Reports and Articles

A Comparative Evaluation of Versions of BASIC

Bennet P. Lientz
University of California, Los Angeles

From its inception, The BASIC language has grown in terms of its usage, scope of usage, and its features. This article compares ten of the current versions of BASIC with each other, with two earlier versions, and with the proposed standard for minimal BASIC. The comparison is arranged by the features of the versions and by computational comparison of computation and times and processing costs.

Key Words and Phrases: BASIC, interpretive language summary
CR Categories: 4.20, 4.6

This work was partially supported by The Information Systems Program, Office of Naval Research under contract N00014-75-C-0266, project NR 049-345. Author's address: Graduate School of Management, University of California, Los Angeles, CA 90024.

1. Introduction

BASIC (Beginners All-purpose Symbolic Instruction Code) was originally developed at Dartmouth College for use by nonprogrammers. It was intended to be used interactively and designed for use in program construction and debugging (see, for example, Bull [8] and Lipp [17]). BASIC has been successfully used at the secondary school level (Koetke [15]). However, the range of usage of BASIC has grown beyond the scope of the originally intended audience. Usage has expanded in universities as well as industrial organizations. Some references are Wilkins and Klopfenstein [30] (laboratory work) and [27] (transmission line problems). An information and retrieval system in BASIC is described in McGeachie and Kreider [18].

The usage of BASIC has been cited as being substantial (Sammet [23], Rosen [22]). This is supported by a DATAPRO report [1], where a survey of 101 organizations revealed that while Fortran was used by 65 percent of the respondents, BASIC was second with 49 percent. This can be contrasted to an earlier study by Philippakis [19] which indicated more limited usage when compared to Cobol, Fortran, Assembler, PL/I and RPG for internal computer usage. Of the 98 timesharing utilities surveyed in [1] over half (58) offer a version of BASIC. BASIC is offered by many utilities in both interactive and batch modes and is available on minicomputers as well as larger computers.

With respect to the language itself, Lee [16] proposes a formal definition for BASIC. The American National Standards Institute (ANSI) is currently working on standards for a minimal BASIC. Braden and Wolf [7] discuss the implementation of BASIC. Arthur [2] explores extensions to BASIC for a real time setting. IBM has developed a version of BASIC (VS BASIC) for use under its timesharing system TSO.

There has been reaction to extensions of the language: one example is Oglin [20], who indicates that BASIC has suffered from this divergent development.

Given the usage, variety of versions, and numbers of vendors, it is useful to evaluate some of the popularly used versions with each other, the proposed ANSI Standard, and with earlier versions of BASIC. A series of tabular comparisons compare the following versions of BASIC, in addition to two earlier versions (CALL OS and BASIC ITF), and the proposed ANSI Standard (Section 2).

- ☐ Digital Equipment Corp. (Tymshare version)
- ☐ General Electric
- ☐ Hewlett-Packard
- ☐ IBM
- ☐ International Timesharing Corp. (ITS)
- ☐ Rapidata
- ☐ Service Bureau Corp. (SBC)
- ☐ Tymshare
- ☐ United Computing Systems (UCS).

In addition, it is of interest to further compare these in terms of processing costs. In Section 3 a computational comparison is carried out which is applicable to the potential user who inputs data for a program, runs the program on this data, and writes a report. The comparison is limited in scope so that an individual comparison should be carried out by each potential user to

