LOOP Center, Inc.

A Curriculum Guide for Teaching BASIC

digital

A Curriculum Guide For Teaching BASIC is part of the BASIC Programming Curriculum Package ($50). This book is also available separately for $4.00 plus .50 postage and handling from:

Software Distribution Center
Digital Equipment Corporation
Maynard, Massachusetts 01754

For information regarding other DIGITAL curriculum publications, obtain a copy of the Curriculum Material Product Catalog from:

Communication Services
Digital Equipment Corporation
Maynard, Massachusetts 01754

1st Printing -- March 1974

DEC-08-ECGBA-A-D

acknowledgements

Special thanks to Pamela H. Ellsworth of Project LOCAL, Inc., for her help and experience which made much of this guide possible.

contents

introduction

This guide is designed to provide assistance in establishing a successful course of instruction in BASIC programming. The recommendations and ideas discussed do not exhaust all possible approaches that may be implemented. They are, however, based on successfully implemented techniques derived from actual classroom experiences.

The guide examines facility planning, instructional prerequisites, ideas for classroom implementation, and three methods of teaching BASIC programming. In addition, a topical index has been included for easy reference to additional discussions, assignments, and projects contained in major BASIC texts. The Appendix lists additional materials which may be obtained directly from the publishers.

It is highly recommended that for most successful results, this guide be used in conjunction with the other elements of the BASIC Programming Curriculum Package. The contents of this package include:

- Populution: A Self-Teaching BASIC Primer; DIGITAL
- EduSystem Handbook; DIGITAL
- 101 BASIC Computer Games edited by D.H. Ahl; DIGITAL
- Getting Started in Classroom Computing by D.H. Ahl; DIGITAL
- 23 Slides For Teaching BASIC (with teacher's guide); DIGITAL
- A Short Course in BASIC by P.H. Ellsworth, Project LOCAL, Inc; DIGITAL
- A Curriculum Guide for Teaching BASIC by S.R. Bower; DIGITAL
- Bibliography of Texts and Resources on the Uses of Computers in Education; DIGITAL

For additional information regarding this package and other resources consult the DIGITAL Curriculum Material Product Catalog.

before you begin

A successful program of BASIC language instruction requires careful planning. Before beginning the task of establishing a programming curriculum, several things should be considered:

1. The capabilities and limitations of the school computer facility must be understood.
2. Students and faculty must be introduced to and be confident about using the computer.
3. The background knowledge of computer fundamentals must be stressed.
4. Classroom procedures and methods should be established.

Each of these considerations are discussed in this guide and recommendations are made based on successful implementations in classroom environments.

the computer facility

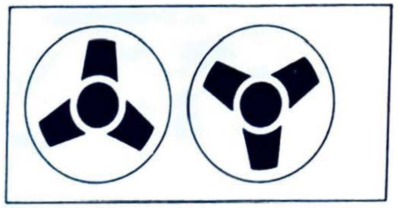
The size and the location of the school computer facility must be determined and an installation plan established before a successful program of instruction may be implemented. The physical size of the computer often determines its location. For example, a small facility consisting of terminals and a small minicomputer may be located in the classroom (classroom laboratory), whereas larger computer systems are often situated in a laboratory or a room designed specifically for housing the equipment (computer laboratory). In some schools a combined approach is used. The classroom laboratory and the computer laboratory each create unique requirements and problems that must be understood before instruction begins.

The classroom laboratory creates a comfortable learning environment because the computer or terminal becomes a permanent and familiar classroom instructional tool. Its easy availability permits the computer to be referenced to illustrate concepts currently being studied. Should questions arise related to discussion of a programming topic, demonstration of examples can take place immediately. Easy access to the computer resource is particularly important where computer fundamentals and programming languages are being taught for the first time; unanticipated questions can be answered immediately using the computer rather than postponed for the next session.

The classroom laboratory arrangement facilitates the supervision of student activities. At the same time, it creates an active classroom environment and one that encourages student-student interaction and involvement. Generally, a class period will begin with discussion of a new topic in BASIC programming and terminate with student experimentation at the terminals.

This combination of classroom instruction and computer time, though mutually reinforcing experiences, may create some problems. Because the terminals are located in the classroom, students may not be able to use the facility outside of class, particularly where there are scheduling conflicts. In addition, the classroom laboratory approach may restrict the use of the resource by other members of the faculty. School-wide usage of the computer facility is a highly desirable goal for many schools; such physical restriction will almost certainly inhibit the growth of the school computer program and interfere with the goal of school-wide usage.

Restricted use of the computer resource has been eliminated in many schools by creating "mobile" terminals. By installing a few simple computer equipment options, and placing terminals on movable carts, the terminals can be transported to different classrooms where they are needed. Thus, the computer becomes a classroom tool, not unlike visual aid equipment. Scheduling of the resource to maximize usage is, of course, an important aspect which cannot be overlooked.



Portable terminals allow wider usage by the faculty and permit greater student usage during free class sessions. However, where the terminal is being used by students outside of the normal class period, faculty supervision must be considered.

Whether the classroom laboratory is permanent or portable, one additional consideration remains. Depending on the model of the terminal used, the noise factor may be important in establishing a schedule for computer use. A reasonably quiet terminal, such as a CRT (Cathode Ray Tube) terminal or DECwriter, may not interfere with concurrent classroom work, thus allowing students to work with the computer during class. However, if the terminal is of the "noisy" variety, restrictions must likely be placed on concurrent activity.

