Curriculum Units by Fellows of the Yale-New Haven Teachers Institute
1981 Volume VI: Computing

The Basics of BASIC

Curriculum Unit 81.06.05
by Kathleen M. Huhner

I should first like to discuss the prerequisites of the course, in an attempt to give a little background. In order to take this course, a student must be a sophomore (though freshmen in the advanced sequence have been accepted) and must have passed a course higher than Applied Math. In addition to this the student must take a regular mathematics course, unless he has completed the two years required by the state, These conditions were stated to avoid some of the difficulties that would not be desirable while developing a new course.

The course work deals mainly with the Radio Shack TRS-80 Level II and Level III computers. Programs are written by the on the computers. Corrections are then is repeated. The class meets five days a week, as is done for major subjects.

The texts used in the course are many, but mainly they are *BASIC: An Introduction to Computer Programming Using the BASIC Language* by William Sharpe and Nancy Jacob and *Computer Pro* g *ramming* in the *BASIC Language* by Neal Golden.

The opinion of the students is that Sharpe and Jacob's book is more useful, at least for the introduction to BASIC.

I propose the following be a tentative schedule for BASIC beginners (you might find that you spend a larger (or smaller) amount of time on any given topic):

Week 1: I, Diagnostic Test

II. Digital and Analog Computers
III. How Digital Computers Work
  A. Communication with the computer
  B. Parts of the Computer

Weeks 2 & 3: I. Programming the Computer

- A. Absolute machine language
- B. Assembly language
- C. Source language
- D. Object language

II. The Programming Capability of BASIC
III. Field Trip to a Computer Center
IV. Flowcharting
V, Logging On and 0ff the Computer
VI. Test on Flowcharting

Week 4: The Basic Vocabulary of BASIC

I. PRINT (handles output), END (last statement)
II. Operations Hierarchy in BASIC
III. RUN (executes all statements in memory), SCR (erases a program from computer memory), LIST (causes the computer to type out all the statements in its memory)
IV. LET (used for computing and storing numbers)
V. Rules Regarding Variables
VI, INPUT (handles input)

Weeks 5 & 6: Data and Transfer of Control

I. READ (names variables; reads in data), DATA (stores values for variables in READ statement)
II. GO TO (unconditional)

III. IF. . .THEN (conditional), STOP (causes the computer to cease the execution of a program)

Weeks 7, 8, & 9: Loops and Library functions

I. FOR. . .NEXT
II. FOR. . .NEXT. . .STEP
III. Nested Loops
IV. DIM (used for subscripted variables)
V. REM (remark statement)
VI. Library Functions in BASIC (SQR, INT, ABS, RND)

The course begins with a diagnostic test. The test measures the student's level of mathematical ability and affords the teacher the opportunity to see how diverse (mathematically) the students are. The following few days are spent *on an* introduction to what a computer is, what physical units it consists of, and mainly what it is not. This is followed by (time and money allowing) a trip to the Yale Computer Center or to the University of New Haven Computer Center to see a machine in operation. By this time the students have begun their study of flowcharting.

It seems to begin the study of algorithms and programming by using a flowchart—a step-by-step outline of the procedure to be followed by the computer. The structure of algorithms can be seen more effectively in flowcharts than in any other representation. The following symbols are used by the students:

Symbols of the Logical Flowchart

(figure available in print form)
The Oval is used to signal the beginning and ending of a flowchart.
(figure available in print form)
The Parallelogram is used to designate input or output.
(figure available in print form)
The Rectangle is used for arithmetic assignments.
(figure available in print form)
The Diamond is used for decisions.
(figure available in print form)
The Circle acts as a connector between one portion of a flowchart and another.
(figure available in print form)
The Arrows show the flow of the steps of the flowchart.

Once the students master the symbols and have seen some examples, they are given their first assignment: Flowchart your actions from the time you awaken until you board the bus for school. The following day the students compare their flowcharts. Through discussion and comparison, they soon understand the properties and structures of algorithms and are able to select (and appreciate) a good algorithm in preference to a poor one.

Problems are introduced that benefit from being programmed and run on the computer. Hence it is a logical step to incorporate problems which involve mathematical notation and BASIC commands into the flowcharts (see Appendix A, example 1). The logic of each flowchart should be followed and tested using a trace (see Appendix A, example 2).

Some flowchart assignments I have given to the students are:

1. Read two real numbers; multiply them and divide the result by two.
2. Compute the (a) perimeter of a triangle and (b) area of the triangle (using Hero's formula) given the lengths of the sides.
3. Read a real number X, If X is nonnegative, give its square root. If X is negative, print "NO REAL SQUARE ROOT."
4. Compute an employee's weekly wage, given his rate-per-hour and the number of hours worked. (Assume that the employee receives time and a half for overtime).