2.1 General Limitations.

Characteristics	ANSI	BASIC ITF	CALL OS	Dartmouth	DEC-TYM	GE	HP	IBM	ITS	Rapidata	SBC	Tymshare	UCS
Precision													
single	–	7	7	7	8	8	6-7	7	10	7	7	11	14
double	–	15	16	–	–	19	–	15	–	15	15	17	28
Range of nos.													
Largest	E+38	E+75	E+75	E+38	E+38	E+38	E+38	E+75	E+308	E+76	E+75	E+75	E+75
smallest	E−38	E−78	E−78	E−38	E−38	E−38	E−38	E−78	E−308	E−77	E−78	E−78	E−78
Max. line no. (digits)	4	5	5	5	5	5	4	5	6	5	5	5	5
Largest program (statements)	Limited by core	600	800	Limited by core	Limited by core	Limited by core	Limited by core	1,000	752 (inter.) 2,000 (batch)	Limited by core	2,400	Limited by core or 8,000 char.	Limited by core
Character arrays	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
Largest character string	18	18	18	4095	single digit	4095	72	255	79	255	255	Limited by core	255
Debugging capability	–	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Max. array size (elements)	Limited by core	255	7,000	Limited by core	Limited by core	Limited by core	49,000	32,767	Approx. 10,000	Limited by core	26,623 (short form), 13,311 (long form)	Limited by core	Limited by core
Max. no. of dimensions	2	2	2	2	2	2	2	2	2	3	2	Limited by core	Limited by core

reflect his own software. Section 4 presents the conclusion of the paper.

2. Feature and Characteristic Comparison

This section contrasts versions of BASIC in terms of their limitations, constant and intrinsic function features, and I/O capabilities. The versions contrasted and their references are listed here:

- ☐ Proposed ANSI Standards for minimal BASIC [21]
- ☐ BASIC ITF and CALL OS
- ☐ Dartmouth [13]
- ☐ Digital Equipment Corp. (Tymshare version DEC-TYM) [11]
- ☐ General Electric Information Systems (GE) [5]
- ☐ Hewlett-Packard (H-P) [6]
- ☐ IBM [26]
- ☐ International Timesharing Corp. (ITS) [14]
- ☐ Rapidata [4]
- ☐ Service Bureau Corp. (SBC) [9]
- ☐ Tymshare (TYMSH) [25]
- ☐ United Computing Systems (UCS) [29]

The selection of the versions was

2.2 Matrix Operations.

	ANSI	BASIC ITF	CALL OS	Dartmouth	DEC-TYM	GE	HP	IBM	ITS	Rapidata	SBC	Tymshare	UCS
2.2.1 Matrix Function Operations													
A=B	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
A+B	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
A−B	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
A*B	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
A*a	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
A=const.	–	N	N	N	Y	N	N	Y	N	Y	N	Y	N
CON	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
INV	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
IDN	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
TRN	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
ZER	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
DOT	–	N	N	N	N	N	N	Y	N	N	N	N	N
PRD	–	N	N	N	N	N	N	Y	N	N	N	N	N
SUM	–	N	N	N	N	N	N	Y	N	N	N	N	N
ASORT	–	N	N	N	N	N	N	Y	N	N	N	N	N
DSORT	–	N	N	N	N	N	N	Y	N	N	N	N	N
DET	–	N	N	Y	Y	Y	N	Y	Y	N	N	Y	N
2.2.2 Matrix I/O Operations													
MAT INPUT	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
MAT GET	–	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
MAT READ	–	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MAT READ FILE	–	N	N	N	Y	Y	Y	Y	Y	Y	N	Y	Y
MAT PUT	–	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
MAT PRINT	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MAT PRINT USING	–	Y	Y	Y	Y	Y	Y	Y	N	Y	N	Y	Y
MAT WRITE FILE	–	N	N	Y	Y	Y	Y	Y	Y	Y	N	N	Y
MAT REWRITE FILE	–	N	N	N	N	N	N	Y	N	Y	N	Y	N

based on a representative set across different types of computers as well as across the size of the utilities. Some of the utilities listed offer multiple versions of BASIC. For example, Rapidata offers BASIC on Honeywell, Digital Equipment, and IBM computers. The version used for Rapidata in the Exhibits is that used on the Honeywell and Digital Equipment computers.