The computer laboratory should be situated in a central location for easy access by all departments within a school. It may be adjacent to the school library or in some cases, in a room specifically designed to house the facility. In either case, central location is critically important if a school-wide computer program is a desired goal.

The major advantage of the computer laboratory arrangement is that students may work independently of classroom instruction; advanced students are free to work in the laboratory during a portion or all of a given class period. As the class becomes competent in programming skills, the entire class may be scheduled for computer time during a class period. The separation of classroom instruction and computer time is desirable because it allows the teacher to conduct review sessions or to help students individually while other students work concurrently with the computer in the separate location.

Two schedules must be established for effective use of the computer laboratory facility; one for faculty supervision and another for student terminal time. If a data processing manager has been appointed, supervision of the facility by other faculty members is probably unnecessary.

Perhaps more important than supervision is the availability of a faculty resource person to answer student questions while the computer is in use. Close observation and discipline, interestingly, is often unnecessary in the computer laboratory. Students often take serious responsibility for the resource and work actively to protect it from misuse by their peers. Computer laboratories in many schools have proven to be the most active and healthy of learning environments; student conversations and interaction are, for the most part, based on exchange and cooperation-- a healthy and desirable atmosphere for learning! Consequently, the computer laboratory should be open for students' participation, whether they are, themselves, using the terminal or not! Some restriction, of course, must be placed on students to avoid massive crowds--a common problem faced in the computer laboratory!

The frequency of student-terminal contact is based on the duration of the programming course, the size of the computer facility, the number of students participating in the program and the complexity of the assignments that require computer use. Many schools have instituted a rigid schedule for computer use out of sheer necessity; where the physical size of the computer resource is small, this is often mandatory. The use of off-line terminals to prepare program paper tapes prior to on-line testing has been successfully implemented in schools that do not have a large computer facility.

If ample computer resource is available, however, a rigid schedule for student use should be avoided. Many schools have found that alloting terminal time on a "first-come, first-served" basis has proven effective. This policy, combined with a priority "schedule" given to specific assignments and a maximum per diem limit of computer use placed on each student, has produced satisfactory results. Plenty of time should remain for work on extra-curricular projects and experimentation with games, drills and simulations. The positive educational value of these latter experiences should not be overlooked!

Sign-up boards and ticket systems have been successfully established in some schools for solving the scheduling problem. Students sign up for a specific time period; penalties may be applied to those students who sign up but miss their segment. Normally, however, there will be enough "interested bystanders" to use that scheduled allotment of time!

In addition to the scheduling of terminal time, guidelines regarding typical terminal usage may be helpful. Student users can be classified into three broad categories:

CATEGORY	TERMINAL USAGE
Casual user	10 minutes/week
Average user	15-20 minutes/week
Computernik	30 minutes and above/week

At the outset, it is anticipated that a successful program of BASIC instruction requires one terminal to support 60 to 80 average users per week in public schools or 90 to 140 average users per week in private schools.[1]

When the instructional program becomes fully integrated into the existing curricula (generally two years from the outset of the program), one terminal is required to provide support for:

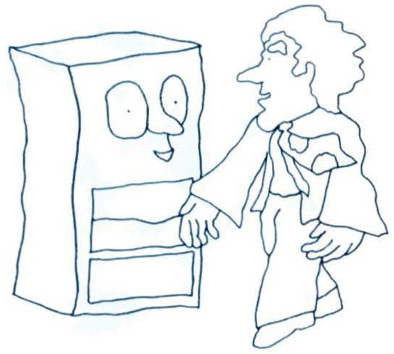
TYPE OF SCHOOL	NO. AVERAGE USERS	TERMINAL USAGE
Public	50	30-40 minutes/week (2 sessions)
Private	100	30-40 minutes/week (2 sessions)

A successfully implemented program of BASIC instruction eventually requires that each student has access to the computer terminal at least twice a week.

Whether a classroom or computer laboratory arrangement is chosen, the preceding considerations and recommendations should aid in laying the foundation for a successful program of computer instruction.

[1]Private school usage per terminal is greater than public school due to more hours of computer access.

introducing the computer



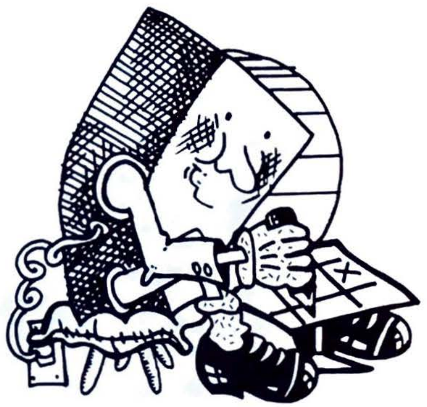
Perhaps the most difficult hurdle to overcome in learning anything new is the initial apprehension of dealing with the unfamiliar. For many students (and faculty members too!), communicating with the computer requires preparation, explanation, and, most important, experience. The introduction to computer use is not unlike a student's first experience with science laboratory equipment; once the initial contact is completed, future encounters are easier and confidence levels raised.

The first interaction between student and machine should take place at the outset of the BASIC programming course. Students who have some knowledge of how to operate the terminal find it easier to understand programming concepts as they are presented in class. Likewise, continued hands-on experience for the duration of the course is critical to total understanding of the course material.

