

THE DEVELOPMENT OF AN INDEPENDENT  
STUDY GUIDE TO PROVIDE INSTRUCTION  
TO HIGH SCHOOL STUDENTS IN WRITING  
PROGRAMS FOR THE TI-82 AND TI-83  
GRAPHICS CALCULATORS

By

JAMES G. BOWEN

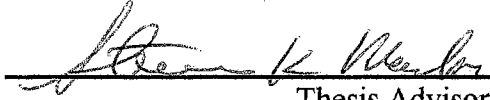
Bachelor of Science  
Oklahoma State University  
Stillwater, Oklahoma  
1971

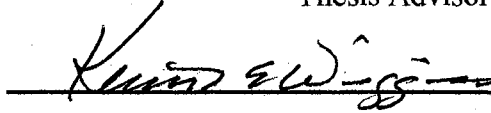
Master of Science  
Oklahoma State University  
Stillwater, Oklahoma  
1985

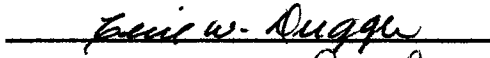
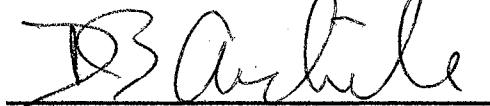
Submitted to the Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
DOCTOR OF EDUCATION  
May, 1997

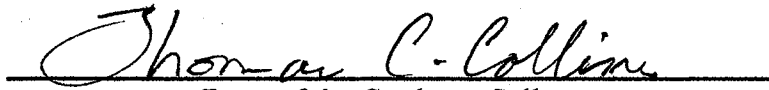
THE DEVELOPMENT OF AN INDEPENDENT  
STUDY GUIDE TO PROVIDE INSTRUCTION  
TO HIGH SCHOOL STUDENTS IN WRITING  
PROGRAMS FOR THE TI-82 AND TI-83  
GRAPHICS CALCULATORS

Thesis Approved:

  
Thesis Advisor



  
Dean of the Graduate College

## ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my major advisor, Dr. Steve Marks for the guidance he has provided me during the preparation of this dissertation. He has been both an excellent advisor and a good friend and I will always appreciate the support he has given me.

I also extend a sincere debt of gratitude to my doctoral committee members Drs. Kenneth Wiggins, Cecil Dugger, and Douglas Aichele for their support.

In particular, I would like to thank Dr. Wiggins for his flexibility in assisting me to meet the requirements of this degree while I was engaged in teaching mathematics at Stillwater High School.

I would like to further acknowledge the work I have done on graphing calculators with Douglas Aichele over the past four years as the inspiration for this dissertation. His thorough understanding of the appropriate role of technology in mathematics education has been most beneficial to me. I have enjoyed the work we have done together and I have learned much about the proper way to teach mathematics.

I would also like to thank Steve DeBauge, Lenda Hill, and Guy Harris of Texas Instruments Inc. for their evaluation of the programming manual which resulted from this dissertation.

## TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION . . . . .	1
Statement of the Problem . . . . .	6
Significance of the Study . . . . .	6
Purpose of the Study . . . . .	7
Limitations of the Study . . . . .	7
Assumptions of the Study . . . . .	7
Definition of Terms . . . . .	8
II. REVIEW OF SELECTED LITERATURE . . . . .	10
The Relationship of Programming to Mathematics Education . . . . .	10
Textbook Support for Graphics Calculators . . . . .	16
Support Programs from Professional Organizations . . . . .	21
Support Materials from Commercial Vendors . . . . .	24
III. DEVELOPMENT OF THE PROGRAMMING MANUAL . . . . .	28
The Selection of a Specific Calculator . . . . .	29
The Format of the Manual . . . . .	32
Selection of Programming Instructions . . . . .	34
Creation of Sample Programs . . . . .	37
Evaluating the Effectiveness of the Manual . . . . .	49
Evaluation by High school Students . . . . .	49
Evaluation by the Staff of Texas Instruments . . . . .	52
Distribution of the Manual . . . . .	53
IV. RESULTS . . . . .	55
Results of the Student Questionnaires . . . . .	55
Results of the Critique by the Texas Instruments Staff . . . . .	57
The Final Version of the Programming Manual . . . . .	58

Chapter	Page
V. SUMMARY AND RECOMMENDATIONS . . . . .	59
Summary . . . . .	59
Recommendations . . . . .	61
BIBLIOGRAPHY . . . . .	63
APPENDIXES . . . . .	67
APPENDIX A - CALCULATOR TECHNICAL SPECIFICATIONS . . . . .	68
APPENDIX B - FINAL VERSION OF THE PROGRAMMING MANUAL . . . . .	74
APPENDIX C - STUDENT QUESTIONNAIRE . . . . .	147
APPENDIX D - TI CARES NEWSLETTER . . . . .	149
APPENDIX E - INSTITUTIONAL REVIEW BOARD APPROVAL FORM . . . . .	152

## CHAPTER I

### INTRODUCTION

During the 1990's, the use of graphics calculator technology in the high school mathematics classroom has become increasingly commonplace. The impressive capabilities of the newest generation of graphics calculators have forced mathematics educators to reevaluate how they teach mathematics. Wonderful new opportunities to increase student enthusiasm toward mathematics are now available to mathematics teachers that were not available prior to the introduction of this new technology. As with the introduction of any new methodology, however, mathematics educators are still undergoing a period of experimentation in which they are attempting to determine which methods should be used to make the most effective use of this new technology.