Tymshare supplies BASIC on Xerox and Digital Equipment computers. The version used in this section for Tymshare is that used on the Xerox 940 computer. The Digital Equipment version for Tymshare is an enhanced version of that offered by Digital Equipment for the PDP 1070. The specific hardware for each utility is examined in Section 3.

The Exhibits in this section partition the features and characteristics of the versions into the categories:

(1) general limitations,
(2) matrix operations and I/O,
(3) intrinsic functions,
(4) intrinsic functions relating to strings and files,
(5) intrinsic constants,
(6) user defined functions,
(7) I/O capabilities,
(8) miscellaneous features.

The data in each table was verified individually by representatives from General Electric, Hewlett-Packard, International Timesharing Corp., Rapidata, Service Bureau Corp, Tymshare, and United Computing Systems. Professor Stephen Garland provided information on the Dartmouth version as well as the program runs in Section 3 for Dartmouth.

2.1 General Limitations

Each version of BASIC has certain limitations and restrictions due to design and implementation. Exhibit 2.1 is a general comparison of the limitations of the various versions. In the Exhibit debugging capability is an indicator of whether the version has the capability to change programs in an interactive mode.

2.2 Matrix Operations and I/O

Two Exhibits are given in this section. One (Exhibit 2.2.1) contrasts the matrix operations by version. The name for the operations varies somewhat but the functions are defined as follows:

CON	set all matrix elements equal to unity
INV	invert the matrix
IDN	set the matrix to the identity matrix
TRN	obtain the transpose of the matrix
ZER	set all elements of the matrix equal to zero
DOT	dot product of vectors
PRD	product of elements in an array
SUM	sum of the elements of the array
ASORT	obtain a matrix with elements in ascending order (e.g. $-b_{11} \leq b_{12} \leq \cdots \leq b_{21} \leq b_{22} \leq \cdots$) from the original matrix
DSORT	obtain a matrix with elements in descending order from the original matrix
DET	obtain the determinant of the matrix.

Exhibit 2.2.2 contrasts the following matrix I/O operations.

MAT INPUT	assign values to matrix from terminal at execution time
MAT GET	retrieve a matrix from a file and read it into a program
MAT READ	assign values to matrix from data created by DATA statement
MAT READ FILE	assign value to matrix from a specified record oriented file
MAT PUT	write an output matrix from a program to a file
MAT PRINT	print matrix in matrix format
MAT PRINT USING	print matrix at a terminal in output form desired
MAT WRITE FILE	add a record to a file
MAT REWRITE FILE	change a record in a file

Some of these operations require the use of supporting software. For example, the IBM VS BASIC requires the use of VSAM for MAT READ FILE, MAT WRITE FILE, and MAT REWRITE FILE.

2.3 Intrinsic Functions

Most current versions offer several intrinsic functions for ease of use. Exhibit 2.3 contains a comparison of the following functions.

ABS	absolute value
ACS	arc cosine
ASN	arc sine
ATN	arc tangent
CEN	conversion from Fahrenheit to centigrade
CLK	time of day in hours, minutes, seconds
COS	cosine
COT	cotangent
CPU	amount of processing units from time of compilation or execution up function
CSC	cosecant
DAT	current Gregorian date
DAY	conversion of data to single number
DEG	conversion from radians to degrees
EXP	natural exponent
FAH	conversion from centigrade to Fahrenheit
HCS	hyperbolic cosine
HSN	hyperbolic sine
HTN	hyperbolic tangent
INT	integer part of quantity
JDY	current Julian date
LGT	logarithm to the base 10
LOG	natural logarithm
LTW	logarithm to the base 2
MAX	maximum value
MIN	minimum value
NUM	arithmetic value of character string
RAD	conversion from degrees to radians
SEC	secant
SGN	signum function ($\mathrm{SGN}(x) = 1$ if $x > 0$, $\mathrm{SGN}(x) = 0$ if $x = 0$, $\mathrm{SGN}(x) = -1$ if $x < 0$)
SIN	sine
SQR	square root
TAN	tangent

In addition to these functions, some versions allow for complex numbers (see data types in Exhibit 2.8). With these are functions for complex conjugates, polar coordinates, and real and imaginary parts of a complex number.