The introduction to terminal use has been successfully achieved by using pre-written programs such as games, simulations, or drills and practice problems. Most students have played games like TIC-TAC-TOE or CHECKERS; students are generally challenged by the task of beating the computer at number games like GUESS and REVERSE. Many appropriate games can be found in 101 BASIC Computer Games, which is contained in the BASIC programming Curriculum Package. 101 BASIC Computer Games can be used in conjunction with Getting Started in Classroom Computing, which illustrates game playing without a computer resource; once the game is understood by students, the same game can be played on the computer. Role playing, discovery learning and other ideas for classroom presentations, discussed later in this guide, may also be used successfully at the introductory level.

If students are not mathematically oriented or the class's interests and abilities are highly diversified, simulations may be effectively used in the introduction to computer use. Of the Huntington II simulation programs that are currently available, POLUT and MARKET are the most applicable general interest programs. Other Huntington simulations may also be suitable for encouraging student-computer interaction as they require little background knowledge and can be "played" by individuals or groups of students. Simulations stimulate student inquiry and are easy to use--a combination which plays an important role in introducing the computer.

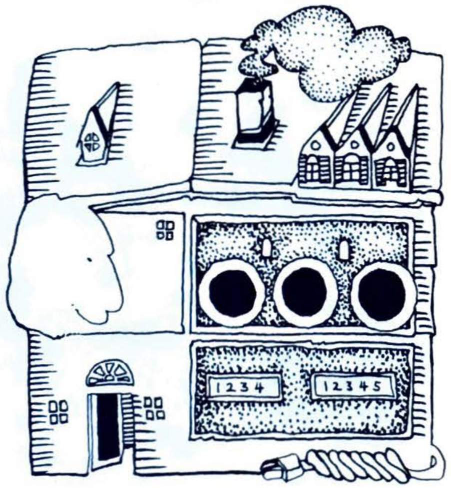
Some programming course introductions have used drill-and-practice programs instead of games and simulations. Regardless of the type of programs used, it is important that programs are easy to use, that students are familiar with the process involved in the program (e.g. TIC-TAC-TOE for games, addition skills for drill and practice, etc.) and that the program encourages students to learn more about the computer.



computer familiarization

In conjunction with the introduction to computer usage, additional background coursework may be required before beginning the study of the BASIC language. The structure and contents of a course in computer fundamentals or computer familiarization may vary depending on several factors: student level and ability, duration of the course, size of the computer installation, and level of faculty expertise. Generally, a computer familiarization course, designed as an introduction to BASIC programming, provides students with additional information about the capabilities and limitations of computers, as well as the roles that computers play in our daily lives.

Traditionally, computer familiarization has been considered by many educators as the least active, creative and challenging segment of computer introductory coursework. For this reason alone, in fact, familiarization has been omitted from many introductory programming courses. Although the topics themselves may be "unexciting," many of the classroom ideas presented later in this guide are applicable to computer familiarization and should be implemented! Understanding computer fundamentals is one very important factor in building an informed citizenry for the future.



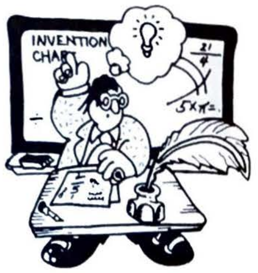
Computer familiarization generally includes some or all of the following topics. Recommendations are made which should enhance the presentation of these concepts. It is worthwhile to note at the outset that the "lecture" approach to these topics, though sometimes required, is not the only, nor is it always the best method of presentation.

Topics	Presentation Ideas
History of Computing	Peer-teaching Research projects Museum visits
Computing Concepts (Definitions: hardware, software, etc.) (Capabilities: speed, reliability, etc.) (Limitations: intelligence, etc.)	Research projects Experiments Role-playing Films
Computers in Society (Uses: business, government, education, industry) (Social issues: privacy, employment)	Field trips Debates Films Simulation
Career Opportunities	Field trips Guest speakers

Whenever possible, students should be actively involved in the learning process, whether through hands-on computer experimentation or through research projects and presentations. Increased motivation and interest will result from the "learning by doing" approach to computer fundamentals.

Additional ideas for coursework related to computer familiarization may be obtained from the Appendix and from the Bibliography of Texts and Resources For Computer Uses in Education which is contained in the BASIC Programming Curriculum Package.

classroom ideas



The strategies and methods implemented in BASIC programming courses have often been the keys to the success of those computer instructional programs. Although it is probably impossible to effectively use all the recommendations described in this section, selected options will aid in the success of new instructional programs. Some of the recommendations are discussed on the basis of topics taught in traditional programming courses, such as flowcharting and debugging; other suggestions and ideas have wide application to all facets of instruction.

Flowcharting

Flowcharting is sometimes taught as an introduction or as an integral part of an existing mathematics course, independent of computer use. In some schools flowcharting is not taught at all. If it is desirable to expose students to this method of analysis and organization of work as a part of the BASIC programming course, several guidelines are recommended.

Flowcharting symbols should be reduced to the minimal number required for efficient construction; typically, the symbols used include the terminal (⬭), connector (○), decision (◇), and process (□). Initial instruction should illustrate a simple process such as going to the store, or getting up in the morning. Flowcharts should never be stated in computer language terms, but rather in terms of problematic or procedural language.

