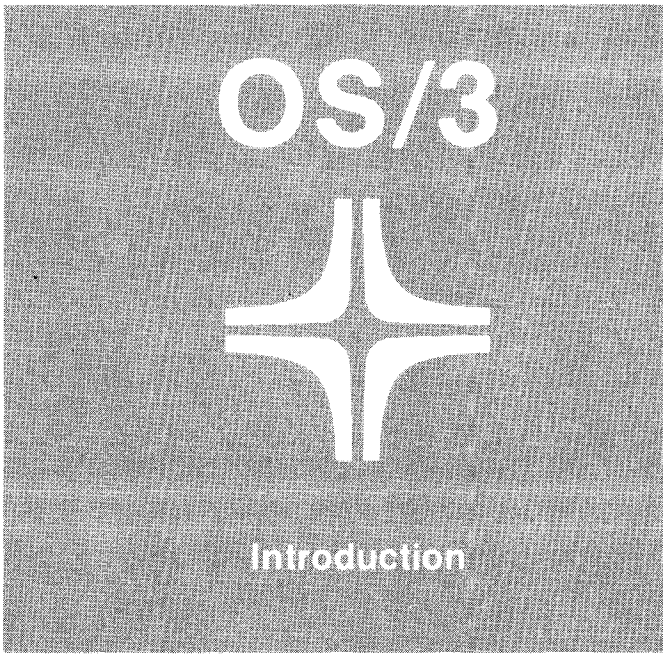
BASIC



SPERRY UNIVAC

UP-8801

H

This document contains the latest information available at the time of preparation. Therefore, it may contain descriptions of functions not implemented at manual distribution time. To ensure that you have the latest information regarding levels of implementation and functional availability, please consult the appropriate release documentation or contact your local Sperry Univac representative.

Sperry Univac reserves the right to modify or revise the content of this document. No contractual obligation by Sperry Univac regarding level, scope, or timing of functional implementation is either expressed or implied in this document. It is further understood that in consideration of the receipt or purchase of this document, the recipient or purchaser agrees not to reproduce or copy it by any means whatsoever, nor to permit such action by others, for any purpose without prior written permission from Sperry Univac.

Sperry Univac is a division of Sperry Corporation.

This document was prepared by Systems Publications using the SPERRY UNIVAC UTS 400 Text Editor. It was printed and distributed by the Customer Information Distribution Center (CIDC), 555 Henderson Rd., King of Prussia, Pa., 19406.

you don't have to be a programmer

The *B*eginner's *A*ll-purpose *S*ymbolic *I*nstruction *C*ode – BASIC – is an easy-to-use programming language you can learn in just an hour or two, even if you've never programmed before.

But don't be misled by the first word in BASIC – Beginner's. While schools and colleges use BASIC extensively to teach programming, it has all the power and flexibility found in sophisticated languages used by computer professionals. That's why engineers, scientists, and businessmen use BASIC every day as their language to solve a wide variety of problems.

Why is BASIC so popular with nonprogrammers? Because BASIC:

- is easy to learn and remember;
- handles both routine business applications and complex mathematical and scientific problems; and
- uses simple English-like statements and familiar mathematical expressions.

But BASIC is more than just a programming language. It's also a programming environment that:

- lets you write, change, and run your program at an interactive terminal located where you work – no waiting for results;
- tells you immediately when you make a mistake and helps you correct it; and
- makes it simple to save your programs for future use and to retrieve them when you want them.

easy to use

Sit down at a workstation, identify yourself to the system with the LOGON command, bring up the BASIC compiler with the EXEC command, and you're ready to write your BASIC program.

```
LOGON ENGHLM
EXEC BASIC
OS/3 BASIC READY  BEGIN:
```

Let's say that you want a program that will calculate the area of a circle, with you supplying the radius at the time you run the program. You start by typing in a line number and then the first line of the program.

```
10 PRINT PROGRAM TO COMPUTE AREA OF A CIRCLE"
```

You forgot to type quotation marks before the word PROGRAM, so BASIC comes back with a question mark and displays your line up to where you made the mistake. To fix the line, just retype it correctly.

```
 10 PRINT PROGRAM TO COMPUTE AREA OF A CIRCLE"
?10 PRINT
 10 PRINT "PROGRAM TO COMPUTE AREA OF A CIRCLE"
```