2.4 Intrinsic Functions Relating to Strings and Files

Exhibit 2.4 gives some of the intrinsic functions for strings and files. Several of these relate to input into sequenced files The function KPS finds the embedded key in the file (bytes position at which the embedded key starts). KLN gives the length of the key in the file.

The remainder of the functions relate to strings and are described as

IDX position of first character of a string within a string
LEN length of character string

The function STR(x,y) is the portion of the string x from the yth character to the end of the string.

2.5 Intrinsic Constants

Some constants are regularly used in several application areas. Exhibit 2.5 compares the constants:

&DPI double precision pi
&PI pi
&E value of e
&SQR2 value of 2
&INCM centimeters per inch
&LBKG kilograms per pound
&GALI liters per gallon

2.6 User Defined Functions and Subroutines

The user defined functions may be limited in terms of the limit of a single line, multiple lines, or multiple arguments. Exhibit 2.6.1 contains these comparisons.

The comparison for subroutines is contained in Exhibit 2.6.2. Subroutines are accessed by GOSUB statements. Like GOTO statements some versions have computed GOSUB statements. For some applications a main program in BASIC may need to access subroutines in non-higher-level languages such as Cobol or Fortran. The ability to access subroutines in non-BASIC languages is a feature shown in 2.6.2.

2.7 I/O Capabilities

There are three areas of I/O capabilities considered here (internal, external, and terminal). The internal I/O consists of the statements DATA (create table of values in a program), READ (read from DATA table and assign to variable), and RESTORE (reset table pointer to first entry). Since all versions compared have each of these statements no exhibit is included.

Exhibit 2.7.1 gives the external I/O characteristics. The number of files refers to the number of simultaneous open files. The file name(s)

	ANSI	BASIC ITF	CALL OS	Dartmouth	DEC-TYM	GE	HP	IBM	ITS	Rapidata	SBC	Tymshare	UCS
2.3 Intrinsic Functions													
ABS	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
ACS	-	Y	Y	N	N	N	N	Y	N	N	Y	Y	N
ASN	-	Y	Y	N	N	N	N	Y	N	N	Y	Y	N
ATN	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
CEN	-	N	N	N	N	N	N	Y	N	N	N	N	N
CLK	-	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
COS	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
COT	-	Y	Y	Y	Y	Y	N	Y	Y	N	Y	N	N
CPU	-	N	N	Y	Y	N	N	Y	Y	Y	Y	Y	Y
CSC	-	Y	Y	N	N	N	N	Y	N	N	Y	N	N
DAT	-	N	N	Y	Y	Y	N	Y	Y	Y	Y	Y	Y
DAY	-	N	N	N	N	N	N	N	N	N	Y	Y	Y
DEG	-	Y	Y	N	N	N	N	Y	N	N	Y	N	N
EXP	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
FAH	-	N	N	N	N	N	N	Y	N	N	N	N	N
HCS	-	Y	Y	N	N	N	N	Y	N	N	Y	Y	N
HSN	-	Y	Y	N	N	N	N	Y	N	N	Y	Y	N
HTN	-	Y	Y	N	N	N	N	Y	N	N	Y	Y	N
INT	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
JDY	-	N	N	N	N	N	Y	Y	N	N	N	N	N
LGT	-	Y	Y	N	Y	Y	N	Y	Y	Y	Y	Y	Y
LOG	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
LTW	-	Y	Y	N	N	Y	N	Y	N	N	Y	Y	Y
MAX	-	N	N	N	Y	N	N	Y	N	Y	N	Y	N
MIN	-	N	N	N	Y	N	N	Y	N	Y	N	Y	N
NUM	-	N	N	Y	Y	Y	N	Y	Y	Y	N	Y	Y
RAD	-	N	Y	N	N	N	N	Y	N	N	Y	N	N
RND	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
SEC	-	Y	Y	N	N	N	N	Y	N	N	Y	N	N
SGN	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
SIN	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
SQR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
TAN	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
2.4 Intrinsic Functions Relating to Strings and Files													
IDX	-	N	N	Y	Y	Y	N	Y	Y	Y	Y	Y	Y
KLN	-	N	N	N	N	N	N	Y	N	Y	N	N	N
KPS	-	N	N	N	N	N	N	Y	N	N	N	N	N
LEN	-	N	N	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
RLN	-	N	N	N	N	Y	N	Y	N	Y	N	Y	Y
STR	-	N	N	Y	Y	Y	N	Y	Y	Y	N	Y	Y
2.5 Intrinsic Constants													
&DPI	-	N	N	N	N	N	N	N	N	N	N	Y	N
&PI	-	Y	Y	N	Y	N	N	Y	Y	Y	Y	Y	Y
&E	-	Y	Y	N	N	N	N	Y	Y	N	Y	N	N
&SQR2	-	Y	Y	N	N	N	N	Y	N	N	Y	N	N
&INCM	-	N	N	N	N	N	N	Y	N	N	N	N	N
&LBKG	-	N	N	N	N	N	N	Y	N	N	N	N	N
&GALI	-	N	N	N	N	N	N	Y	N	N	N	N	N