Frequently flowcharts have been required of students as a normal part of program documentation. Correction and program transferability are facilitated when flowcharts accompany programming work. In addition, the student has a completely documented program to retain for future review or reference.

Many programming courses have required students to flowchart problems initially; as the students become more competent, the requirement is eased so not to impede progress. Flowcharting has also been an effective tool to aid students in planning and logical analysis prior to writing a BASIC program.

Debugging

Debugging a program or correcting syntax and logical program errors and retesting a program, has proven, in many cases, to be the most challenging and valuable programming learning experience. In fact, a student often gains more from debugging a program than from writing it! The debugging experience is analogous to the activity of taking a test, discovering the incorrect answers, correcting those answers, and submitting the test for a second correction! And maybe a third.

Debugging techniques and BASIC language debugging aids are discussed fully in <u>A Short Course in BASIC</u> which is contained in the BASIC Programming Curriculum Package. However, some additional comments are worthwhile.

Students should be encouraged to discover their own mistakes, and to experiment with the computer to achieve a fully operational program. Teacher aid should be given only when the student has given sufficient effort to the problem. Student-student interaction should be encouraged, especially for debugging tasks. Not only does student interaction release the teacher to handle the more difficult problems and reviews, but students learn from others' mistakes as well as from their own!

Syntax errors are the most basic errors and are generally easy to discover. For example, if a student misspells PRINT in a program line, a quick check of that line readily reveals the error. Computer diagnostics or error messages also point out this type of error to students.

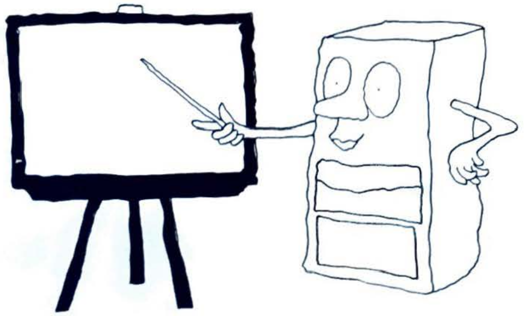
Logical errors are more difficult, sometimes nearly impossible (it seems!) to find. It is a good practice for students to check their written programs against their flowcharts to ensure that the process is fundamentally correct and that it has been properly translated into BASIC. Hand execution of a program often reveals a logical program error; thus, it is highly recommended that students be introduced to this procedure.

Classroom Demonstrations

Many methods of presenting classroom examples, illustrations, and demonstrations have been used in BASIC programming instruction. Different and equally creative strategies have been used for teaching BASIC language elements and concepts, writing programs, and illustrating the output of a program. Several of these practices are discussed and recommended for incorporation into BASIC courses.

In teaching BASIC programming concepts, particularly the elements of the language, examples have traditionally been used that incorporate the newly introduced language statement in a sample program. This exercise typically is executed at the blackboard with encouragement given for student participation. Although this approach is often sufficient, other methods are equally, if not more, effective.

The "discovery" approach is an example of an alternative illustrative method. Students are given a worksheet containing simple BASIC programs; new BASIC statements are incorporated into these programs. Using the computer, students run the program in order to discover the operation of the statement. This discovery approach is particularly useful when coupled with hand execution of the program; the computer solution is then compared to the students' calculations! The topics in BASIC programming particularly appropriate for this approach include order of operations, special functions (SQR, SIN, etc.), BASIC commands (LIST, SCR, DELETE, etc.), and programming techniques (round off, looping, ordering numbers, etc.)



A number of alternative methods are also possible for presentations related to program writing. Students may write program assignments at home or outside of class. However, significant learning has taken place in classrooms where students cooperatively "create" a program. Group programs can be written with or without a flowchart model which the class has constructed. Once written, the program can be run and debugged by the entire class. Alternately, programs can be designed with intentional errors to give students programming and debugging experience.

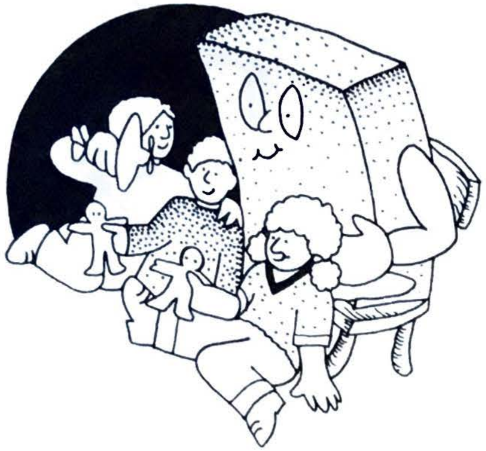
The classroom display of program output can be achieved in several ways. Some CRT terminals, such as the DECterminal, are designed to drive standard television monitors available from many school A-V departments. Classroom demonstrations are simple and effective where such equipment is available.

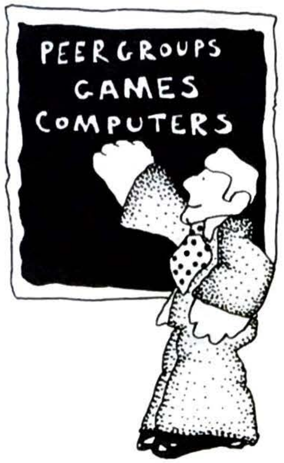
However, several alternatives are possible with hard copy terminals such as teletypewriters and DECwriters. Some schools have implemented the use of opaque projectors to display program output to the entire class. As the computer prints its results on the terminal, the paper is transferred to the projector so that students essentially receive immediate feedback of results. Another method involves the use of Ditto masters; terminal paper may be replaced with a master and program results recorded. Each student then receives a copy of the output for his own records.