The National Council of Teachers of Mathematics (NCTM) has taken the lead in attempting to give some direction to mathematics education in the 1990's through the publication of three related documents; Curriculum and Evaluation Standards for School Mathematics (NCTM, 1989), Professional Standards for Teaching Mathematics (NCTM, 1991), and Assessment Standards for School Mathematics (NCTM, 1995). This collection of standards emphasizes the appropriate use of technology throughout the mathematics curriculum. In fact, the NCTM's position assumes that students in grades

nine through twelve will have access to graphing calculator technology at all times (NCTM, 1989).

Since 1990, there has been a dramatic increase in the quantity and quality of the support material available to teachers who are attempting to integrate graphics calculator technology into their mathematics curriculum. The available support may be broadly grouped into three categories.

The first category consists of support from national professional organizations, such as the NCTM, and from local and state professional mathematics organizations. The NCTM organizes regional and national meeting in which mathematics educators have an opportunity to meet with their colleagues and exchange ideas about creative new ways in which to utilize graphics calculator technology. Other professional groups, such as Teachers Teaching with Technology (Teachers Teaching with Technology, 1996), conduct summer institutes to provide additional training to math and science teachers desiring to learn how to use this technology. The NCTM also publishes recommendations in the form of guidebooks to aid teachers implementing the NCTM standards (Farrell, 1994).

The second category involves the support material available from commercial vendors. These materials generally take the form of workbooks in which teachers are provided with suggestions on how calculators may be effectively used in specific mathematics classes. Some of the workbooks include experiments and exercises which the teacher may use to supplement their classroom instruction (Best & Penner, 1994). Others contain sample calculator programs which, when entered into the calculator, allow the students to complete activities which are contained in the workbook

(Brueningsen & Kraviec, 1994). The workbooks are not consistent, however, in their attitude toward copyright restrictions. Certain authors encourage teachers to make photocopies of the activities in their manuals for their classrooms (Lund & Anderson, 1996), while other authors attempt to protect their copyright restrictions by insisting that no part of the work be reproduced by any means (Kelly, 1992).

The third category involves the support materials that textbook publishers have included in the latest editions of their mathematics textbooks. The most recent editions of mathematics textbooks have attempted to integrate graphing calculator activities throughout their textbooks. Many textbooks include calculator activities at the end of each instructional block (Demana, Waits, & Clemens, 1992). Many of the new textbooks include a supplemental manual containing additional activities requiring the use of technology (Leiva & Brown, 1997). Certain new textbooks now contain an appendix which lists calculator programs which may be entered into a calculator by the student using the textbook (Larson & Hostetler, 1997). Upon review, it is clear that textbook publishers in the 1990's have at least attempted to adhere to the recommendations regarding technology contained in the NCTM standards, and these efforts will certainly benefit teachers as they attempt to modify their classroom instruction to conform to the same standards.

It is possible that an individual investigating the current status of graphics calculator technology in mathematics education might conclude, from the numerous sources of support now available to teachers, that all questions regarding the use of calculators have been successfully addressed. There remains, however, one area regarding the use of graphing calculator technology which has received essentially no



attention from the groups listed above. That area involves the skill required to write programs for the graphics calculator to answer different types of questions. Perhaps the primary reason this area has been neglected is the fact that there is no consensus among mathematics educators on exactly where the subject of programming should fit into the mathematics curriculum. In fact, there is little agreement on whether or not programming has any place in the mathematics curriculum. As a result, while each of the sources above might make occasional reference to calculator programming and some might even contain program listings which students may copy, there still exists no single source of information providing instruction on how to program a graphics calculator which has been specifically written for high school students.

The inability of high school students to obtain instruction on how to program their graphics calculators limits the students' ability to make the most efficient use of their calculator as an educational tool in solving new and different types of problems. As an example, one group of high school science students attempted to predict the orbits of satellites by creating a model on their calculator. In designing the calculator model they faced several problems, including the fact that none of them had ever programmed a graphing calculator (Papay, Serum, & Donnelly, 1996).

There are several other specific areas where the ability to program a calculator may benefit high school students. The use of programmable graphics calculators has now been approved on the ACT, SAT, and Advanced Placement (AP) Calculus exams (Educational Testing Services, 1996). Each of these exams support the NCTM recommendation that if calculators are to be an integral part of any mathematics course, then they should also be a part of the instruments used in the assessment process. The

AP Calculus exam recognizes that there are differences in capabilities between expensive and inexpensive calculators, and to eliminate those differences students are encouraged to store any programs they choose in their calculators prior to the exam (Educational Testing Services, 1996). However, students are cautioned to make sure they have a thorough understanding of how those programs work before taking the exam.