2.6 Functions and Subroutines.

Capability	ANSI	BASIC ITF	CALL OS	Dartmouth	DEC-TYM	GE	HP	IBM	ITS	Rapidata	SBC	Tymshare	UCS
2.6.1 User Defined Function Limitations													
Single line	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Multiple lines	-	N	N	Y	Y	Y	N	Y	Y	Y	N	Y	N
Multiple arguments	-	N	N	Y	Y	Y	N	Y	Y	Y	N	Y	Y
2.6.2 Subroutines													
GOSUB	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Computed GOSUB	-	N	N	Y	N	Y	Y	Y	Y	N	Y	Y	Y
Access to subroutines in non-BASIC languages	-	N	N	N	N	N	Y	*	N	N	N	N	N

* Functions can be written in a non-BASIC language and can be in a private library.

2.7 I/O Capabilities.

Characteristic	ANSI	BASIC ITF	CALL OS	Dartmouth	DEC-TYM	GE	HP	IBM	ITS	Rapidata	SBC	Tymshare	UCS
2.7.1 External I/O													
Sequential access	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Direct access	-	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
No. of files	-	No limit	4	16	12	8	16	15	4	15	Limit imposed by core	4	24
File name (characters)	-	3	8	8	9	8	6	8	9	14	8	45	7
EOF checks	-	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Reset files to start	-	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Reset files to spec. record	-	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Read, write same file	-	N	N	Y	Y	N	Y	Y	N	N	Y	Y	N
Update file in place	-	N	N	Y	Y	N	Y	Y	N	N	Y	Y	N
Reread file	-	N	N	Y	N	Y	Y	Y	N	N	Y	Y	N
Delete record	-	N	N	N	Y	Y	Y	Y	Y	Y	N	Y	Y
Formatted I/O	-	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
2.7.2 Terminal I/O													
Input	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Print	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Time out on input	-	N	N	N	Y	N	N	Y	N	Y	N	N	N
Formatted print	-	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Exit	-	N	N	N	N	Y	N	Y	N	N	Y	Y	Y

is the maximum number of characters possible for the file name.

Terminal I/O statements are given in Exhibit 2.7.2. Formatted print is the capability to have terminal output formatted using skipping and positioning.

2.8 Miscellaneous Features and Characteristics

Exhibit 2.8 contains a comparison for several features not covered in previous exhibits including the CHAIN statement itself and the ability to pass and access parameters. *Data types* refer to the types of numbers that can be handled (complex, integer).