Activities, Projects, and Student Involvement

Many of the creative classroom ideas recommended in this section have been successfully implemented in all areas of school computer instructional programs. None of the ideas or strategies are restricted to use in BASIC programming; rather, all provide fresh ideas that may be integrated into BASIC coursework.

Role playing has been used with excellent results (according to both teachers and students) from grade school children to university engineering and business management students. It is a particularly good approach when applied to computer fundamentals such as computer operation and architecture. Role playing is also applicable to program execution and the function of BASIC language elements. Getting Started in Classroom Computing illustrates one of the many ways that this approach can be used in classroom instruction.





Peer teaching has provided challenge and motivation in many classrooms; it is particularly appropriate when student ability levels are varied, and classes are relatively large. Student involvement in teaching creates a healthy environment for learning and interaction; students tend to take on a major share of responsibility for their own learning and for the achievement of other students. The teacher becomes a resource, a guide--and is free to deal with individual problems on a regular basis.

Student involvement can be utilized in several different ways: students act as tutors, helping others with programming problems; they research topics and make presentations to the class; they "control" the computer facility and are responsible for computer operation. In any case, valuable learning takes place while students have the opportunity to develop qualities of leadership and responsibility.

There never has, and likely never will be, a class of students that are the same level of ability with the same interests. For this reason, group activity is an excellent approach and can be easily integrated into a BASIC programming course. Small groups of students may be assigned the same programming problem or project; students work together to solve the problem and to help each other understand the scope of the program. Group activity has been exceedingly successful in writing simulation programs; students actively work in concert to achieve the goal of a finished "product".

An alternative in dealing with multi-level, varied-skill students is to provide the same open-ended project to the class. Advanced students will work through the entire project and, perhaps, design their own extensions to the problem. Slower students are able to experience the satisfaction of completing the "required" segment of the project and will be positively motivated if they exceed the teacher's and their own expectations. An excellent example of such a project (The Game of Life) can be found in Understanding Mathematics and Logic Using BASIC Computer Games; this book, published by DIGITAL, is not included in the BASIC Programming Curriculum Package.

Field trips and guest speakers give students diversion from normal classroom activities and broaden their knowledge of computing and the use of computers. For example, visiting the local newspaper production facilities may give students new insight into the use of computers in the publishing industry. Inviting a representative from a computer manufacturer to speak to a class can provide students with information about the manufacturing, assembly and testing of new computers.

Points to Remember

A few additional suggestions that have not been mentioned relate directly to the teacher's role, not only in a BASIC programming course, but in all teaching endeavors. Lest they be forgotten, they are repeated here!

NEVER FEEL THAT YOU NEED TO KNOW ALL THE ANSWERS! Particularly in the beginning, it's probably impossible to know them all. Students have been known to be so motivated by computing, and have so much extra time on their hands to experiment and discover, that few teachers can keep up with them. Use that student resource as much as possible. Capitalize on the opportunity; when the answer isn't at hand, use the peer teaching approach.

One last reminder--REMEMBER THAT THE VERSION OF BASIC THAT YOUR COMPUTER SUPPORTS MAY BE DIFFERENT FROM THE VERSION THAT THE TEXTS AND RESOURCES ARE BASED ON. For this reason it is essential that all programs and illustrations be tested on the school computer prior to presentation of the concept.

three methods for teaching basic

Choosing the method for teaching BASIC programming is as critical as the planning of the computer facility and plays an important role in the ultimate success of the program of instruction.

Three possible methods are discussed here:

(1) Self-teaching method (programmed instruction)
(2) Short course method (ten two-hour sessions)
(3) Integrated course method (semester or full year)

Each method has a unique set of requirements and considerations which must be understood and upon which a choice must be based.

1 self-teaching method

The self-teaching method is accomplished by using a programmed instruction or self-teaching BASIC text at the computer terminal. Students acquire hands-on expertise as they perform the daily lesson assignments and generally are able to write BASIC programs within two to ten hours of beginning this type of instruction. This approach permits students to work independently, at their own pace, while it frees the teacher to deal with individual problems. Should several students have trouble with the same concept, group tutoring sessions can be designed using A Short Course in BASIC.

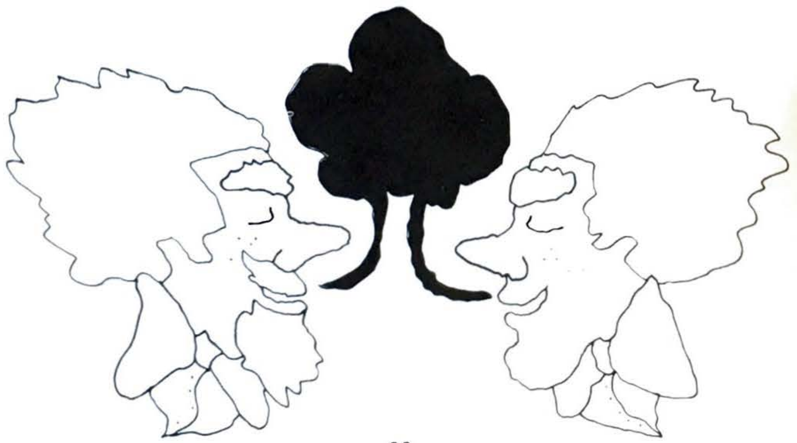
Two self-teaching BASIC texts are included in the BASIC Programming Curriculum Package: Population (for grades 9 and above) and Chapter 1 of the EduSystem Handbook (for grades 7 and above). Additional programmed instruction texts are listed in the Appendix of this guide. Regardless of the text that is chosen, each student should have a personal copy to facilitate the recording of answers.