Again, students compare their flowcharts in class. They also perform traces and rectify any errors or omissions.

Beginning with his section and continuing throughout the rest of the course, students learn how to write programs in the BASIC language. The flowchart assignments were chosen with BASIC in mind, so students should find the transition from flowcharting to programming to be relatively simple. For example, study the BASIC program beside its corresponding flowchart on the following page.

Students now go to the computer room and run their programs. As you can well imagine, the students watch—with considerable excitement- their own programs run. From then on, students use the computer room more and more. They seem to enjoy the busy atmosphere of the place which teems with students, By the middle of October, students may begin using their study halls and free periods to get in runs of their programming assignments.

*Note: If the students run the following program, they soon find that the computer calculates and prints three areas. The fourth time it returns to the READ statement, it finds there are no more numbers in the data list and prints OD 10 (out of data in line 10). It then terminates execution of the program.

PROGRAM AND FLOWCHART TO CALCULATE AND PRINT THE AREAS OF RECTANGLES

(figure available in print form)

(figure available in print form)

10 READ L, W

(figure available in print form)

20 LET A = L * W

(figure available in print form)

30 PRINT “AREA =”;A

(figure available in print form)

40 GO TO 10

(figure available in print form)

50 DATA 33,5,7,9,2.8,1.1

(figure available in print form)

60 END

Data to be used: 33,5, 7, 9, 2.8, 1.1

Until now, the teacher and students have been working together. The teacher has given assignments, has discussed them with the students, and has given them a solid understanding of flowcharting and eight commands in BASIC. A test should be given. It should test the students’ knowledge of flowcharting, error detection, and programming abilities. (Appendix B contains a sample test for this purpose.) The students should be graded on their logic and programming efficiency. As soon as the tests are returned, the flowcharts should be discussed with the class.

Students should now feel comfortable with the ideas of flowcharting and programming. They should be able to analyze and construct a program from a flowchart (and vice versa), and they should be able to write and process a computer program utilizing everything contained in the schedule weeks two through six (part II).

At this point, students are ready for an introduction to loops. Looping is an important and powerful technique. As such, they deserve careful study. The diagram below provides the essential components of any loop.

(figure available in print form)

These components may occur in a different order (except the initialization), but the, must all be present in a loop, Also, note that EXIT does not necessarily imply STOP—when a loop is completed, the computer exits to the first step (after the loop) that can be executed, To make sense out of the diagram on the preceding page, look at the following programs (each component is identified as it occurs),

Write a program to print the squares of 4,6,8, and 10.

10 LET X=4	Initialization
20 PRINT “THE SQUARE OF”;X;”IS”;XI2	Body
30 LET X=X+2	Incrementation
40 IF X > 10 THEN 60	Stopping Mechanism
50 GO TO 20	Re-entry of loop
60 END	Exit

Write a program to read five test scores and print the average of these scores.

```
10 LET Z=0

 20 LET I=1                        Initialization
 30 READ T                         Body
 40 LET Z=Z+T                      Body
 50 IF I=5 THEN 80                 Stopping Mechanism
 60 LET I=I+1                      Incrementation
 70 GO TO 30                       Re-entry of loop
 80 PRINT “THE AVERAGE IS”;Z/5 Exit
90 DATA 97,85,63,76,81

100 END
```

Since an iterative process is common, computer languages simplify it. The students should now learn a pair of commands to accomplish the same tasks as the programs above—the FOR/NEXT loop.

The FOR statement contains the index, the initial value, the increment value, and the final (or test) value, The NEXT statement contains the increment step, the test step, and the re-entry step. For example, the second program above can be compressed into:

```
10 LET Z-O

20 FOR I=1 TO 5

30 READ T

40 LET Z=Z+T

50 NEXT I
```

The rest of the program remains the same.

It should be stressed that the FOR/NEXT loop does not apply to indefinite loops (e.g. to count a sequence of numbers of unknown length)—it is used only when you know how many times to repeat a loop.

A test should be given to the students. A few of my favorite problems follow.

1. Given any three numbers, in any order, print the numbers in decreasing order.
2. Manhattan Island was purchased in 1626 for $24. If those early buyers had invested the same amount in certificates of deposit (at 17 1/2% interest), how much would their investment be worth today?
3. As shipping clerk you are responsible for shipping quantities of products as cheaply as possible. It is generally less expensive to send 50 items in one large box than 25 in each of 2 boxes. Read in the following data:
    A. Stock number (SN)

B. Number of items to be shipped (IS)
C. Maximum number of items that can be contained in
I. the largest box (LB)
II. the middle-sized box (MB)
III. the smallest box (SB)
It is understood that LB MB SB, and that there may be only one or two sizes of boxes.
For each set of data, write a line containing
A. Stock number
8. N1 = the number of boxes of LB
C. N2 = the number of boxes of MB
D. N3 = the number of boxes of SB
Your program should place as many of the items as possible in the largest number of the large boxes, then place as many {or any remaining) items in the smallest box in which they will fit.