3. Computational Comparison

The purpose of a computational comparison is to determine for a class of problems the comparative variation between versions of BASIC. There are several criteria involved in such a comparison. First, the program should not depend on software beyond BASIC and the operating system structure. Second, it should be minimally dependent on the operating system. The violation of either of these criteria will cloud the comparison. Third, the comparison should be carried out to represent a class of users of BASIC. With the growth of BASIC this is more difficult, since we can see from Section 1 that there are sophisticated programmer users with production systems as well as nonprofessional programmers who are more casual users.

Based on these criteria a BASIC program was written to compute the determinant of a matrix. The algorithm is based on work by Trotter [28]. This program satisfies the first two criteria on software dependence. It meets the third criterion since the process involving the program consists of establishing a small program, inputting data, running the program, and writing a report at a terminal.

A similar comparison was used for Fortran [12]. It is noted that

2.8 Miscellaneous Features and Characteristics.

	ANSI	BASIC ITF	CALL OS	Dartmouth	DEC-TYM	GE	HP	IBM	ITS	Rapidata	SBC	Tymshare	UCS
Characters Letters, Nos., @, $, #	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Operators Standard	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
and	–	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
or	–	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
concatenation	–	No	No	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes
IF THEN ELSE	–	No	No	No	Yes	No	No	Yes	No	No	No	Yes	Yes
IF THEN	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CHAIN	–	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
pass param.	–	–	–	No	No	Char. String	No	Char. String	No	No	Char. String	Char. String	–
access param.	–	–	–	No	No	Yes	No	Yes	–	–	Yes	Yes	–
Character strings variable	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
vector	–	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
arrays	–	No	No	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes
Max. length of string variables	18	18	18	4095	255	158	72	255	255	255	Limited by core	Limited by core	Limited by core
Data types	Integer	Integer	Integer	Integer	Integer	Integer	Integer	Integer	Integer	Integer	Integer	Complex, Integer	Complex, Integer

even though the comparison meets the criteria mentioned already this is not sufficient for a potential user organization. In that case it is best to evaluate and compare based on a mixture of the workload in BASIC. This might lead to results substantially different from the comparison presented here.

It can be argued that a computational comparison is irrelevant for a language that was not intended to be a production language, but rather to be an easy to use language. However, this remark does not apply to the type of work done by non-professional programmers using BASIC extensively.

In our comparison the processing change was used as the measure. The costs of terminal connection were excluded, as were other costs and volume discounts. The program was run three times for matrices of sizes 5×5, 6×6, and 7×7. Exhibit 3.1 gives the processing units and cost for the average of each problem. The processing units varied substantially due to the differences in the definition of a processing unit between utilities. Problems 1, 2, and 3 refer to the 5×5, 6×6, and 7×7 matrices, respectively.

In addition to the processing information, the machine used is given for each set of runs. The processing cost per unit is given in Exhibit 3.2. The IBM runs for VS BASIC were performed on an IBM 370/168 with an estimated rate of $793 per hour. All other rates were at standard commercial rates available to any user.

A further comparison can be made by factoring the cost by the cycle time of the hardware in microseconds to lessen the dependence of the results on the computers used. Data was collected from Auerbach [3] and DATAPRO [10]. Exhibit 3.3 gives this comparison.

From Exhibit 3.1 the mean, median, standard deviation, and range can be computed.

	Problem 1	Problem 2	Problem 3
Mean	.246	.6068	3.5633
Median	.25	.5025	1.6555
Std.dev.	.094	.3345	2.81
Range	.313	1.542	11.853

For problem 1 the utility (ITS) with the highest cost is ten times the lowest (UCS). This spread increases with problems 2 and 3. Leaving out UCS the cost spread is still greater than than a factor of 2.