It is particularly important when using this method that the differences between the versions of BASIC supported by the computer system and the text are fully understood. Because students are working independently, any discrepancy in language features will cause great confusion if it is not pointed out before the student confronts it.

In addition, student reference handouts are critically important in this method. Because students are learning BASIC at the terminal, questions or problems not only impede progress, but are likely to waste valuable computer time. Many problems can be easily avoided if students are provided with reference cards or handouts for error diagnostics, special function operations, procedures for punching paper tape, debugging aids, complex BASIC commands or instructions, and the differences between the versions of BASIC supported by the computer system and the text.

The self-teaching method has been successfully used with small classes where sufficient terminal time is available for each student. It has been particularly successful in teacher training sessions and faculty workshops. In some cases this method has been combined with a second method; the self-teaching approach is used for a single one-hour session to familiarize students with terminal operation and several simple BASIC instructions. This session is then followed by continued instruction utilizing either the short or integrated course method.

Whether the self-teaching method is used alone or in conjunction with another approach, it has been observed that the hands-on experience provides increased student motivation and permits students to achieve a knowledge of the BASIC language at their own pace.



2 short course method

The short course method combines teacher presentations of BASIC language elements with hands-on computer experience through a series of exercises and class assignments. A short course is generally designed as ten two-hour or twenty one-hour class sessions; student programs are normally written and tested outside of these classroom sessions.

The short course approach can consist of a two week mini-course or a ten week course with one session per week. It has been implemented as part of a beginning algebra course and as a course of instruction independent of existing curricula.

<u>A Short Course in BASIC</u>, which is contained in the BASIC Programming Curriculum Package, contains a full discussion of the teaching approach for the short course method. The order of topics in the text was selected as the most successful of many classroom-tested sequences but may be easily altered to suit local requirements. The discussion which accompanies each topic does not reiterate the discussions contained in the numerous available BASIC tests; instead, it deals with items related to each topic that should be emphasized in classroom presentation, provides ideas for effective teaching, and references additional resources which may be used to augment discussion, exercises and activities. Each of the ten lessons contain exercises that may be used for classroom demonstration or student assignments. The exercises are elementary and can be used with students of different ability levels.

The short course method has been chosen by many schools for advanced students and faculty training and has been particularly successful where a school-wide computer program is being established.

3 integrated course method

This method, like the short course method, combines classroom presentations of BASIC language concepts with hands-on computer programming experience. Integrated courses are semester or full year courses and are often taught in conjunction with existing mathematics or science curricula.

The integration of BASIC programming and mathematics or science courses can be implemented in several ways:

(1) BASIC programming is taught at the outset of a mathematics or science course. Algebra I is often considered the appropriate course for this integration.

(2) BASIC programming is taught in weekly sessions, as part of math or science. For example, Monday through Thursday algebra is taught, and BASIC is taught on Friday.

(3) Programming concepts are scattered throughout a mathematics or science course; programming topics are introduced as they are applicable to the math or science topics currently being studied. For example, discussion of BASIC matrix operations and trigonometric functions would be postponed until those topics are reached in the mathematics curriculum.

Choosing the approach for implementing the integrated course method depends on individual preference and the limitations of the local situation. For example, if the available computer facility is too small to accommodate all interested students, approach (1) might be used. A single class in BASIC programming could be scheduled at several different times during the year; in this way computer scheduling conflicts are reduced and participating students have ample computer time to complete hands-on work.

The integrated course method has been successfully implemented beginning at the junior high school level. Many teachers have considered programming instruction of this nature as "advanced" coursework and have, as a result, limited enrollment only to above-average students. However, in many schools, low-ability students have gained tremendous benefit from this exposure to programming. Because programming concepts are not difficult to learn, low-ability students are able to achieve repeated successes with the computer--and each success provides the motivation necessary to encourage students to achieve higher goals. Indeed, the integrated course method has resulted in exceptional achievement for all levels of students.

Numerous texts and resource guides are available for use in a one-semester BASIC programming course. Many of these resources are referenced in the Appendix; others are listed in the _Bibliography of Texts and Resources on the Uses of Computers in Education_ which is contained in the BASIC Programming Curriculum Package. In addition, _A Short Course in BASIC_ can be modified to suit the needs of the integrated course method. Additional ideas for classroom discussions and activities are contained in the next section of this guide.

index of basic topics

This index contains individual BASIC programming topics and references for each topic to several selected texts. The index is separated into three parts:

Part I: Discussion Index

References chapters within the texts that contain good discussions and explanations of BASIC topics.

Part II: Assignment Index

References chapters within the texts that contain exercises and student assignments for BASIC topics.