I have found that students should make a copy of their programs. Then, they take their copies, debug their programs, and get them to run. In this manner, the students become aware of certain errors to avoid.

A natural result of discussion of the FOR/NEXT is the discussion of subscripted variables. Subscripted variables have a wide variety of programming applications. For instance, a student wishes to sort a list of numbers. Before entering a list of numbers, a DIM statement must be used to set aside space in the memory for each list to be used. For example,

```
10 DIM X(100)
```

Secondly, the student must read in the amount of numbers in the array (not to exceed 100 for this example). This can be accomplished as follows:

```
20 READ N
```

```
30 FOR I=1 TO N

40 READ X(I)

50 NEXT I

60 DATA .,. . . .. . . ..
```

Thirdly, a nested loop (a loop within a loop) is necessary in order to perform the sort. The partial program below shows how to sort an array into descending order.

```
80 REM THE OUTER LOOP

BEGINS

90 FOR I=1 TO N-1

100 REM THE INNER LOOP

BEGINS

110 FOR J=I+1 TO N

120 IF X(I) =X(J) THEN 160

130 LET Z=X(J)

140 LET X(J)=X(I)

150 LET X(I)=Z

160 NEXT J

17O REM THE INNER LOOP ENDS

180 NEXT I

190 REM THE OUTER LOOP ENDS
```

Lines 120 through 140 pull a “switch.” These lines are a swapping mechanism—a dummy location (Z) is needed in order to keep the value of X(J). (Without the “dummy” X(J)=X(I) at the end of the loop.)

Locating the real roots of a polynomial are handled in programming by means of subscripted variables, where we think of a polynomial as an array containing the coefficients in ascending powers of X. Here are some examples:

POLYNOMIAL	ARRAY
$5x^2 + 6x^4$ Ð$3x^5$	0 0 5 0 6 -3
94x	0 94
$5-6x^3 + 4x$	5 4 0 -6

Notice in the first and third examples that 0's are used in the array for missing powers of the variable. An assignment should be given to the students once they feel comfortable with the idea of subscripted variables—two of my pet problems follow.

1. The Fibonacci sequence begins 1,1,2,3,5,8,13, . . ., where each term from the third on is the sum of the two preceding terms.
   (a) Given A and B, print all terms of the Fibonacci sequence that lie between A and B.
   (b) Find N such that the sum of the first N terms of the Fibonacci sequence exceeds 10 7.
2. (a) Write a program that creates a file which contains a list of N student numbers and test scores. There are five scores for each student. The computer should input student numbers and scores until "9999" is typed as the student number.
   (b) Write a program that uses the file of (a) to type out a table with each student's number and his average score for the five tests.
   (c) Write a program that uses the data of (a) to type out the number of the student who received the highest score on each test and the number of the student who received the lowest score (Remember the number 9999 tells the computer to stop looking.).

Thus, the introduction to BASIC ends.

Unfortunately, I haven't the space to complete a curriculum guide for my entire programming course. I would like to mention that the second and third marking periods are concerned with Advanced BASIC. Students are taught the following: matrix manipulation, setting up, storing, and using files, statistics, plotting (on the CRT and line printer), and game and business simulations.

The fourth marking period is spent on the final project. This consists of any game which the student wishes to program. For examples of what my students have become capable of doing by this time, I shall mention some of their projects. One student wrote an impressive chess program for two players (try to make an illegal move and see what happens!); another wrote a "connect the dots" program (don't try to cheat on this one—your lines will disappear and you'll see spots before your eyes!); another wrote a Bingo game (complete with "blackout"); still another wrote a dog racing game (complete with bets and dogs running across the CRT). I think though, by far, the most impressive program was written by one of my freshmen—an adventure game (hopefully you will make it through and locate the hidden treasure).

All of the projects were programmed in BASIC and all of the students (except two) had little or no knowledge of programming when they began the course. I have incorporated all of the material, which I have presented, for the past two years and have met with success.

APPENDIX A

EXAMPLE 1: Compute the average of three numbers.

(figure available in print form)

(figure available in print form)

The programmer chooses X, Y Z three letters (X,Y,&Z) to represent the numbers to be averaged.

(figure available in print form)

The computing step uses the standard formula for finding an average, the programmer electing to call the answer “A.”

