Operator). No reply expected.
)*OPR* [text]	Address text to recording terminal (APL Operator), and lock sender's keyboard. Reply expected.

SYSTEM AND PROGRAM INFORMATION

APL\360 permits access to several items of information concerning the status of programs in the user's workspace and the status of the APL system. These are called I-beam functions; the I-beam symbol ⌶ is formed by overstriking ⊥ and ⊤ . The following I-beam functions are currently available to users:

⌶19 Cumulative keyboard-unlocked time since sign-on in 60ths of a second

⌶20 Time of day in 60ths of a second since 00:00 on date given by ⌶25

⌶21 Compute time since sign-on in 60ths of a second

⌶22 The amount, in bytes, of unused space remaining in user's workspace

⌶23 The number of users currently signed on

⌶24 Connect time since sign-on in 60ths of a second

⌶25 Date on which the system was initiated

⌶26 The number of the line of the function currently being executed

⌶27 A vector containing the line numbers of all functions whose execution has been started but not yet completed, with the most recent function first

I28 A code indicating the terminal device being used

I29 User sign-on number

PART II: OPERATING APL\360

INTRODUCTION

Special functions and privileged commands are provided for use by the operations staff to control system resources. Through these commands, the system operator can, for example, install or remove users, broadcast messages to all users, determine user status, and delete a public library from the system. Operator dialog with the system is achieved through the means of a recording terminal. These and other support features are discussed below.

THE RECORDING TERMINAL

As connections with remote user terminals are established or broken, and users commence or terminate work sessions, a printed log of these events is generated at a system device called the recording terminal. The recording terminal, which is usually but not necessarily located at the central computer site, is ordinarily attended by an APL\360 System operator who monitors the operation of the system, and provides a common point of contact for all users.

There are certain supervisory functions, essential to the operation of APL\360, which can be effected from the recording terminal. Thus, this terminal holds a privileged position relative to user terminals. The enrollment of new users to an installation and the allocation of library space are two examples of this kind of function.

A terminal signed on with a special operator number automatically becomes a recording terminal. It behaves in the same way as a user terminal with the following exceptions:

- Its keyboard is normally locked. The system operator must first signal attention before a keyboard entry can by typed. When the keyboard is normally locked, output interrupted by an attention signal is retransmitted.
- It receives messages such as sign-ons and sign-offs generated by APL users.
- It is always privileged and can perform system management functions not generally available to terminal users.

- Other users can sign on only if the recording terminal is signed on and its keyboard is locked.

PRIVILEGED SYSTEM COMMANDS

The following privileged system commands are supplied for use by the system operator:

)ADD	Enroll a new terminal user, create a public library, or change a previously enrolled user's name, account number lock, workspace quota, or CPU time limit.
)DELETE	Remove a public library or user account number from the system and drop all workspaces in the corresponding library.
)HI	Prepare a message to be delivered to each terminal user as he signs on.
)PA	Send a public address message to all terminal users currently signed on.
)HIPA	Prepare combined sign on and public address message.
)LOCK	Prevent a user from signing on.
)UNLOCK	Reinstate a previously locked-out user.

OPERATOR'S WORKSPACE

In addition to the full capabilities of APL, the system operator is supplied with a workspace (called OPFNS) which contains a collection of APL\360 functions for controlling the online system. In general, operator functions will often be invoked in combination with APL expressions and with each other. Table 1 summarizes the special operator functions.

Table 1. Summary of APL\360 Special Operator Functions

Function	Description
'XX' AH 'XX'	Hexadecimal addition *
ALL	Returns all ports in system
VP1 AND VP2	Returns intersection of ports
BOUNCE VP	Bounce ports VP
DEFUSE VP	Defuse ports VP
DEFUSED	Returns defused ports
DEPRIVILEGE VP	Deprivilege users at ports VP
DISPLAY 'XX'	Display 4 bytes starting at XX *
'YY' DISPLV 'XX'	Display YY bytes starting at XX *
DOWN	Returns disabled ports
DTH N	Decimal to hex conversion *
VP1 EXCEPT VP2	Returns combination of ports
EXPRESS	Returns express ports
FREE	Reports free phone numbers
HTD 'XX'	Hex to decimal conversion *
INITIALIZE	Initialize OPFNS workspace *
KBLOCK VP	Keyboard lock ports VP
KBUNLK VP	Keyboard unlock ports VP
LIMIT VP	Make ports VP express
MAP	Display control section (CSECT) map.
MPXAD 'XX'	Returns port number of MPX address XX
NOT VP	Returns ports not specified
OFF	Returns ports signed off
ON	Returns ports signed on
VP1 OR VP2	Returns union of ports
'XX' PATCH 'YY'	Patches location XX to value YY * **
PAWAIT	Returns ports which have not received PA
PHONE VI	Returns port numbers of phones VI
PORT VP	Reports brief port information
PORTS VP	Reports detailed port information
PRIORITY A,B	Adjust multiprogramming priority
PRIVILEGE VP	Privilege users at ports VP
PRIVILEGED	Returns privileged ports
SETLIMIT N	Sets express time limit
'XX' SH 'XX'	Hexadecimal subtraction *
SHOWHI	Shows HI message
SHOWLIMIT	Shows express time limit
SHOWPA	Shows last PA message
SHUTDOWN	Initiate system shutdown
SUSPECT	Returns idle ports
UNLIMIT VP	De-express express ports
USER VI	Returns ports for user account numbers
USER 'ABC'	Returns ports for designated user code
'XX' VERIFY 'YY'	Verify contents of location XX *

*This function not required for normal operation.

**This function can cause system failure if misused.

APL\360 OFFLINE UTILITY PROGRAM

APL\360 provides the installation with a comprehensive utility program for the maintenance of user libraries. The utility is run as an ordinary batch job. Examples of typical utility operations include the following:

- Formatting of disk packs
- Backup of user library on tape or disk
- Verification of disk and tape readability
- Retrieval of individual workspaces or libraries from backup copies
- Reallocation of APL\360 library to conform with changed disk extent boundaries
- Write selected workspaces onto tape for transmission to another APL\360 installation

Provision is also made for the incorporation of installation-written billing routines. These routines may be added to the utility program, if billing of users is required, either during system generation time or at some later time. Table 2 summarizes all utility operations.