There are several obstacles facing high school students attempting to learn how to program their calculators. Most high school mathematics teachers are unfamiliar with how to program a graphics calculator and, therefore, they can offer little help to interested students. Presently, the best way for high school students to learn how to program their calculators is for them to enroll in a computer programming course. At the completion of the programming course it should be possible for students to apply many of the skills they have learned to writing programs for a graphics calculator. Because of differences in the syntax of the programming languages, however, students can expect to have certain difficulties as they begin to write calculator programs. While the guidebook that is packaged with the calculator does describe the syntax of the calculator programming language, it offers little help in describing how the programming commands may be combined into efficient programs.

The commercial vendors have also tended to neglect the subject of calculator programming. A very small number of programming guides for graphics calculators do exist, but all of these guides were written for a broader audience than just high school students. As a result, there is a need for an independent study guide to provide instruction to high school students wishing to learn how to program their graphics calculators.

## Statement of the Problem

Although there are many sources of information available to assist teachers and students attempting to integrate graphics calculator technology into the mathematics curriculum, there currently exists no single source of information providing instruction on how to program a graphics calculator which is specifically targeted for high school students.

The problem addressed in this study is the development of an independent study guide providing programming instruction for a graphics calculator, written specifically for high school students and which can be used in an extracurricular setting.

## Significance of the Study

The ability to program a graphics calculator offers several advantages to high school mathematics students. While it is true that calculator programming skills are not an absolute necessity for high school students, students who possess these skills will have the ability to be much more creative in their approach to problem solving now and in the future. For example, students who can program their graphics calculators to create dynamic models of experiments performed in their math and sciences classes can develop a more thorough understanding of the concepts being studied. The students who write their own programs also have total creative control over the model being investigated. While students who lack the ability to program their calculators may utilize programs available through commercial vendors, they must rely on the format presented

by the vendor and they sacrifice the option to be able to create their own custom model. The literature review will identify many additional advantages enjoyed by students who have the ability to write their own graphics calculator programs.

### Purpose of the Study

The purpose of this study is to develop an independent study guide to provide calculator programming instruction to high school students in an extracurricular setting. This guide will enable high school students to obtain instruction on how to program their graphics calculators outside the classroom without relying on their teachers.

### Limitations of the Study

The purpose of this study was to develop an independent study guide to provide programming instruction to high school students in an extracurricular setting. The study did not address the issue of how programming should be integrated into the high school mathematics curriculum.

### Assumptions of the Study

First, it is assumed that there is a need for an independent study guide to provide instruction to high school students who desire to learn how to program their graphics calculators.

It is also assumed that in order to maximize distribution of the resulting manual, the final version of the manual will be distributed free of all copyright restrictions and, when possible, free of charge. The manual will also be placed in a website on the internet for downloading by interested students.

### Definition of Terms

BASIC. BASIC is an acronym for Beginner's All-purpose Symbolic Instruction Code. BASIC is a high level computer programming language developed in the 1960's at Dartmouth College by professors John Kemeny and Thomas Kurtz. There are several versions of BASIC including QBASIC developed by Microsoft Corporation (Baumann & Mandell, 1992).

Calculator Based Laboratory (CBL). The Texas Instruments CBL is an electronic data gathering instrument that was introduced in 1994. Experimental data may be collected using a variety of electronic probes. The data may be stored in the CBL or passed by link cable to a graphics calculator where an analysis of the data may be performed by utilizing the statistical capabilities of the calculator (Texas Instrument, 1994).

High School Student. For the purposes of this study, a high school student is a student in the ninth, tenth, eleventh, or twelfth grade.

National Council of Teachers of Mathematics (NCTM). The NCTM, founded in 1920, is a nonprofit professional organization dedicated to the improvement of mathematics education for all students. With more than 125,000 members, the NCTM is

the largest mathematical organization in the world and it is recognized as the leader in efforts to insure excellence in mathematical education (NCTM, 1989).

PASCAL. PASCAL is a high level computer programming language that was developed in 1969 by professor Niklaus Wirth in Switzerland. It is named after Blaise Pascal, a 17<sup>th</sup> century mathematician and philosopher. PASCAL is the official language of the Advanced Placement Computer Science Examination for high school students.

TI-82 Graphics Calculator. The Texas Instruments TI-82 graphics calculator was introduced in 1994. This calculator was created specifically for high school mathematics classes and it has become the most popular graphics calculator in use at the high school level. A detailed technical description of the TI-82 calculator may be found in Appendix A.

TI-83 Graphics Calculator. The Texas Instruments TI-83 graphics calculator was introduced in 1996. This calculator is an enhanced version of the TI-82 calculator containing many improvements which were recommended by high school teachers using the TI-82. Programs written for the TI-82 calculator will also run on the TI-83 calculator. A detailed technical description of the TI-83 calculator may be found in Appendix A.

## CHAPTER II

### REVIEW OF THE LITERATURE

This chapter contains a review of the literature that is relevant to the relationship of programming to mathematics education. The review begins with a survey of the various attitudes shown by mathematics educators concerning the role of programming in mathematics education. This is followed by a review of current editions of mathematics textbooks available to teachers and students who are attempting to integrate graphing calculator technology into the mathematics curriculum. A review of graphing calculator support material available to teachers from professional educational organizations is then presented. The remainder of the literature review focuses on current graphing calculator material available from commercial vendors.