When you run the program, you'll want to type in a number for the radius. So you add an INPUT statement. INPUT R means the program is to read input (the radius) into location R.

```
 10 PRINT PROGRAM TO COMPUTE AREA OF A CIRCLE"
?10 PRINT
 10 PRINT "PROGRAM TO COMPUTE AREA OF A CIRCLE"
 20 INPUT R
```

But now you realize that you left out a step. You will want the program to ask for the radius. No problem! You just enter a PRINT statement with a line number in the proper sequence.

```
 10 PRINT PROGRAM TO COMPUTE AREA OF A CIRCLE"
?10 PRINT
 10 PRINT "PROGRAM TO COMPUTE AREA OF A CIRCLE"
 20 INPUT R
 15 PRINT "PLEASE ENTER RADIUS OF CIRCLE"
```

Add three more lines to compute the area of the circle, print out the answer, and end the program. Then type in the RUN command.

```
 10 PRINT PROGRAM TO COMPUTE AREA OF A CIRCLE"
?10 PRINT
 10 PRINT "PROGRAM TO COMPUTE AREA OF A CIRCLE"
 20 INPUT R
 15 PRINT "PLEASE ENTER RADIUS OF CIRCLE"
 30 A = 3.14159*R↑2
 40 PRINT "AREA OF CIRCLE IS ";A
 50 END
 RUN
```

BASIC runs your program and gives you the results immediately. At the INPUT statement, the program will stop to let you enter the radius. Here's the output from this program.

```
 PROGRAM TO COMPUTE AREA OF A CIRCLE
 PLEASE ENTER RADIUS OF CIRCLE
?17.4
 AREA OF CIRCLE IS 951.147
```

powerful computational tool

It's easy to solve computational problems with BASIC, because BASIC uses familiar mathematical expressions and follows normal arithmetic rules. For instance, the BASIC expression

$$A + B - C/D$$

is equivalent to the algebraic expression

$$a + b - \frac{c}{d}$$

BASIC uses these symbols in arithmetic statements:

Operation	Symbol
Addition	+
Subtraction	-
Multiplication	*
Division	/
Exponentiation	↑

The symbols for multiplication and exponentiation are different from the arithmetic symbols, because x for multiplication could be confused with the letter X and raised numerals for exponentiation are meaningless to a computer.

You use parentheses in your BASIC statements the same way you use them in math - to show that the operation inside the parentheses is performed before the rest of the statement. The BASIC statement

$$Y = (N * M) \uparrow 2$$

means

$$y = (nm)^2.$$

BASIC has a large library of built-in mathematical and trigonometric functions that you can use in your program by simply writing them in an abbreviated form in your BASIC statements. For instance, the statement

$$Y = SQR\ (X + 2)$$

means

$$y = \sqrt{x+2}$$

If you want to write very large or very small numbers in your BASIC program, you can use either ordinary decimal numbers or a special notation similar to that used by scientists. The number 460,000,000 or 4.6×10^8 may be written in BASIC as

```
4.6E + 8
```

The E + 8 means that 4.6 is multiplied by 10^8. Similarly, the number .000000046 may be written in BASIC as

```
4.6E - 8
```

The E - 8 means that 4.6 is divided by 10^8.

a lesson in BASIC

The computational prowess of BASIC makes it a natural for solving scientific problems. But BASIC is equally adept with routine business applications. This easy-to-follow program calculates the amount of money accumulated at the end of 5 years on a principal of $4000 at 5 percent interest, compounded daily.

```
10 DATA 4000.00,0.05,5,365
20 READ P,R,N,C
30 LET A=P*(1+R/C)↑(N*C)
40 PRINT "AMOUNT AT END OF 5 YEARS ";A
50 END

RUN

AMOUNT AT END OF 5 YEARS 5136.014
```

The letters in the READ statements are variables containing the principal, rate of interest, number of years, and number of times compounded per year (in this case, 365). The DATA statement enters values for those variables into the program in the order in which they appear in the READ statement. The LET statement performs the required computation. The PRINT statement displays the result of the calculation along with an explanation, and the END statement ends the program.