Table 2. Summary of APL\360 Utility Operations

Operation Name	Parameter	Optional Listing	May Run with APL	Disk or Tape I/O	Purpose and Remarks
ACCTG	0 1	ws* names	Yes	Disk	Lists users (0) and, optionally, ws names (1).
BILLING	None	None	No	Disk	Produces billing information using installation-defined routines.
CREATE	None	ws names written on disk	No	Both	Special-purpose form of RESTORE for SYSGEN.
DISKFMT	Library extent number	None	No	Disk	Writes full-track records on a library extent.
DUMP	Tape record length	ws names written on tape	No	Both	Writes all directories and workspaces to tape.
INCDUMP	Tape record length	ws names written on tape	No	Both	Writes all directories and recently saved ws to tape.
RESTORE	None	ws names written on disk	No	Both	Writes directories and workspaces from tape to library disk.
RETRIEVE	None	ws names written on disk	No	Both	Searches dump tape for selected ws and adds them to library.
SELDUMP	Tape record length	ws names written on tape	Yes	Both	Writes selected workspaces to tape.
SELREST	None	ws names written on disk	No	Both	Adds all workspaces on tape to library.
TESTBILL	None	None	Yes	Disk	Like BILLING, but no accounting reset.
TVERIFY	None	ws names on tape	Yes	Tape	Readback check of dump tape.
VERIFY	Library extent number	None	Yes	Disk	Readback check of library extent.
WSDUMP	None	None	Yes	None	Prints SNAP dumps of damaged WS on WSDUMP data set.
WSLIST	None	None	Yes	None	Makes other operations list ws names.

* Workspace(s)

PART III: INSTALLATION PLANNING

INTRODUCTION

This section is intended to aid the systems manager, systems analyst, systems programmer, and other responsible personnel in planning for the installation and implementation of the APL\360 System. The following aspects are discussed:

- System configuration
- Terminal features
- Storage estimates
- Priorities in a multiprogramming environment
- Performance

The user is advised that with APL\360, as with other systems, attention should be given to preinstallation systems analysis and design, and communications requirements.

SYSTEM CONFIGURATION

APL\360 operates as a problem program under the control of either the IBM System/360 Operating System (OS/360) Multiprogramming with a Fixed Number of Tasks (MFT with subtasking), or Multiprogramming with a Variable Number of Tasks (MVT); or the IBM System/360 Disk Operating System (DOS/360).

Both APL\360-OS (5734-XM6) and APL\360-DOS (5736-XM6) are functionally equivalent except for the internal interface with the host system and for the specific operating procedure at the central computer location.

The selection of hardware configuration for either version of APL\360 is largely dependent upon the total system requirements of a user installation--that is, by the requirements of OS/360 or DOS/360, APL\360, and the application demands of the specific in-house installation. Minimum requirements for both APL\360-OS and APL\360-DOS are described below, and are summarized in Table 3.

MINIMUM SYSTEM REQUIREMENTS--APL\360-OS

APL\360-OS is structured to support the following minimum devices for online operation:

- One IBM System/360, at least Model 2050HG for OS/360 (MVT) or Model 2040H for OS/360 (MFT with subtasking), with a minimum partition or region size of 170K bytes (based upon maintaining two 36,000-byte workspaces core resident). By functionally restricting the APL\360-OS System, the OS/360 (MFT/MVT) partition or region may be reduced to approximately 128K bytes. This subject is described more fully later in this section.

- One Selector Channel

- One IBM 2311 Disk Storage Drive or one disk storage module on an IBM 2314 Direct Access Storage Facility

- One IBM 2701 Data Adapter Unit, or one 2702 or 2703 Transmission Control Unit (2702 or 2703 is recommended since 2701 does not allow interruption of terminal output)

- One IBM 2400-series nine-track Magnetic Tape Unit

- One IBM 2740-1 with receive-interrupt RPQ or one IBM 2741 Communication Terminal, or one IBM 1050 Data Communications System containing at least one IBM 1052 Printer-Keyboard for use as a recording terminal

- Universal Instruction Set

- Storage Protection

- Interval Timer

MINIMUM SYSTEM REQUIREMENTS--APL\360-DOS

APL\360-DOS is structured to support the same minimum devices for online operation as those listed for APL\360-OS with the exception of the CPU. For APL\360-DOS, the minimum processor is the IBM System/360, Model 2040GF with a minimum partition size of 170K bytes (based upon maintaining two 36,000-byte workspaces core resident).

By functionally restricting the APL\360-DOS System, the DOS/360 partition may be reduced to approximately 128K bytes.

A standalone variation of the APL\360-DOS System with a core storage requirement approximately equal to the DOS/360 partition size requirement may be generated from the distributed system. All other requirements remain as stipulated above.

A variation of the standalone version of the APL\360 System may be generated from the distributed system, which allows for minor variations between a real System/360 and the virtual System/360 provided by Control Program-67 (CP-67). The requirements of the virtual machine provided by CP-67 are identical to those required for the standalone version of APL\360 plus 480 bytes per port.

Table 3. Summary of APL\360 Minimum System Requirements

SYSTEM TYPE	APL\360-DOS		APL\360-OS	
	STANDALONE	DOS	MFT	MVT
CPU*	MODEL 2040GF		MODEL 2040H	MODEL 2050HG
PARTITION/ REGION SIZE**	170K BYTES			
CPU FEATURES	INTERVAL TIMER UNIVERSAL INSTRUCTION SET ONE SELECTOR CHANNEL			
DASD	ONE 2311 OR ONE 2314 STORAGE MODULE			
TAPE	ONE 2400-SERIES 9-TRACK			
TRANSMISSION CONTROL UNIT	ONE 2701, 2702, OR 2703			

*Minimum based on estimated APL\360 and host system storage requirements.

**Based on maintaining two 36,000-byte workspaces core resident. See also "Storage Estimates".

TERMINALS

APL\360 provides terminal handling programs for the following terminals:

- IBM 2741 Communications Terminal
- IBM 2740 Communications Terminal, Model 1
- IBM 1050 Data Communications System

Tables 4, 5, and 6 supply the required, recommended, and optional features for these terminals.