#### The Relationship of Programming to Mathematics Education

The relationship between programming and mathematics is not new. The world's first electronic computer, the ENIAC, introduced in 1945, was created to solve complex mathematical problems (Halacy, 1962). The earliest relationship between programming and mathematics was not a matter of choice, but one of necessity. Because no software

existed to run the computer, mathematicians had no choice but to program the computer themselves if they wanted to use the computer to solve problems.

With the development of sophisticated programming languages in the 1960's and 1970's, many university engineering and mathematics departments added courses in numerical methods, in which languages such as FORTRAN were used to program the existing mainframe computers. In the 1970's, the introduction of programmable microcomputers and programmable hand held calculators offered mathematics educators new opportunities to combine programming with mathematics education. As the following opinions will demonstrate, however, there exists no consensus on the role of programming in mathematics education. In fact, several authors who indicate that, while they recognize the potential benefit that programming offers to mathematics education, they have not committed themselves to a clear position on exactly where programming should fit into the mathematics curriculum, if indeed it should be included at all.

One of the most influential supporters of programming in mathematics education is Seymour Papert, who created the LOGO computer programming language in the late 1970's. The LOGO programming language was written for elementary level schoolchildren and reflected Papert's belief that even schoolchildren at the elementary level could derive great benefits in their mathematics education from programming computers (Papert, 1980). When explaining why he favored children writing programs over traditional methods of computer aided instruction, Papert (1980) stated:

In many schools today, the phrase "computer aided instruction" means making the computer teach the child. One might say the computer is being used to program the child. In my vision, the child programs the computer and, in doing so, both acquires a sense of mastery over a piece of the most modern and powerful technology and establishes an intimate contact with some of the deepest



ideas from science, from mathematics, and from the art of intellectual model building.

I shall describe the learning paths that have led hundreds of children to becoming quite sophisticated programmers. Once programming is seen in the proper perspective, there is nothing very surprising about the fact that this should happen. Programming a computer means nothing more than communicating to it in a language that it and the human user can both understand. And learning languages is one of the things children do best. Every normal child learns to talk. Why then should a child not learn to talk to a computer? (p. 5)

Shumway (1984), while investigating the use of the BASIC programming language with elementary schoolchildren, takes an even stronger position stating, “Regarding computers in public education, it is my basic view that as soon as children are in school they should have the opportunity to program computers to solve mathematics problems” (p. 127). At the completion of his study Shumway (1984) concluded:

It is difficult to make direct claims, but such programming activities as these seem to offer a wealth of logical reasoning experiences for children. In my view, a substantial change in our curriculum can be made as young children have the opportunity to write programs similar to those illustrated. It seems to me that we can use programming daily as an integral part of the mathematics we teach. The richness of the mathematics encountered will be multiplied dramatically; more young children will see mathematics as a dynamic, creative activity; and teaching and learning mathematics will be a lively, active sport indeed. (p. 134)

Stwertka (1987) investigated several recent developments which were creating major changes in mathematics. He considered the effect of computers on mathematics to be among the most important changes, stating:

Until quite recently, most mathematicians thought of computers as large number crunchers that do arithmetic quickly and store huge amounts of data. Much to their surprise, and almost against their will, the influence of computers on mathematics itself has been increasing at an accelerating rate. It has established itself as a kind of mathematics laboratory instrument, an exploratory device that is helping mathematicians create new fields and make new discoveries. It is changing the way many mathematicians go about proving a mathematical statement or theorem. A large part of modern statistics, for example, is devoted to

making intelligent use of the computer. Mathematicians can use the computer to quickly eliminate hundreds of possible proofs, or use computer graphics to give them valuable information on how some complex mathematical functions behave.(p. 88)

Smith (1984) investigated programming using the BASIC programming language in the middle school curriculum in an attempt to improve students problem solving skills.

She stated:

Instruction in computer programming that is integrated with the traditional topics of seventh- and eighth-grade curriculum can address each of the issues: basic skills proficiency, the lack of freshness in the middle school curriculum, computer literacy, and time. Being able to write computer programs is a significant component of computer literacy; thus, programming as an integral component of the mathematics instruction necessarily contributes to computer literacy. In the process of writing programs, students explore and develop their own algorithms for solving all kinds of problems. This is an extension of their basic skills. They must also work a number of computational exercises to make sure the computer is giving the answers they want - thus gaining practice. Many facets of the basic skills of problem solving are either taught or practiced in computer programming. Trial and error, an important problem solving technique, is seldom used by students with conventional textbook problems, but it is very effective in the computer solution of certain problems. Thinking ahead is another problem solving competency used in computer programming. Students must ask themselves at each step, "What will happen if I do this?" Once a program is written, they can be encouraged to ask, "What will happen if I change this?" The latter question is the looking back step in problem solving.