In Exhibits 3.1 and 3.3 a crossover can be observed for several cases. For example, the costs of Rapidata's PDP 1070 for processing problems 1 and 2 is more than those of SBC, but is less for problem 3. This is caused by the processing unit algorithm since both cost per unit and cycle time are applied uniformly.

From Exhibit 3.3 the mean, median, standard deviation, and range are given by:

	Problem 1	Problem 2	Problem 3
Mean	.2894	.7448	4.0958
Median	.2990	.6705	3.9255
Std.dev.	.29	.44	2.99
Range	.325	1.524	11.261

The size of the standard deviation is still substantial.

4. Conclusion

The feature comparison and the computational comparison are the more tangible parts of an evaluation for a specific user. However, also involved are the ease of use

3.1 Computational Comparison.

Utility	Machine	Problem 1 Units	Problem 1 Cost ($)	Problem 2 Units	Problem 2 Cost ($)	Problem 3 Units	Problem 3 Cost ($)
Dartmouth	Honeywell 6635	.328, 52	.31	1.02, 56	.433	5.936, 60	1.19
G.E.	Honeywell 6080	1.70	.17	5.25	.525	33.11	3.311
H-P	Hewlett-Packard 2000F	1.56	.156	6.33	.633	56.27	5.627
IBM	370/168 (under TSO)	1*	.244*	3	.472	10	2.44
ITS	CDC-3300	3	.36	5.5	.66	28	3.36
Rapidata							
	Honeywell 437	5.8	.348	14.6	.876	78.3	4.698
	370/145 (under CMS)	.94, 88	.298	1.75, 88	.46	9.75, 88	2.06
	PDP 1070	5.4	.324	12.6	.756	63.9	3.834
SBC	IBM 370/158	1	.16	3	.48	18	2.88
Tymshare	XDS-940	6	.30	30	1.50	226	11.30
	PDP 1070	.52	.25	2.62	.45	6.39	2.02
UCS	CDC-6500	.799, 15	.035	.874, 15	.037	.974, 15	.039

* This is based on a one-second minimum.

3.2 Charge Formulas.

Utility	Processing Unit Charge
Dartmouth	$.15 per unit, $5.00/1000 I/O
G.E.	$.10 per unit
H-P	$.10 per unit
IBM	$.22 per unit
IMS	$.12 per unit
Rapidata	
Honeywell, DEC	$.06 per unit
IBM	$.20 per CPU unit, $.00125 per I/O
SBC	$.16 per unit
Tymshare	$.05 per CPU unit (XDS and PDP)
UCS	$.025 per 1024 words/unit, .001 per program file I/O

3.3 Run Cost Factored by Cycle Time (in μsec).

Utility	Problem 1	Problem 2	Problem 3
Dartmouth	$.31	$.433	$1.19
G.E.	.34	1.05	6.62
H-P	.159	.645	5.747
IBM	.508	1.524	5.08
ITS	.288	.528	2.688
Rapidata			
Honeywell 437	.129	.324	1.733
IBM 370/145	.552	.852	3.815
PDP 1070	.341	.796	4.036
SBC	.232	.696	4.174
Tymshare			
XDS 940	.316	1.579	11.895
PDP 1070	.263	.474	2.126
UCS	.035	.037	.039

of the timesharing system, the cost characteristics offered by the utility, and other variables.

The feature comparison reveals that the current versions satisfy the proposed ANSI Standard for minimal BASIC. Many offer much more to be competitive and to be easier to use. The remarks in Section 1 on the changes undergone by BASIC are supported by comparing BASIC ITF or CALL OS with later versions. There are, however, limitations for some versions which substantially affect the scope of applications. One is the 1000 statement limit per program in VS BASIC. Another is the small range of intrinsic functions offered in other versions. There remain drawbacks in file I/O and in accessing subroutines not written in BASIC. The computational comparison reveals a somewhat surprising disparity in costs especially given the limited nature of the program used.