Table 4. IBM 2741 Communications Terminal Features

FEATURE	NUMBER	PREREQUISITES	REMARKS
Receive Interrupt**	4708		Recommended
Typamatic Keys	8341		Recommended
Modems:		Data Set:	Required
Dial Up***	3255	WE 103A2*	
Data Set Attachment	9114	WE 103A2*	
Data Set Attachment	9115	WE 103F2*	
IBM Line Adapter	46nn	Fixed Connection	
Keys, APL Keyboard****	RPQ M40174		Recommended
PTTC/BCD			P/N 5195500
Standard SELECTRIC			P/N 5156576
APL Printing Element:			Recommended
PTTC/BCD	RPQ F24235		P/N 1167988
Standard SELECTRIC	RPQ E62267		P/N 1167987
Character Spacing, 10/in.	9104		Recommended
Line Feeding, 6/in.	9435		Recommended
Pin Feed Platen	9509		Optional
Voltage	988n		Required
Transmit Interrupt	7900		Ignored
Print Inhibit	5501		Ignored

*Or equivalent.
**Use of this feature requires Feature No. 8055 (2741 break feature) or Feature No. 8200, Type 1 Terminal Interrupt Feature, on the 2702 or 2703. Required companion feature not available on 2701.
Feature required if used as the recording terminal.
***Required only if used on switched network (dial-up).
****Existing keyboards can be modified by pasting APL characters on the keytops (Order APL Characters, GX20-1783).

Table 5. IBM 2740 Communications Terminal, Model 1, Features

FEATURE	NUMBER	PREREQUISITES	REMARKS
Automatic EOB	1313	#6114	Recommended
Record Checking	6114		Recommended
Transmit Control	8028	#3255	Optional
Receive Interrupt**	RPQ F17913		Recommended
Modems:			Required
Dial Up***	3255	Data Set WE 103A2*	
Data Set Attachment	9114	Data Set WE 103A2*	
Data Set Attachment	9115	Data Set WE 103F2*	
IBM Line Adapter	46nn	Fixed Connection	
Keys, APL Keyboard****	RPQ M40174		Recommended
PTTC/BCD			P/N 5195500
Standard SELECTRIC			P/N 5156576
APL Printing Element:			Recommended
PTTC/BCD	RPQ F24235		P/N 1167988
Standard SELECTRIC	RPQ E62267		P/N 1167987
Character Spacing, 10/in.	9104		Recommended
Line Feeding, 6/in.	9435		Recommended
Pin Feed Platen	9509		Optional
Voltage	988n		Required
2260 Attachment	8301		Prohibited

*Or equivalent.
**Required companion feature not available on 2701. Feature required if used as the recording terminal.
***Required only if used on switched network (dial-up).
****Existing keyboards can be modified by pasting the APL characters on the keytops (Order APL Characters, GX20-1783).

Table 6. IBM 1050 Data Communications System Features

FEATURE	NUMBER	PREREQUISITES	REMARKS
1051 Control Unit	Mod 1 or 2		Required
Automatic EOB	RPQ E28235		Recommended
Receive Interrrupt**	6100		Recommended
Text Time-Out Suppress.	9698		Recommended
Automatic Ribbon Shift	1295		Recommended
Modem:			Required
Data Set Attachment	9114	Data Set WE 103A2*	
Data Set Attachment	9115	Data Set WE 103F2*	
IBM Line Adapter	46nn	Fixed Connection	
First Printer Attach.	4408		Required
Second Printer Attach.	6381		Optional
Card Punch Attachment	1635		Optional
First Punch Attachment	4410		Optional
Second Punch Attach.	6383		Optional
First Reader Attach.	4411		Optional
Second Reader Attach.	6384		Optional
Keyboard Request	4770		Optional
Line Correction	4795		Optional
Line Correc. Release	4796		Optional
Blower	9030		Optional
Forms Stand Stacker	4450		Optional
I/O Component Table	4632		Optional
Typing Table	9705	#904n	Recommended
Voltage	988n		Required
Automatic EOB	1313		Prohibited
1052 Printer-Keyboard ***	Mod 1 or 2		Required
Acceler. Carrier Ret.	1006		Recommended
APL Printing Element:			Recommended
PTTC/BCD	RPQ F24235		P/N 1167988
Char. Spacing, 10/in.	9104		Recommended
Line Feeding, 6/in.	9435		Recommended
Cable Exit	902n		Required
Pin Feed Platen	9509		Optional
1053 Printer	Model 1	#6381 on 1051	Optional
Acceler. Carrier Ret.	1006		Recommended
APL Printing Element:			Recommended
PTTC/BCD	RPQ F24235		P/N 1167988
Char. Spacing, 10/in.	9104		Recommended
Line Feeding, 6/in.	9435		Recommended
Pin Feed Platen	9509		Optional
1054 Paper Tape Reader	Model 1	#4411/6384 on 1051	Optional
1055 Paper Tape Punch	Model 1	#4410/6383 on 1051	Optional
1056 Card Reader	Model 1	#4411/6384 on 1051	Optional
Extended Char. Reading	3861		Required
1057 Card Punch	Model 1	#4410/6383 on 1051	Optional
Extended Char. Punch	3860	#1635 on 1051	Required
Operator Panel	5478		Recommended
Voltage	988n		Required

* Or equivalent
** Required companion feature not available on 2701. Feature required if used as a recording terminal.
***Existing keyboards can be modified by pasting APL characters on the keytops (Order APL Characters, GX20-1783).

STORAGE ESTIMATES

The amount of core and disk storage required by APL\360 depends upon several installation-related options and/or the configuration of the specific machine upon which APL\360 is to be run. Three primary storage requirements should be estimated in planning any APL\360 installation:

- The main core storage (APLCORE) required for the APL\360 supervisor, interpreter, active workspaces in core, terminal buffers, and control information
- The size in cylinders (SWAPEXT) of the contiguous single-extent data set used to contain swapped-out workspaces
- The minimum size in cylinders (LIBEXT) of the first or only contiguous single-extent data set used for containing library directories and workspaces. All library extents must be on the same device type, although the Library and Swap devices may differ.