If programming is integrated with the mathematics curriculum, then the extra time spent in learning to program is minimal. For example, programming can be incorporated into, and enhance the learning of, new concepts and skills. The following is one of many models. The teacher introduces a new skill, the students work a number of problems until they think they can tell the computer how to do them, and finally the students program the computer and test their program. This programming activity gives students additional practice on the concept, tests their understanding of the concept, increases their programming skills, and gives them a better understanding of computers. (p. 137)

There are instances where the introduction of programming into mathematics classes has been initiated by the students and not the teachers. Ruthven (1992), while introducing programmable graphing calculators to his students, noted that:

Unlike calculating and graphing, programming is not integral to advanced secondary school mathematics in its present form. Consequently, this aspect of calculator use was not given equal stress by all project teachers. However, after one school year, around two-thirds of the students were making confident and spontaneous use of the programming facility. Here, some gender differences were noted: in particular, a lower proportion of female students expressed confidence in programming, although this must be judged in the light both of the tendency of male students to be less self-critical and of the lower proportion of female students with prior programming experience. (p. 93)

Other researchers have been more cautious in their attitude regarding the role of programming in mathematics education. Hatfield (1984) investigated the manner in which computer technology was being integrated into mathematics instruction. He understood the potential advantages that programming a computer offered, but he also recognized problems that might arise if programming was used improperly. He observed:

When mathematics learners construct their own computer programs, powerful learning experiences can occur. Indeed, many mathematics teachers have realized the apparent connections between the thinking involved in building, testing, correcting, and refining one's own computer algorithm and many aspects of mathematical thought. In addition, many mathematics educators have recognized the potential effects that solving a curriculum oriented computer programming task can have on stimulating or enhancing understanding of a mathematical concept, problem, or procedure. It has seemed almost a natural relationship for computer programming to be taught within the mathematics curriculum; this may have led to serious distortions of our goals and objectives for a sound mathematical education. We know that students can learn to write computer programs; we must now consider the more critical questions about why our mathematics students should become engaged in such tasks. (p. 2)

In his conclusion, Hatfield cautions:

Lastly, we must be sensitive to potential misuses and abuses of the computer. It is wrong to expect the computer to take over. Computers do not make good replacements for mathematics teachers. When using student programming, we must employ a constructive, problematic pedagogy; it violates the interactive, solving context to give completed computer programs to our students. (p. 7)

Camp and Marchionini (1984) also acknowledge the advantages programming gives to mathematics education, however, they are quick to identify several specific problems which must be overcome before programming can be successfully integrated into the mathematics curriculum. They state:

There are now programming languages that can be used with students throughout the grades, and recent work on the format of programming exercises makes programming a viable means for promoting mathematics learning.

... There is a role for computer programming in mathematics education. But, as you consider the options, make a clear distinction between the study of programming, which belongs to the domain of computer literacy and computer science, and the use of programming to achieve learning objectives in mathematics. Programming in mathematics education is defensible to the extent that it helps achieve goals for school mathematics. But if programming is to be a general strategy for mathematics education, then particular attention must be given to format, formality, and methods by which programming is integrated into curriculum and classroom practice. (p. 118)

They continue:

The problem with programming in currently available languages is that it can be a difficult, multi-step, time-consuming task. Teachers prefer devoting their time and their students' time to learning mathematics. They recognize that programming in its purest sense can potentially interfere with mathematics learning because it (1) takes too long, and (2) focuses attention on programming rather than mathematics. (p. 119)

Camp and Marchionini conclude:

Integrating programming to promote learning into mathematics teaching and curriculum will be no simple task. First of all, there is the massive problem of teacher education. Teachers not only need to be computer literate but also need to

be able to use computers in a variety of ways, including programming, to support instruction. Second is the problem of curriculum development. To date, there are no elementary, middle, or high school textbook series that truly enhance mathematics learning through computer experiences - programs to run, like simulations, management, drill and practice, and so forth, or programming to do. (p. 125)

The lack of a consensus on the role of programming in mathematics education has resulted in the fact that it is currently quite rare to find high school mathematics classrooms in which students receive any instruction on how to program their graphics calculators. Camp and Marchionini (1984) correctly identified one of the greatest problems which must be overcome, and that is the problem of teacher training. The simple fact is that very few high school math teachers know how to program a graphics calculator and, therefore, they can offer no assistance to their students. Until the mathematics community comes to a consensus and agrees that there is a place in the mathematics curriculum for programming and begins to provide teachers with formal training in programming, the best source of calculator programming instruction for high school students will probably be in the form of an independent study programming guide.

### Textbook Support for Graphics Calculators

Camp and Marchionini (1984) reported that in 1984 there existed no high school textbooks which truly supported computer experiences for students. Thankfully, that situation had changed dramatically by 1997. Two factors were primarily responsible for this change.

The first was the introduction of hand held graphics calculators in the late 1980's. For the first time, high school students had a tool which was relatively inexpensive but had the capability to allow students to investigate the graphs of functions. It had the added advantage of portability, which was missing in microcomputers, and this allowed students to take the new tool home at night to assist in their homework.

The second factor was the introduction of the NCTM Curriculum and Evaluation Standards (NCTM, 1989) which called for all students to have access to graphics calculators at all times. Textbook authors and publishers responded to this call by including reference to graphics calculator technology throughout their new textbooks from beginning algebra through calculus.