Both Sections 2 and 3 indicate that there are substantial differences between versions which should be evaluated for each application area.

Acknowledgment. The author wishes to express his appreciation to Robert E. Yarnall for motivating this work, to the referees for their helpful suggestions, to Thomas Kurtz for his comments regarding the proposed ANSI standard for minimal BASIC, and to the staff members of the firms cited in the paper.

References

1. All about remote computing services. DATAPRO Res. Corp., 1975.
2. Arthur, A.J. Realtime Extensions to BASIC. *IBM Tech. Disclosure. Bull. 17*, 3, (Oct.-Nov. 1974).
3. *Auerbach Standard EDP Reports*, *3*, 6 (1972).
4. *BASIC*. Rapidata, Inc., 1973.
5. *BASIC Language—Mark III Foreground Reference Manual*. General Electric Co., Nov. 1973.
6. *BASIC/3000 Programmers Pocket Guide*. Hewlett-Packard, Feb. 1973.
7. Braden, H.V., and Wolf, W.A. The implementation of a BASIC system in a multiprogramming environment. *Comm. ACM 11*, 10 (Oct. 1968), 688-692.
8. Bull, G.M. Dynamic debugging in BASIC. *Computer J. 15* (1972), 21-24.
9. *CALL/370 BASIC Reference Manual*, Service Bureau Corp., 1974.
10. *DATAPRO, 1*. DATAPRO Res. Corp., 1974.
11. *DEC System 10 BASIC—Conversational Language Manual*. Digital Equipment Corp., March 1973 (Tymshare-PDP 1070).
12. Fortran System for PDP 11's. *Computer Digest 9* (1974), 9-10.
13. Garland, S.J. *Dartmouth BASIC-A Specification*. Kiewit Computation Center, Dartmouth College, 1973.
14. *ITS—EXBASIC Reference Manual*. International Timesharing Corp., 1971.
15. Koetke, W. The impact of computing on the teaching of mathematics. AFIPS Conf. Proc. Vol. 40, 1972 SJCC, AFIPS Press, Montvale, N.J., 1972, pp. 1043-1049.
16. Lee, J.N. The formal definition of the BASIC language. *Comp. J. 15* (1972), 37-41.
17. Lipp, M. The language BASIC and its role in timesharing. *Computers and Automation* (Oct. 1969) 42-43.
18. McGeachie, J.S., and Kreider, D.L. Project FLNP—an integrated information and modeling system for management. AFIPS Conf. Proc., Vol. 43, 1974 NCC, AFIPS Press, Montvale, N.J., pp. 529-535.
19. Philippakis, S. Programming language usage. *Datamation 19* (1973), 109-114.
20. Oglin, J.L. The case against . . . BASIC. *Datamation 17* (1971), 34-41.
21. *Proposed standard for Minimal BASIC*. ANSI, 1975.
22. Rosen, S. Programming systems and languages: history and future. *Comm. ACM 15*, 7 (July 1972), 591-600.
23. Sammet, J.E. Programming Languages: history and future. *Comm. ACM, 15*, 7 (July, 1972), 601-610.
24. Spencer, D.G. *A Guide to BASIC Programming: A Timesharing Language*. Addison-Wesley, Reading, Mass, 1970.
25. *Super BASIC*. Tymshare Manuals Reference Series, Tymshare, Inc., May 1971 (XDS-940).
26. *System/370-VS BASIC Language*. IBM Manual GC-8303-0, 1974.
27. Transmission line problems solved fast with BASIC. *Electr. Des. 17* (1971), 76.
28. Trotter, H.F. Algorithm 115—PERM. *Comm. ACM 5*, 8 (Aug. 1962), 434-435.
29. *UCS-VI Super BASIC Reference Manual*. United Computing Systems, 1972.
30. Wilkins, C.L., and Klopfenstein, C.E. Laboratory computing with real time BASIC. *Chemtech* (Sep. 1972), 560-564.