PARAMETERS

The following parameters are used in estimating APL\360 storage requirements:

- (SDISK) The number of tracks per cylinder on the direct access device (10 for the IBM 2311, 20 for the IBM 2314) used for the Swap Extent.
- (LDISK) The number of tracks per cylinder on the direct access device used for the Library Extent.
- (PORTS) The number of transmission control unit ports, including gaps within a control unit, used for APL\360.
- (WSSIZE) The number of bytes per workspace in the system. The standard size workspace is 36,000 bytes, but this may be specified by the installation to be as small as 20,480 or as large as 21,600 times the smaller of SDISK or LDISK.
- (INCORE) The number of areas of core storage reserved for active workspaces when APL\360 is running. INCORE must be at least 2. Experience gained under APL\360-DOS indicates that on a System/360, Model 50, the performance improvement achieved by increasing INCORE from 2 to 3 is not particularly significant until the number of ports on

the system exceeds 20. When the number of ports passes 50, a fourth INCORE area is recommended.

- (DIRS) The number of user directories in the APL\360 library. These directories contain the account number, name, billing information, and pointers to the saved workspaces of all enrolled users. The size of a directory is identical to that of a workspace. Directory capacity is directly proportional to the specified workspace size. With standard size workspaces (36,000 bytes), each directory has capacity for about 150 accounts.

ESTIMATING CORE

The core storage estimate is the sum of:

- 88,000 bytes for the APL\360 supervisor, interpreter, and control information
- 336 bytes for each port
- The number of workspaces in core times 8 plus the workspaces size rounded up to a 2K boundary as follows:

```
APLCORE←88000+(336×PORTS)+INCORE×8+2048×⌈WSSIZE÷2048
```

ESTIMATING SWAP EXTENT

The Swap Extent must contain enough space to handle one workspace for every APL port on the system plus room for three extra workspaces. The size of the Swap Extent is determined by first calculating the number of full tracks required to contain a workspace by dividing the number of bytes in a workspace by the number of bytes per track and rounding up to the next whole track:

```
STRK←⌈WSSIZE÷362.4×SDISK
```

Then, if the number of tracks (STRK) is not less than one cylinder (SDISK), the Swap Extent is the number of whole cylinder per workspace, times 3 plus the number of ports:

```
SWAPEXT←(PORTS+3)×⌈STRK÷SDISK
```

Otherwise, the Swap Extent is 3 plus the number of ports divided by the number of whole workspaces per cylinder rounded to the next whole cylinder:

```
SWAPEXT←⌈(PORTS+3)÷⌊SDISK÷STRK
```

It is advisable to provide one or two extra workspace areas in the Swap Extent since a flagged bad track in a workspace swap area causes the entire set of tracks allocated to that workspace to be abandoned. This allowance can be made in the above formulas by increasing the 3 to a 4 or 5.

ESTIMATING MINIMUM LIBRARY DISK STORAGE

The minimum size library extent 0 provides disk storage for primary and alternate copies of the directories. Additional disk storage must, however, be provided for workspaces saved in the library. This extra space may be supplied by increasing library extent 0 and/or providing additional library extents. The amount of disk storage needed to save a single workspace varies from one 2314 disk track or two 2311 disk tracks up to the maximum number of tracks required to contain a full workspace. This variability results from the fact that only the meaningful contents of a workspace are stored on disk when it is saved in a user's library.

An average of approximately 400 36,000-byte workspaces can be contained on a 1316 disk pack and about 1600 on a 2316 disk pack.

The minimum size of library extent 0 (the first or only library extent) is determined by first computing the number of whole tracks required to contain a workspace:

```
LTRK←⌈WSSIZE÷362.4×LDISK
```

Then, the minimum size of library extent 0 is 2 times the number of directories times the number of tracks per workspace, plus one additional track per directory if a workspace occupies a number of tracks that is an integral number of cylinders:

```
LIBEXT←⌈(DIRS×(2×LTRK)+0=LDISK|LTRK)÷LDISK
```

FUNCTIONALLY RESTRICING APL\360 TO REQUIRE LESS THAN THE MINIMUM SPECIFIED PARTITION/REGION SIZE

Several avenues of approach may be used by the installation systems programmer to reduce the APL\360 core storage requirements:

- Reduce workspace size from 36,000 to a minimum of 20,480 bytes. (This will limit capacity of workspaces and directories.)

- Remove tables from the APL supervisor used to accumulate operational statistics.

- Reduce the IODEBUG parameter to 1 for APL system generation.

- Reduce the number of APL ports.

Some combination of the above may be used to reduce the partition or region size to approximately 128K.

Caution: Library workspaces provided as part of APL\360-DOS and APL\360-OS may not work on a functionally restricted system.

PRIORITIES IN A MULTIPROGRAMMING ENVIRONMENT

When APL\360 is operating in a multiprogramming environment under OS/360 or DOS/360, it ensures that other partitions or regions receive frequent CPU service by alternating its own priority between high and low. Thus, when APL\360 has low priority, the other partitions or regions will receive CPU time if they are not quiescent.

The nominal proportion of time during which APL\360 has high priority is controlled by two system parameters, preset in the distributed system to fifty percent, but modifiable at any time by the APL\360 System operator. The two parameters specify the proportion of APL\360 high-priority time when no ports are in use, and when all ports are in use. This proportion varies approximately linearly with the number of ports in use.

Unless the APL\360 System and some other partition or region require heavy CPU service, the priority setting will have little or no effect on overall system behavior.

PERFORMANCE

When operating in a multiprogramming environment, performance is primarily controlled by the user as specified under "Priorities in a Multiprogramming Environment".

The apparent performance of APL\360 at a terminal is primarily dependent upon:

- The speed of the CPU, core storage, and DASD used

- The priority setting specified above

- The number of workspaces core resident (installation option)
- The number of active terminals
- The specific work being done by the user

The apparent performance is normally measured in terms of reaction time, that is, the length of time for the system to respond to a trivial request (carrier return to start of printing).