(figure available in print form)

(figure available in print form)

EXAMPLE 2: Trace the following flowchart for the data values listed.

(figure available in print form)

(See actual trace on following page)

1. Trace: Input 8 for X and 2 for Y

(figure available in print form)

2.Trace: Input 10 for X and -5 for Y

(figure available in print form)

APPENDIX B

1. Sample Test following weeks 4 through 6.

I. Inspect the following program. What errors will be detected when the program is run?

```
10 READ X,Y

20 LET X+2=Y-3

30 LET Z=X+Y-X

40 PRINT “ANSWER Z-Y

50 GO TO 80

60 LET I=(Y+X)(Y-X)

70 GO 50
```

II. Construct a flowchart and write a program for one of the following problems.

1. Calculate a man’s gross weekly pay given his rate-per-hour, his overtime rate-per-hour, and the number of hours that he worked that week,

2. Compute the perimeter of a right triangle given the length of the hypotenuse and the length of one leg.

III. will the output look like?

1.
```
50 LET I-1
55 PRINT I;
60 IF I> 12 THEN 90
70 LET I = I+1
80 GO TO 55
90 PRINT "          END"
100 END
```
2.
```
5 READ X
10 READ Y
15 READ Z
20 READ Z2
25 PRINT X,Y,X*Y,Z, Z2,Z2-Z
30 GO TO 5
35 DATA 5,2.5,40,2.97,37,2,3.1,75
40 END
```

Annotated Bibliography for Students

Dwyer, Thomas A. and Margaret Critchfield. *Computer Resource Book Algebra* , Boston: Houghton Mifflin Co., 1975. Provides a collection of interesting algebraic problems to be programmed and run. May be used as a supplement to an Algebra course.

Dwyer, Thomas A. and Michael S. Kaufman. *A Guided Tour of Compute* r *Programming in BASIC* . Boston: Houghton Mifflin Co., 1977. A good introductory programming book for remedial classes. Contains good review sections.

My Computer Understands Me When I Speak BASIC . Menlo Park, Ca.: DYMAX, 1971

An excellent book for remedial classes. Needs supplementing.

Gottfried, Byron S. *Theory and Problems of Pro* g *ramming with BASIC.* New York: McGraw-Hill Book Co., 1975. A good reference for students. Contains answers and commentaries. Also contains many good business applications.

Schoman, Kenneth E. *The BASIC Workbook* . Rochelle Park, N.J.: Hayden Book Co., Inc., 1977.

A limited vocabulary of BASIC (22 words). May be used as a supplement to a BASIC course. Contains interesting problems in Mathematics.

Annotated Bibliography for Teachers

Jacobs, Zeney P., Francis G. French, William J. Moulds, and Jacob G. Schuchman. *Computer Programming in the BASIC Language* . Boston: Allyn and Bacon, Inc. 1978. A good book containing many business applications.

Simon, David E. *BASIC From the Ground Up* . Rochelle Park: Hayden Book Co., 1978.

Assumes no prior knowledge of computers. Good examples contained in chapters and an excellent glossary.

Tanenbaum, Andrew S. *Structured Computer Organization* . Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1976. Assembly language programming and exercises.

Vazsonyi, Andrew. *Introduction to Electronic Data Processing* . Homewood, Il.: Richard D. Irwin, Inc., 1977. Covers data processing, programming approaches, and sample case studies.

Magazines

Calculators/Computers Magazine. DYMAX, P.O.Box 310, Dept.22, Menlo Park, Ca. 94025.

Creative Computing, P.O. Box 789-M, Morristown, N.J. 07960.

Recreational Computing, 1263 El Camino Real, Box E, Menlo Park Ca. 94025

Annotated List of Materials for Classroom Use

Golden, Neal. *Computer Programming in the BASIC Language* . New York: Harcourt Brace Jovanovich, Inc., 1975. Contains over 500 problems. Covers Algebra I through Analytic Geometry, physics, statistics, and business. An excellent book for course use.

Sharpe, William F. and Nancy L. Jacob. *BASIC : An Introduction to Computer Programming Using the BASIC Language* . New York: The Free Press, 1971.

Part I gives essential features of BASIC. Part II provides procedures useful to mathematicians and scientists. Contains problems and solutions. An excellent book for an introductory course.

Radio Shack Reference Books for TRS-80

BASIC Computer Language

Includes programmed instruction and user programs.

Level II Programming

Examples of Level II BASIC commands and routines for the TRS-80 Model II.

Level III Programming

Examples of Level III BASIC routines.

TRS-80 Line Printer

How to handle output on the line printer. From tables to “fat letters.”

TRS-80 Graphics

How to create graphic displays on the CRT. Includes animation techniques.