Part III: Project Index

References chapters within the texts that contain ideas for projects and additional activities for BASIC topics.

PART I: DISCUSSION INDEX

TOPIC	BLAKESLEE	COAN	KEMENY	PAVLOVICH	EDUSYSTEM HANDBOOK	SLIDE
Teletypewriter	Appendix A				1	1A or 1B
Errors	Chapter 10	Appendix B	13-14	21	6-26 to 28	
PRINT	3-1 to 3-9	1-3,10	5-9	11-19	1	6
Order of Operations	4-9 to 4-17	4	8-9	22-24	1	2
Special Functions	7-1 to 7-10	47-48	26-27	47-54	6-26	22
Direct/Program Mode	1-3 to 1-4				6-3	3
Line Numbers	2-5		13	10-11	1	3
SCR, LIST, RUN, CTRL/C	1-3 to 1-4, 10-2 to 10-3	4-5	12	11	1	18 and 19
END	2-6	2	5-9	12	1	4
Variables	2-3 to 2-4	8	8	31-32	1	2
String Variables	2-4		33-37	71-78	6-8 to 10	
LET	4-7 to 4-18	5-6	5-9	29-35	1	5
INPUT	3-3 to 3-4, 4-1 to 4-2	91-100	12-13	41-43	6-4	7
TAPE/KEY		Appendix A		7-8	6-39 to 40	21
REMARK	3-3, 11-1	13	42	111	1	14
GOTO	5-14	3	6-9	79-83	1	9
Floating Point Numbers	2-1	7-8	8-9	19-21	1	

TOPIC	BLAKESLEE	COAN	KEMENY	PAVLOVICH	EDUSYSTEM HANDBOOK	SLIDE
Size of Numbers	2-1			57	1	
Significant Digits				57	1	
Flowcharting	11-1 to 11-2	13-15 Appendix D		60-62,69	1	
IF-THEN	5-10 to 5-14	15-22	6-9	67-71	1	11
STOP	5-16 to 17	59		81	1	16
FIX(X)					6-8	22
TAB (X)	3-9 to 10, 9-1 to 2	217-219	53-54	18	1	22
EDIT				267-269	6-29 to 6-30	20
ALT MODE				27-28		
Direct GOTO					6-3	
Abbreviated Commands					6-2	
Appended Comments					6-4 to 6-5	
Commands After IF-THEN					6-5	
Multiple Statements per Line					6-2	
FOR/NEXT	5-1 to 5-7	26-31	16-18	91-96	1	10
Nested Loops	5-7 to 5-9	31-34	18-20	96-98	1	
DEFINE	7-8	55-61	27-30	130-146	1	12

TOPIC	BLAKESLEE	COAN	KEMENY	PAVLOVICH	EDUSYSTEM HANDBOOK	SLIDE
GOSUB/RETURN	8-1 to 8-5	41-45	30-31	107-113	1	13
READ/DATA	4-2 to 4-6	3-4	5-9	35-38	1	8
RESTORE	4-6 to 4-7	100-103		38-40	1	
DIMENSION	6-3 to 6-4	66	23-25	103-104	1	
Subscripted Variables	6-1 to 6-16	35-38	21-25	101-105	1	
ON-GOTO	5-14 to 5-15		66	83-85	6-6	17
ON-GOSUB					6-6	
RND (X)	7-10 to 7-13	62-69	65-67	49-51	1	22, 15
RANDOMIZE	7-13		67	50-51	6-7	15
LINPUT					6-10	
Comparing Strings				77	6-11	
CHR$					6-11 to 6-12	
MID					6-12	
LEN					6-13	
CAT					6-13	
Debugging	Chapter 10	39-43	148-154			

PART II: ASSIGNMENT INDEX

TOPIC	BLAKESLEE	COAN	KEMENY	PAVLOVICH	SMITH
PRINT	3-11	11-12	10		15-24, 31-34, 37-44, 109 117, 120
TAB(X)	3-11				
INPUT	4-19	96, 100	14-15	43,66-67	35-36
READ/DATA	4-19	11-12	10	40-41	43-44
FOR/NEXT	5-18 to 5-19	31, 34-35	20	101	57-59, 71-74, 97, 104, 124
IF-THEN	5-18 to 5-19	22-23	10	71	15-24, 31-34, 37-44, 109 117, 120
GOTO	5-18 to 5-19		10	85	15-24, 31-34, 37-44, 109 117, 120
LET		11-12	10		15-24, 31-34, 37-44, 109 117, 120
Order of Operations		11-12		24-25	23-24
Special Functions	7-14 to 7-15	50	32	56-57	23-24, 108
Variables		11	9		
Flowcharting		22-23		62	
Numbers in BASIC		11-12	9	58-59	23
Manipulating Strings			37-38	78, 129-130	
GOSUB/RETURN	8-6	45-46	32	117-118	67-68, 71-74, 105, 107
Debugging			43		3
DEFINE		61	32	146	63-66, 71-72

TOPIC	BLAKESLEE	COAN	KEMENY	PAVLOVICH	SMITH
RESTORE		103			
Subscripts	6-17	38-39, 45-46	25	105-106	46-56, 69-74, 110, 114-116, 121, 129-130, 133, 146
Sorting		45-46	25		48-49, 122-123, 127-128
Random Numbers	7-14 to 7-15	72	74-75		103, 106, 111-113, 118
Nested Loops	5-18 to 5-19	34-35	20		60-62, 71-74, 119,135-136