When APL\360 is operating, the APL supervisor collects statistics that are available to "privileged" terminal users. These statistics may be utilized by the user to determine the performance of the APL\360 System in any operational environment.

APPENDIX A: SAMPLE TERMINAL SESSION

```
)1776
010)  19.32.36  07/03/68  JANET

   A P L \ 3 6 0
```

FUNDAMENTALS

```
      3×4
12
      X←3×4

      X
12
      Y←¯5

      X+Y
7
      144E¯2
1.44
      P←1 2 3 4
      P×P
1  4  9  16
      P×Y
¯5  ¯10  ¯15  ¯20
      Q←'CATS'
      Q
CATS
      YZ←5
      YZ1←5
      YZ+YZ1
10
      3+4×5+6
         v
         +5+6
18
      X←3
      Y←4
      (X×Y)+4
16
      X×Y+4
24
```

Entry automatically indented.
Response not indented.
X is assigned value of the expression.

Value of X typed out.
Negative sign for negative constants.

Exponential form of constant.

Four-element vector.
Functions apply element-by-element.
Scalar applies to all elements.

Character constant (4-element vector).

Multi-character names.

Correction by backspace and attention.

Executed from right to left.

```
      X Y
SYNTAX ERROR
      X Y
      ^
      XY
VALUE ERROR
      XY
      ^
```

Entry of invalid expression.
Shows type of error committed.
Retypes invalid statement with caret where execution stopped.
Multi-character name (not X times Y).
XY had not been assigned a value.

SCALAR FUNCTIONS

```
      4×3⌈5.1
20.4
      (4×3)⌈5.1
12
      4×⌈5.1
24
      X←⍳5
      X
1  2  3  4  5
      ⍳0

      Y←5-X
      Y
4  3  2  1  0
      X⌈Y
4  3  3  4  5
      X≤Y
1  1  0  0  0
      ○1
3.141592654
      ○÷1 2
3.141592654  1.570796327
      X←45 90
      ○X÷180
0.7853981634  1.570796327
      1○1
0.8414709848
      2○1 2
0.5403023059  ¯0.4161468365
      3○1
1.557407725
      ¯3○1
0.7853981634
      3○¯3○⍳7
1  2  3  4  5  6  7
      Y←1 2
      4○Y
1.414213562  2.236067977
      0○÷Y
0  0.8660254038
```

Dyadic maximum(two arguments).

Monadic ceiling(one argument).

Index generator function.

Empty vector prints as a blank line.
All scalar functions extend to vectors.

Relations produce logical (0 or 1) results.
Pi times 1.

Pi divided by 1 2

Conversion of X to radians.

Sin 1

Cos 1 2

Tan 1

Arctan 1

Tan Arctan 1 2 3 4 5 6 7

(1+Y*2)*.5

(1-reciprocal of Y*2)*.5

```
      7○1 2                          Tanh 1 2
0.761594156  0.9640275801
      ¯7○7○1 2                       Arctanh Tanh 1 2
1  2
```

DEFINED FUNCTIONS

```
      ∇Z←X F Y                       Header (2 args and result).
[1]   Z←((X*2)+Y*2)*.5               Function body.
[2]   ∇                              Close of definition.
      3 F 4                          Execution of dyadic function F.
5
      P←7
      Q←(P+1)F P-1                   Use of F with expressions as
      Q                                arguments.
10
      4×3 F 4
20
      ∇B←G A                         G is the signum function.
[1]   B←(A>0)-A<0                    A and B are local variables.
[2]   ∇
      G 4
1
      G ¯6
¯1
      X←¯6
      G X
¯1
      ∇H A                           Like G but has no explicit re-
[1]   P←(A>0)-A<0                      sult.  P is a global variable.
[2]   ∇
      H¯6
      P
¯1
      Y←H¯6                          H has no explicit result and
VALUE ERROR                            hence produces a value error
      Y←H ¯6                           when used to right of assign-
        ∧                              ment.
      ∇Z←FAC N;I                     FAC is the factorial function.
[1]   Z←1
[2]   I←0
[3]   L1:I←I+1                       L1 becomes 3 at close of def.
[4]   →0÷⍳I>N                        Branch to 0 (out) or to next.
[5]   Z←Z×I
[6]   →L1                            Branch to L1 (that is, 3).
[7]   ∇
      FAC 3
6
      FAC 5
120
```

```
      T∆FAC←3 5
      X←FAC 3
FAC[3] 1
FAC[5] 1
FAC[3] 2
FAC[5] 2
FAC[3] 3
FAC[5] 6
FAC[3] 4
      T∆FAC←0
```

Set trace on lines 3 and 5 of FAC.
Trace of FAC.

Reset trace control.

MECHANICS OF FUNCTION DEFINITION

```
      ∇G←M GCD N
[1]   G←N
[2]   M←M|N
[3]   →4×M≠0
[4]   [1]G←M
[2]   [4]N←G
[5]   [1□]
[1]   G←M
[1]   [□]
    ∇ G←M GCD N
[1]   G←M
[2]   M←M|N
[3]   →4×M≠0
[4]   N←G
    ∇
[5]   →1
[6]   ∇
      36 GCD 44
4
      ∇GCD
[6]   [4.1]M,N
[4.2] [□]
    ∇ G←M GCD N
[1]   G←M
[2]   M←M|N
[3]   →4×M≠0
[4]   N←G
[4.1] M,N
[5]   →1
    ∇
[6]   ∇
      36 GCD 44
8  36
4  8
4
```

Greatest common divisor function based on the Euclidean algorithm.

Correction of line 1.
Resume with line 4.
Display line 1.

Display entire GCD function.

Close of display, not definition.
Enter line 5.
Close of definition.
Use of GCD.
4 is GCD of 36 and 44.
Reopen def. (Use ∇ and name only).
Insert between 4 and 5.
Display entire function.

Fraction stays until close of definition.
End of display.
Close of definition.