One of the most obvious changes in the new textbooks may be seen on the front cover. Many of the new textbooks now include the phrase "A Graphing Approach", positioned immediately after the title of the book. (Demana, Waits, & Clemens, 1992; Demana et al. 1994; Finney, Thomas, Demana, & Waits, 1993; Larson, Hostetler, & Edwards, 1993) This new graphing approach to textbook design reflects a significant change from previous editions, and it represents an attempt by publishers to assist teachers and students in integrating graphing calculator technology. Because it is important to understand exactly what is implied by the term "A Graphing Approach", a complete description of this term as found in the Demana, Waits, and Clemens textbook Precalculus Mathematics: A Graphing Approach (1994) is presented:

As in the previous editions, this text is designed to be used in a one or two semester precalculus course. We take advantage of the power and speed of

modern technology to apply a graphing approach to this course. The characteristics of this course are described below.

#### Integration of Technology:

Use of a graphing utility - whether a hand held graphing calculator or computer graphing software - is not optional. Technology allows the focus of the course to be on problem solving and exploration, while building a deeper understanding of algebraic techniques. Students are expected to have regular and frequent access to a graphing utility for class activities as well as homework.

#### Problem Solving:

The ultimate power of mathematics is that it can be used to solve problems. Technology removes the need for contrived problems and opens the door for realistic and interesting applications. Throughout this text, we focus on what we call problem situations - situations from the physical world, from our social environment, or from the quantitative world of mathematics. Using real life situations makes math more understandable to the students, and students come to value mathematics because they appreciate its power.

Throughout this text we used a three step problem solving process.

Students will be asked to:

1. Find an algebraic representation of the problem.
2. Find a complete graph of the algebraic representation; and
3. Find a complete graph of the problem solving situation.

#### Multiple Representations:

A quantitative mathematical problem can often be approached using multiple representations. In a traditional precalculus course, problems are analyzed using an algebraic representation, and perhaps a numerical representation. However, modern technology allows us to take full advantage of a graphical, or geometric, representation of a problem. Our understanding of the problem is enriched by exploring it numerically, algebraically, and graphically.

#### Exploration:

We believe that a technology based approach enriches the student's mathematical intuition by exploration. With modern technology, accurate graphs can be obtained quickly and used to study the properties of functions. Students learn to decide for themselves what technique should be used. The speed and power of graphing technology allows an emphasis on exploration.

#### Geometric Transformations:

The exploratory nature of graphing helps students learn how to transform a graph geometrically by horizontal or vertical shifts, horizontal or vertical stretches and shrinks, and reflection with respect to axes. This develops students' abilities so they can sketch graphs of functions quickly and understand the behavior of graphs.

#### Foreshadowing Calculus:

We foreshadow important concepts of calculus through an emphasis on graphs. Using graphs, students can find maxima and minima of functions, and intervals where functions are increasing or decreasing and limiting behavior of functions are quickly determined graphically. We do not borrow the techniques of calculus - rather we lay the foundation for later study by providing students with rich intuitions about functions and graphs.

#### Approximate Answers:

Technology allows a proper balance between exact answers that are rarely needed in the real world and accurate approximations. Graphing techniques such as zoom-in provide an excellent geometric vehicle for discussion about error in answers. Students can read answers from graphs with accuracy up to the limits of machine precision.

#### Visualization:

Graphing helps students gain an understanding of the properties of graphs and makes the addition of geometric representations to the usual numeric and algebraic representations very natural. Exploring the connection between graphical representations and problem situations deepens student understanding about mathematical concepts and helps them appreciate the role of mathematics. (p. vii)

Another popular approach used by textbook publishers utilizes the phrase “An Integrated Approach”. (Larson, Boswell, & Stiff, 1995; Larson, Boswell, Kanold, & Stiff, 1996; Larson, Kanold, & Stiff, 1997) New textbooks containing an integrated approach provide for a vertical integration of material from what were previously separate and independent courses. For example, concepts from geometry may be intermixed in an algebra course and vice versa. These textbooks also have completely integrated graphing calculator technology throughout the book.

Other new textbooks, which retain only their old titles such as Calculus, Precalculus, and Trigonometry, have also updated the new editions for graphing



calculator technology.(Larson, Hostetler, & Edwards, 1994; Larson & Hostetler, 1997; Leiva & Brown, 1997)

High school mathematics textbooks covering Geometry have also attempted to meet the NCTM standards. The new textbooks utilize a microcomputer software, such as Cabri Geometry II which enables students to manipulate geometric figures on a computer graphics screen, and hand held graphics calculator technology (Aichele, Hopfensperger, Leiva, Mason, Murphy, Schel, & Vheru, 1998).

Taken as a whole, the new text do an excellent job in assisting teachers and students in using graphing calculators to explore mathematics. All the books have special “technology” problems contained at the end of each section which require the use of graphing technology for their solutions. In addition, most of the new textbooks have technology supplements in the form of workbooks which teachers can use to enhance their presentations. A careful examination of the new books reveals that the publishers have done a good job in complying with the NCTM standards regarding the use of technology.

There remains, however, one area which has been neglected in the new textbooks, and that area involves providing calculator programming instruction for students. Some of the textbooks do contain listing of calculator programs (Larson & Hostetler, 1997), however, none of the books provide instruction on how to write original calculator programs. As a result, students and teachers will find the new textbooks to be of little use in learning how to program their calculators.