PART III. PROJECT INDEX

APPLICATIONS	BLAKESLEE	COAN	KEMENY	PAVLOVICH	SMITH
Graphing	9		53-56	171-180	
Trigonometry			47-50,56	154-171 Appendix C	131-132
Geometry				Appendix B	
Analytical Geometry				Appendix D	
Calculus				Appendix E	
Algebra				Appendix A	
Probability		13	17	Appendix F	149-150, 155-158
Statistics			15	225-235	101-102, 137-140, 159-164
Number Theory		48,54,70-78 104-113	57-64		
Quadratic Function Analysis		8			
Complex Numbers		10			
Polynomials		11	50-53, 56	181-187	
Matrices and Vectors		12	16	8	75-96
Sorting	6-8 to 6-13	40-45	23		

APPLICATION	BLAKESLEE	COAN	KEMENY	PAVLOVICH	SMITH
Area Under a Curve				235-257	
Files			13		
Text Processing/ Coding			14		
Special Topics				Appendix G	
Business			12		141-145, 147-148, 151-152
Simulations			65-75		126
Games			11		125

appendix

BASIC TEXTS

Blakeslee, Theodore. Introducing BASIC. EduComp Corporation, 1972.

Coan, James. Basic BASIC. Hayden Book Company, Inc., 1970.

Kemeny, John and Thomas Kurtz. BASIC Programming. John Wiley and Sons, Inc., 1971.

Pavlovich, Joseph and Thomas Tahan. Computer Programming in BASIC. Holden-Day Inc., 1971.

Smith, Robert E. Discovering BASIC. Hayden Book Company, Inc., 1970.

BASIC TEXTS (Self-Instruction)

Albrecht, Robert. Teach Yourself BASIC (Volumes I and II). Tecnica Education Corporation, 1970.

EduSystem Handbook. Digital Equipment Corporation, 1973.

Population: A Self-Teaching BASIC Primer. Digital Equipment Corporation, 1972.

COMPUTER FAMILIARIZATION

Ahl, David H. (ed.). 101 BASIC Computer Games. Maynard, Mass.: Digital Equipment Corporation, 1973.

Ahl, David H. Getting Started in Classroom Computing. Maynard, Mass.: Digital Equipment Corporation, 1973.

Ball, Marion. What Is A Computer? Boston, Mass.: Houghton Mifflin Company, 1972.

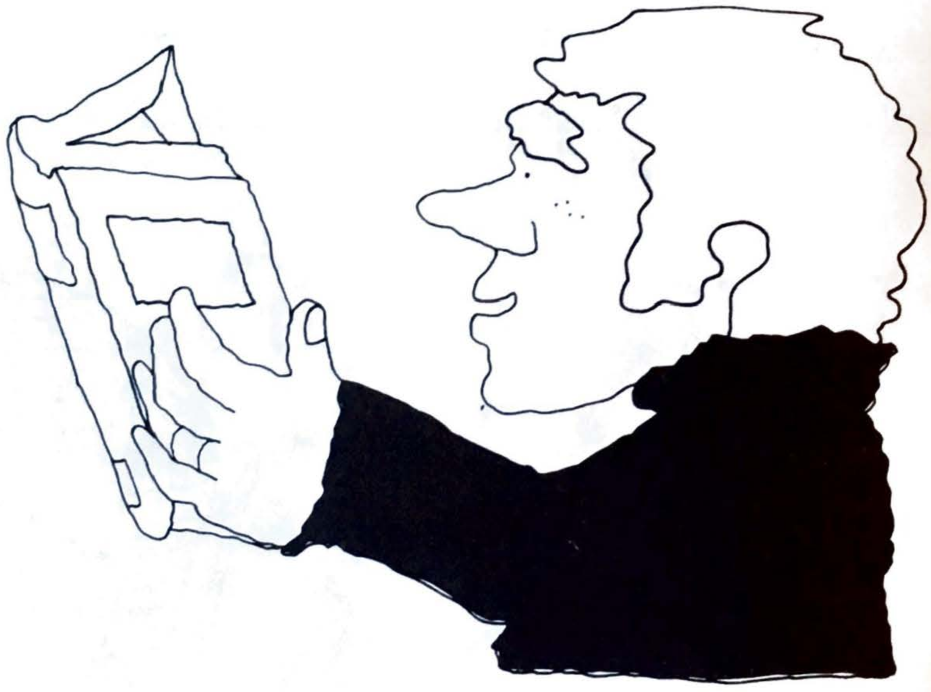
Bower, Sally R. A Curriculum Guide for Teaching BASIC. Maynard, Mass.: Digital Equipment Corporation, 1974.

Ellsworth, Pamela H. A Short Course in BASIC. Maynard, Mass.: Digital Equipment Corporation, 1974.

Gerald, Curtis. Computers and the Art of Computation. Reading, Mass.: Addison-Wesley Publishing Company, 1972.

Rothman, Stanley and Charles Mosmann. Computers and Society. New York: Science Research Associates, Inc., 1972.

The Computer: How It's Changing Our Lives. Washington, D.C.: U.S. News and World Report, 1972.



ZLTEA

A Curriculum Guide for Teaching BASIC

digital