Iterations printed by line 5 (was line 4.1).
Final result.

```
      ∇GCD[⎕]∇                    Reopen, display, and close GCD.
    ∇ G←M GCD N
[1]   G←M
[2]   M←M|N
[3]   →4×M≠0
[4]   N←G
[5]   M,N                         Line numbers have been reassign-
[6]   →1                            ed as integers.
    ∇                             Close (Even number of ∇'s in all).
      ∇GCD                        Reopen definition of GCD.
[7]   [5]                         Delete line 5 by using attention.
          ∨


[6]   ∇                           Close definition.
      ∇Z←ABC X                    A function to show line editing.
[1]   Z←(33×Q+(R×5)-6             A line to be corrected.
[2]   [1⎕9]                       Initiate edit of line 1.
[1]   Z←(33×Q+(R×5)-6             Types line, stops ball under 9.
          /   1 /1                Slash deletes, digit inserts.
[1]   Z←(3×Q)+(T×5)-6             Ball stops at first new space.
[2]   ∇                             Then enter the ) and the T.
      FAC 5                       FAC still defined.
120
      )ERASE FAC                  Erase function FAC.
      FAC 5                       Function FAC no longer exists.
SYNTAX ERROR
      FAC 5
      ∧
      ∇Z←BIN N                    An (erroneous) function for
[1]   LA:Z←(Z,0)+0,Z                binomial coefficients.
[2]   →LA×N≥ρZ∇
      BIN 3
VALUE ERROR
BIN[1]LA:Z←(Z,0)+0,Z              Suspended execution.
                  ∧
      Z←1                         Assign value to Z.
      →1                          Resume execution.
1  3  3  1                       Binomial coefficients of order 3.
      BIN 4
VALUE ERROR                       Same error (local variable Z
BIN[1]LA:Z←(Z,0)+0,Z                does not retain its value).
                  ∧
      ∇BIN[.1]Z←1∇                Insert line to initialize Z.
      )SI                         Display state indicator.
BIN[1] *                          Suspended on line 1 of BIN.
      →1                          Resume execution (BIN now cor-
1  4  6  4  1                      rect).
```

```
      ∇BIN[□]∇
    ∇ Z←BIN N
[1]   Z←1
[2]  LA:Z←(Z,0)+0,Z
[3]   →LA×N≥ρZ
    ∇
```

Display revised function and close definition.

```
      SΔBIN←2
      Q←BIN 3

BIN[2]
      Z
1
      →2
```

Set stop on line 2.
Execute BIN.

Stop due to stop control.
Display current value of Z.

Resume execution.

```
BIN[2]
      →2
```

Stop again on next iteration.
Resume.

```
BIN[2]
      →0
```

Stop again.
Branch to 0 (terminate).

INPUT AND OUTPUT

```
      ∇MULTDRILL N;Y;X
[1]   Y←?N
[2]   Y
[3]   X←□
[4]   →0×ιX='S'
[5]   →ιX=×/Y
[6]   'WRONG, TRY AGAIN'
[7]   →3∇
      MULTDRILL 12 12
2  10
□:
      37
WRONG, TRY AGAIN
□:
      20
6  7
□:
      'S'
      ∇Z←ENTERTEXT
[1]   Z←''
[2]   D←ρZ
[3]   Z←Z,⍞
[4]   →2×D≠ρZ
[5]   ∇
```

A multiplication drill.
ρN random integers.
Print the random factors.
Keyboard input.
Stop if entry is the letter S.
Repeat if entry is right product.
Prints if preceding branch fails.
Branch to 3 for retry.
Drill for pairs in range 1 to 12.

Indicates that keyboard entry is awaited.

Entry of letter S stops drill.
Example of Character (⍞) input.
Make Z an empty vector.
D is the length of Z.
Append character keyboard entry.
Branch to 2 if length increased.
(entry was not empty).

```
      Q←ENTERTEXT
THIS IS ALL
 CHARACTER INPUT

      Q
THIS IS ALL CHARACTER INPUT
      N←5
      'NOTE: ⍳';N;' IS ';⍳N

NOTE: ⍳5 IS 1  2  3  4  5

      P←2 3 5 7
      ⍴P
4
      T←'OH MY'
      ⍴T
5
      P,P
2  3  5  7  2  3  5  7
      T,T
OH MYOH MY
      T,P
DOMAIN ERROR
      T,P
      ∧
      M←2 3⍴2 3 5 7 11 13
      M
   2   3   5
   7  11  13
      2 4⍴T
OH M
YOH
      6⍴M
2  3  5  7  11  13
      ,M
2  3  5  7  11  13
      P←,M
      P[3]
5
      P[1 3 5]
2  5  11
      P[⍳3]
2  3  5
      P[⍴P]
13
      M[1;2]
3
      M[1;]
2  3  5
```

Keyboard entries.
Empty input to terminate.
Display Q.

Mixed output statement.

RECTANGULAR ARRAYS

Dimension of P.

Character vector.

Catenation.

Characters cannot be catenated with numbers.

Reshape to produce a 2x3 matrix.

A 2x4 matrix of characters.

A matrix reshaped to a vector.

Elements in row-major order.

Indexing (third element of P).

A vector index.

The first three elements of P.

Last element of P.

Element in row 1, column 2 of M.

Row 1 of M.

```
      M[1 1;3 2]
   5  3
   5  3
      A←'ABCDEFGHIJKLMNOPQ'
      A[M]
BCE
GKM
      A[M[1 1;3 2]]
EC
EC
      M[1;]←15 3 12
      M
  15   3  12
   7  11  13
      Q←3 1 5 2 4 6
      P[Q]
5  2  11  3  7  13
      Q[Q]
5  3  4  1  2  6
      P[3]
5
      )ORIGIN 0
WAS 1
      P[3]
7
      P[0 1 2]
2  3  5
      ⍳5
0  1  2  3  4
      )ORIGIN 1
WAS 0
      ⍳5
1  2  3  4  5
```

Rows 1 and 1, columns 3 2.

The alphabet to Q.
A matrix index produces a matrix result.

Respecifying the first row of M.

A permutation vector.
Permutation of P.

A new permutation.

Present index origin is 1.

Set index origin to 0.

First three elements of P.

Result of index generator begins at origin.