## Support Programs from Professional Organizations

In recent years, the support available to teachers from national professional organizations such as the National Council of Teachers of Mathematics (NCTM) has also increased substantially. The program committees responsible for the selection of presentations at the regional and national NCTM meetings have made sure that information regarding the use of graphing calculators occupies a prominent position. Teachers attending these meetings can obtain ideas covering a wide variety of current graphing calculator projects. In addition, the NCTM professional journal "The Mathematics Teacher" for contains a special section dedicated to technology in each issue.

One of the most important sources of information on graphing calculator technology is the Teachers Teaching with Technology (pronounced T-cubed) program. This nationwide program was established in 1986 by Professors Bert K. Waits and Franklin Demana of The Ohio State University. The T-cubed program provides hands on training to public school teachers and college instructors in the use of Texas Instruments graphing calculator technology. This training takes place in various summer institutes located on high school and college campuses throughout the United States. The institutes are subject specific and are taught by instructors who are practicing educators, experienced in the use of graphing calculators. Since 1986, the T-cubed program has expanded to include 14 institute topics. A description of each institute and its subject material (Teachers Teaching with Technology, 1996) is as follows:

1. Elementary Mathematics: A two day institute focusing on K-6 mathematics topics that can be enhanced with the Math Mate AOS four-function calculator and with the Math Explorer fraction calculator. It is designed for elementary school teachers in grades K-6.
2. Middle School Mathematics: A one week institute focusing on mathematics that can be enhanced by the TI-12 Math Explorer, Explorer Plus, and the TI-82 or TI-80 graphing calculators. It is designed for mathematics teachers in grades 5-8 including pre-algebra teachers.
3. Algebra: A one week institute focusing on algebra for high school teachers of pre-algebra, algebra, and advanced algebra using the TI-82, Ti-83, and TI-85 graphing calculators.
4. Geometry: A one week institute focusing on the Cabri Geometry II interactive geometry software package for high school geometry teachers.
5. Precalculus: A one week institute for teachers of advanced algebra, trigonometry, precalculus, math analysis, or calculus using the TI-83 calculator.
6. Calculus: A one week institute for high school AP calculus teachers using the TI-83, TI-85, and TI-86 graphics calculators.
7. Calculus Reform and the TI-92: A one week institute for secondary mathematics teachers interested in using the TI-92 and interested in implementing calculus reform in their classes.
8. Statistics: A one week institute focusing on statistics topics which can be integrated into grades 9-12 mathematics using the TI-83. This institute is for

high school pre-algebra, algebra, consumer math, applied math, general mathematics, business mathematics, advanced algebra, statistics, and precalculus teachers.

9. **Advanced Placement Statistics:** A one week institute focusing on AP statistics using the TI-83 calculator. This institute targets high school AP statistics teachers.
10. **Connecting Mathematics and Science:** A one week institute for high school teachers of physics, chemistry, precalculus, and calculus who are interested in the use of the Calculator Based Lab ( CBL ) and the TI-83 graphics calculator.
11. **Chemistry and Biology:** A one week institute for secondary chemistry and biology teachers focusing on enhancing chemistry and biology experiments both inside and outside the lab using the CBL and TI-83 calculator.
12. **Algebra and the TI-92:** A two week institute focusing on preparing secondary mathematics teachers to use the TI-92 to enhance a high school algebra course.
13. **Modeling and Data Analysis:** A one week institute for high school precalculus teachers focusing on the North Carolina School of Science and Mathematics modeling with functions and data analysis precalculus curriculum. Instructors will be from the North Carolina School of Science and Mathematics.
14. **Integrated Mathematics Using the TI-92:** A one week institute for high school mathematics teachers interested in integrating mathematics using the TI-92 calculator. The participants are introduced to a wide range of investigations

emphasizing the relationships of topics within mathematics as well as between mathematics and other disciplines.

The average cost to attend one of the institutes is \$100. These institutes provide high school and college mathematics teachers with an excellent opportunity to become knowledgeable about the appropriate integration of Texas Instruments graphing calculator technology into their mathematics classrooms. One disadvantage of these institutes is that they utilize only technology provided by Texas Instruments Inc. and they do not discuss technology available through other manufactures.

Once again, although these organizations do an excellent job in training teachers how to use the “built-in” functions of the calculator, very little attention, if any, is paid to providing instruction on how to write programs for the calculators. Another source of information regarding programming is still needed.

#### Support Materials from Commercial Vendors

The quantity of graphics calculator support material available to teachers and students has increased substantially during the past five years. In 1992, Dr. Brendan Kelly correctly noted that “There is a dearth of (approximately zero) instructional materials which teachers can use with their students to help them use the graphics calculators to do mathematics” (p.2). Fortunately, there has been a rapid expansion in both the variety and quality of the material from commercial vendors that teachers may now utilize in their mathematics classes. In most cases, these materials take the form of instructional guides or workbooks, with each workbook dedicated to a particular make and model graphing

calculator. These materials are available from a number of different publishers and authors. The various publications may be grouped into several specific types.