You can easily run this program with any values for the principal, interest rate, length of time, and compounding period by simply replacing the DATA statement with a new one. But there's another way you can change the values in your program every time you run it, and you don't have to keep changing the program. You use an INPUT statement.

The INPUT statement lets you enter data from the workstation interactively when you run your program. When the program reaches an INPUT statement, it waits for you to enter the data. If you place PRINT statements just before the INPUT statement, you'll get a message asking you to enter the data at the right time. Here's the same program written with an INPUT statement:

```
10 PRINT "ENTER VALUES FOR PRINCIPAL, INTEREST RATE,"
20 PRINT "NUMBER OF YEARS, AND COMPOUNDING RATE"
30 INPUT P,R,N,C
40 LET A=P*(1+R/C)↑(N*C)
50 PRINT "AMOUNT AT END OF ";N;"YEARS";A
60 END

RUN

ENTER VALUES FOR PRINCIPAL, INTEREST RATE,
NUMBER OF YEARS, AND COMPOUNDING RATE
?4000.00,5.25,5,365
AMOUNT AT END OF 5 YEARS 5200.608
```

So far, we've concentrated on the computational capabilities of BASIC. But BASIC can do non-numeric problem-solving just as easily. In fact, BASIC is especially suited to creating interactive question-and-answer sessions.

In our interest-computing example, suppose you wanted to make it easier on the person running the program by asking for the compounding period instead of the compounding rate. You want your program to print out something like this:

```
HOW DO YOU WANT THE INTEREST COMPOUNDED?
TYPE IN A FOR ANNUALLY, S FOR SEMIANNUALLY,
Q FOR QUARTERLY, OR D FOR DAILY
```

Your program, of course, will have to convert the response entered at the workstation to the compounding rate (for example, 365 for D) needed in the compound interest formula, but BASIC can handle the extra programming steps with ease.

more than a language

BASIC is more than a programming language – it is a complete programming system. BASIC has its own built-in editor that lets you build your program at the workstation, run it, and save it for future use. When you want to use the program later, BASIC gets it out of the file so you can update it or rerun it. You never concern yourself with job control statements, setting up files and accessing them, or the like. BASIC does it all for you.

When you build your program, you precede each statement with a line number. The line number tells BASIC that the statement is part of your program and also places it in the proper sequence. If you want to insert or change a line in your program, the line number tells BASIC where to insert the statement or which line to replace.

```
10 PRINT "PROGRAM TO COMPUTE AREA OF A CIRCLE"
20 INPUT R
15 PRINT "PLEASE ENTER RADIUS OF CIRCLE"
```

As you type each line, BASIC checks it for syntax errors, like leaving out the initial quotation marks, as shown in the following example. If BASIC finds an error, it responds with a question mark followed by the incorrect statement up to the error. If you forget to type in the line number, you just get a question mark. You then type in the correct statement.

```
10 PRINT PROGRAM TO COMPUTE AREA OF A CIRCLE"
?10 PRINT
10 PRINT "PROGRAM TO COMPUTE AREA OF A CIRCLE"
INPUT R
?
20 INPUT R
```

Suppose you have made several corrections in your program and you want to see a clean copy of what you have written so far, in proper line order. Enter the LIST command.

```
LIST
10 PRINT "PROGRAM TO COMPUTE AREA OF A CIRCLE"
15 PRINT "PLEASE ENTER RADIUS OF CIRCLE"
20 INPUT R
```

Commands like LIST and RUN are not part of your program and are never preceded by line numbers. BASIC commands let you:

- name your program;
- modify the statements in your program;
- execute your program;
- save your program;
- recall a saved program so you can update or rerun it; and
- end the BASIC session.

the language you need

BASIC is the language you need when you have a problem to solve, but don't have the skills of a professional programmer. It's the language you need when you want to build a program right at the workstation and see the results immediately. It's the language you need when you want a program written quickly and inexpensively. It's the language you need when you want a program that works interactively – one that lets you enter data from the workstation while the program is running and gives you different results each time on the basis of that data. BASIC is one of the simplest of all programming languages, yet it is a powerful and flexible language.

BASIC is the language you need when you want more than a language – BASIC offers you a complete programming system that edits, compiles, executes, and stores your program for you.