FUNCTIONS ON ARRAYS

```
      V←?3⍴9
      M←?3 3⍴9
      N←?3 3⍴9
      V
2  1  7
```

Vector of 3 random integers (1-9).
Random 3 by 3 matrix.
Random 3 by 3 matrix.

```
      M
 7  9  4
 5  8  1
 1  5  7
      N
 1  4  1
 4  7  6
 9  8  5
      M+N
  8  13   5
  9  15   7
 10  13  12
      M⌈N
 7  9  4
 5  8  6
 9  8  7
      M≤N
0 0 0
0 0 1
1 1 0
      +/V
10
      ×/V
14
      +/[1]M
13  22  12
      +/[2]M
20  14  13
      +/M
20  14  13
      ⌈/M
9  8  7
      X←1.5
      +/(1 2○X)*2
1
      ○/1 2,X
0.07067822453
      Y←○/0 2,X
      Y
0.9974949866
      Y=1○X
1
      M+.×N
  79  123   81
  46   84   58
  84   95   66
      M+.≤N
 1  1  1
 1  1  1
 2  3  2
```

Sum (element-by-element).

Maximum.

Comparison.

Sum-reduction of V.

Product-reduction.

Sum over first coordinate of M (down columns).

Sum over second coordinate of M (over rows).

Sum over last coordinate.

Maximum over last coordinate.

Sin squared plus Cos squared.

Sin (Cos X).

```
(1-(COS X)*2)*.5
```

An identity.

Ordinary matrix (+.× inner) product.

An inner product.

```
      M+.×V
51  25  56
      V
2  1  7
      V∘.×⍳5
 2  4  6  8 10
 1  2  3  4  5
 7 14 21 28 35
      V∘.≤⍳9
0 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1 1
      V∘.×M
 14 18  8
 10 16  2
  2 10 14

  7  9  4
  5  8  1
  1  5  7

 49 63 28
 35 56  7
  7 35 49
```

+.× inner product with vector right argument.

Outer product (times).

Outer product.

An outer product of rank 3.

A blank line between planes.

MIXED FUNCTIONS

```
      Q←?10ρ5
      Q
1  4  3  4  5  4  2  1  4  2
      +/[1]Q∘.=⍳5
2  2  1  4  1

      2 1⍉M
 7  5  1
 9  8  5
 4  1  7
      ⍉M
 7  5  1
 9  8  5
 4  1  7
      T←2 3 4ρ⍳24
      T
 1  2  3  4
 5  6  7  8
 9 10 11 12

13 14 15 16
17 18 19 20
21 22 23 24
```

A random 10 element vector (range 1 to 5).

Ith element of result in number of occurrences of the value I in Q.

Ordinary transpose of M.

Ordinary transpose of M (monadic).

An array of rank 3.

```
      3 1 2⍉T
 1  13
 2  14
 3  15
 4  16

 5  17
 6  18
 7  19
 8  20

 9  21
10  22
11  23
12  24
      1 1⍉M
7  8  7
      1 1 2⍉T
 1   2   3   4
17  18  19  20
      X←○(0,ι5)÷6
      )DIGITS 4
WAS 10
      ⍉1  2  3∘.○X
 0.000E0     1.000E0     0.000E0
 5.000E¯1    8.660E¯1    5.774E¯1
 8.660E¯1    5.000E¯1    1.732E0
 1.000E0     1.744E¯16   5.734E15
 8.660E¯1   ¯5.000E¯1   ¯1.732E0
 5.000E¯1   ¯8.660E¯1   ¯5.774E¯1
      Q
1  4  3  4  5  4  2  1  4  2
      3⌽Q
4  5  4  2  1  4  2  1  4  3
     ¯3⌽Q
1  4  2  1  4  3  4  5  4  2
      0 1 2⌽[1]M
 7  8  7
 5  5  4
 1  9  1
     ¯2⌽[2]M
 9  4  7
 8  1  5
 5  7  1
      1 2 3⌽M
 9  4  7
 1  5  8
 1  5  7
      ⌽Q
2  4  1  2  4  5  4  3  4  1
```

Transpose of T (dimension of result is 3 4 2).

Diagonal of M.

Diagonal section in first two coordinates of T.

Set number of output digits to 4.

Table of sines, cosines, and tangents in intervals of 30 degrees.

Rotate to left by 3 places.

Rotate to right by 3 places.

Rotate columns by different amounts.

Rotation of rows all by 2 to right.

Rotation of rows.

Reversal of Q.

```
      ⌽[1]M
  1  5  7
  5  8  1
  7  9  4
      ⌽M
  4  9  7
  1  8  5
  7  5  1
      U←Q>4
      U
0  0  0  0  1  0  0  0  0  0
      U/Q
5
      (~U)/Q
1  4  3  4  4  2  1  4  2
      +/U/Q
5
      1 0 1/[1]M
  7  9  4
  1  5  7
      1 0 1/M
  7  4
  5  1
  1  7
      (,M>5)/,M
7  9  8  7
      V←1 0 1 0 1
      V\⍳3
1  0  2  0  3
      V\M
  7  0  9  0  4
  5  0  8  0  1
  1  0  5  0  7
      V\'ABC'
A  B  C
      10⊥1 7 7 6
1776
      8⊥1 7 7 6
1022
      (4⍴10)⊤1776
1  7  7  6
      (3⍴10)⊤1776
7  7  6
      10 10⊤1776
7  6
      10⊤1776
6
      24 60 60⊥1 3 25
3805
```

Reversal of M along first coordinate.

Reversal along last coordinate.

Compression of Q by logical vector U.

Compression by not U.

Compression along first coordinate of M.

Compression along last coordinate.

,M is 7 9 4 5 8 1 1 5 7
All elements of M which exceed 5.

Expansion of iota 3.

Expansion of rows of M.

Expansion of literal vector inserts spaces.

Base 10 value of vector 1 7 7 6.

Base 8 value of 1 7 7 6.

4-digit base 10 representation of number 1766.

3-digit base 10 representation of number 1766.

Mixed base value of 1 3 25 (time radix).

```
      24 60 60⊤3805
1  3  25
      2⊥1 0 1 1 0
22
      M
  7  9  4
  5  8  1
  1  5  7
      )ORIGIN 0

WAS 1
      M[2;0]
1
      (,M)[(⍴M)⊤2,0]
1
      )ORIGIN 1
WAS 0
      P
2  3  5  7  11  13
      P⍳7
4
      P⍳6
7
      P⍳4 5 6 7
7  3  7  4
      Q←5 1 3 2 4
      R←Q⍳⍳⍴Q
      R
2  4  3  5  1
      Q[R]
1  2  3  4  5
      A←'ABCDEFGHIJKLMNOPQ'
      A←A,'RSTUVWXYZ'
      A
ABCDEFGHIJKLMNOPQRSTUVWXYZ
      A⍳'C'
3
      J←A⍳'CAT'
      J
3  1  20
      A[J]
CAT
      M←3 5⍴'THREESHORTWORDS'
      M
THREE
SHORT
WORDS
```

Representation of number 3805 in time radix.
Base 2 value.

Indexing of matrix in 0-origin. Note relation to indexing of ravel of M.

Restore 1-origin.

Index of 7 in vector P.
7 is 4th element of P.
6 does not occur in P, hence result is 1+⍴P.

A permutation vector.
R is the permutation inverse to Q.

A is the alphabet.

Rank of letter C in alphabet is 3.

A matrix of characters.

```
      J←A⍳M
      J
 20   8  18   5   5
 19   8  15  18  20
 23  15  18   4  19
      A[J]

THREE
SHORT
WORDS
      3?5
5  1  2
      6?5
DOMAIN ERROR
      6?5
       ∧
      X←8?8
      X
4  6  7  2  5  1  8  3
      ⍋X
6  4  8  1  5  2  3  7
      X[⍋X]
1  2  3  4  5  6  7  8
      X[⍒X]
8  7  6  5  4  3  2  1
      U←A∈'NOW IS THE TIME'
      '01'[1+U]
0000100110001110001100100 0
      U/A
EHIMNOSTW
      (⍳8)∈3 7 5
0  0  1  0  1  0  1  0
```

Ranking of M produces a matrix.

Indexing by a matrix produces a matrix.

Random choice of 3 out of 5 without replacement.

A random permutation vector.

Grading of X.

Arrange in ascending order.

Arrange in descending order.

Membership.

BIBLIOGRAPHY

Anscombe, F.J., Use of Iverson's Language APL for Statistical Computing. New Haven: Department of Statistics, Yale University, July 1968.

APL\1130 Primer. IBM Corporation, 1968 (C20-1697).

APL\360 User's Manual. IBM Corporation, 1969 (H20-0683).

APL\360 Primer. IBM Corporation, 1969 (GH20-0689).

Falkoff, A.D. and Iverson, K.E. "The APL\360 Terminal System", Interactive Systems for Applied Mathematics. New York & London: Academic Press, 1968, pp. 22-37.

Falkoff, A.D., Iverson, K.E. and Sussenguth, E.H. "A Formal Description of System/360", IBM Systems Journal. III, No. 3 (1964), pp. 193-262.

Hellerman, H., Digital Computer System Principles. New York: McGraw-Hill, 1967.

Iverson, K.E., "A Common Language for Hardware, Software and Applications", Eastern Joint Computer Conference, December, 1962, pp. 121-129 (RC 749).

Iverson, K.E., "The Description of Finite Sequential Processes", Proceedings of the Fourth London Symposium on Information Theory, August, 1960, Colin Cherry, Ed., Butterworth and Company.

Iverson, K.E., Elementary Functions: An Algorithmic Treatment. Chicago: Science Research Associates, 1966.

Iverson, K.E., Formalism in Programming Language. Yorktown Heights: T. J. Watson Research Center, IBM Corporation, July 2, 1963.

Iverson, K.E., "A Programming Language", Spring Joint Computer Conference, May 1962. pp. 245-351.

Iverson, K.E., A Programming Language. New York: John Wiley & Sons, Inc., 1962.

Iverson, K.E., "Programming Notation in Systems Design", IBM Systems Journal, June, 1963.

Iverson, K.E., "Recent Applications of a Universal Programming Language", New York: IFIP Congress, May 24, 1965.

Iverson, K.E., The Role of Computers in Teaching. Kingston, Ontario, Canada: Queen's University, Queen's Papers on Pure and Applied Mathematics, No. 13, 1968.

Iverson, K.E., The Use of APL in Teaching. IBM Corporation, 1969. (320-0996).

Kolsky, H., "Problem Formulation in APL", IBM Systems Journal #3 (1969). (G231-0018).

Montalbano, M., "High-Speed Calculation of the Critical Paths of Large Networks", IBM Systems Journal, Vol. 6 #3, 1967.

Pakin, Sandra, APL\360 Reference Manual. Chicago: Science Research Associates, 1968. (NO. 17-1).

Raucher, S.M., "APL and Its Use in the Classroom", Journal of the Association for Educational Data Systems, December, 1968.

Smillie, K.W., STATPACK 2: An APL Statistical Package. Edmonton, Alberta, Canada: Department of Computer Science, University of Alberta, Publication No. 17, January, 1968.

Surkan, A.J., "Symbolic Polynomial Operations with APL," IBM Journal Research and Development, March, 1969.

Thurber, K.J. and Myrna, J.W., "Systems Design of a Cellular APL Computer", IEEE Transactions on Computers, Vol C-19 #4, April, 1970.

Woodrum, L.,"Sorting Techniques", IBM Systems Journal #3 (1969).

READER'S COMMENT FORM

APL\360 OS & APL\360 DOS
GIM

GH20-0850-1

Please comment on the usefulness and readability of this publication, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you wish a reply, be sure to include your name and address.

COMMENTS

fold

fold

fold

fold

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A. FOLD ON TWO LINES, STAPLE AND MAIL.

YOUR COMMENTS PLEASE...

Your comments on the other side of this form will help us improve future editions of this publication. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material.

fold fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY...

IBM Corporation
1133 Westchester Avenue
White Plains, N.Y. 10604

Attention: Technical Publications

fold fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

APL\360 OS & APL\360 DOS GIM Printed in U.S.A. GH20-0850-1

GH20-0850-1

APL\360 OS & APL\360 DOS GIM Printed in U.S.A. GH20-0850-1

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)