The first type consists of publications that are concerned primarily with describing the basic operation of the calculator to new users. An example of this type publication, A Guided Tour of the TI-85 Graphics Programmable Calculator (Lucas & Lucas, 1992), contains only a description of the operation of the calculator. Other publications of this type contain classroom exercises or experiments which teachers may photocopy and distribute to their students (Best & Penner, 1994; Lund & Anderson, 1996; Rich, Rose, & Gilligan, 1996). These publications are generally more useful to teachers because they demonstrate specific ways in which the various function of the calculator may be used to support mathematics instruction. Some of these publications also contain a brief description of calculator programs (Best & Penner, 1994; Rich et al., 1996), however, their focus is on teaching math with the calculator, and not on calculator programming instruction.

Another type of publication consists of manuals for both science and math teachers which provide instruction on how new data gathering instruments, such as the Texas Instruments Calculator Based Laboratory (CBL), may be used in combination with graphics calculators (Brueningsen, Bower, Antinone, & Brueningsen, 1994; Brueningsen & Krawiec, 1994; Nichols, 1995). These manuals typically consist of a series of exercises or experiments which may be photocopied by the teacher for use in the classroom. The students use probes attached to the CBL to gather data during an experiment, and then connect the CBL to a graphics calculator and pass the collected data to the calculator for analysis. All of these publications use calculator programs because the CBL cannot be

used in conjunction with a graphics calculator unless a calculator program is written and executed to act as a driver for the CBL. Each of the exercises in the manuals contain listing of calculator programs which a teacher or student may enter into the calculator. Like previous calculator support material, these manuals make no attempt to provide instruction to the student on how to write programs for the calculator. Instead, the students are provided with a completed program that they simply enter into their calculators without necessarily understanding how or why the program works. This is obviously a less than ideal situation because if a student desires to modify one of the experiments listed in the manual, unless they can find some source of instruction on programming the calculator, they will be unable to make the custom changes they desire.

There are an extremely limited number of publications in the final type, specifically, those publications devoted to instruction on how to program a graphics calculator. The first examples of this type are the guidebooks packed with the calculators when they are purchased. The guidebooks provided with the Texas Instrument series of calculators do contain a chapter on programming, but the chapter primarily describes only the syntax of the programming language, and it makes a very poor reference for students who have no previous programming experience (Texas Instruments, 1993, 1996). In an attempt to improve the quality of programming instruction for their calculators, Texas Instruments published a series of brochures expanding the descriptions to include three sample programs and a discussion on how the various programming commands work (Texas Instruments, 1993, 1994, 1996). While these brochures represent an improvement over the calculator guidebooks, the limited number of examples makes them less than ideal as a programming guide for high school students.

As of 1996, only one other author, Dr. Brendan Kelly (1992), had written a manual specifically dedicated to providing instruction on how to program graphics calculators. Several of the examples in Kelly's manual, however, are clearly intended for a wider audience than just high school students, and as a result, it is also less than ideal as a programming guide for high school students.

The conclusion that can be drawn from a review of the literature associated with graphics calculators, is that in spite of the improvement in variety and quality of material available to support teachers and students, there still exists a need for an independent study guide written to provide instruction specifically to high school students on how to program their graphics calculators.



## CHAPTER III

### THE DEVELOPMENT OF THE PROGRAMMING MANUAL

In order to develop a calculator programming guide specifically written for high school students, a number of decisions had to be made which would give direction to the project. Because the results of one decision would influence later decisions, a specific order was created for each option. First, each of the decisions which guided the development of this manual will be listed, followed by a detailed description of how the decision presented by each choice was resolved. The decisions which guided the development of this manual were:

1. Which specific calculator should be selected for this project?
2. What should be the format of the manual?
3. Which of the available calculator programming instructions should be included in the manual?
4. What sample programs should be included in the manual?
5. What evaluation process should be used to evaluate the effectiveness of the manual for its intended audience?
6. What distribution process should be used to maximize distribution of the final version of the manual?

The resolution of each of the questions posed will now be discussed in detail.

### 1. The Selection of a Specific Calculator

There are several manufacturers of quality hand held graphing calculators. Among these are Hewlett - Packard, Casio, Sharp, and Texas Instruments. Graphing calculators from each of these manufacturers are found to varying degrees in high school classrooms today. The selection of a specific calculator is further complicated by the fact that most manufacturers produce a series of graphing calculators targeted for different educational levels from middle school through college. Each calculator in a series offers a wide range of both capabilities and prices. Certain calculators are clearly more appropriate at one level than another, and teachers must use care in deciding which calculator would be most appropriate for their students.

Calculators from each of the above manufacturers were carefully reviewed, including the complete line of educational support material offered by the manufacturer. The result of this investigation identified the Texas Instruments TI-82 and TI-83 graphing calculators to be the best choice for the specific calculators to be used in this project.

This decision was based on the following factors:

1. If one of the goals of this project was to provide programming instruction to the maximum number of students, then the selection of the calculator used should be based, in part, upon which calculator is most commonly found in high school mathematics classes. The TI-82 and TI-83 calculators were specifically targeted for high school mathematics students by Texas

Instruments Inc., and they represent the most popular graphing calculators in use at the high school level today. Even though they have different names, the TI-82 and TI-83 are essentially the same calculator. The TI-82 was introduced in 1994 and quickly became the most popular graphing calculator at the high school level. The TI-83, introduced in 1996, is an improved version of the TI-82. The design of the TI-83 was based upon suggested improvements to the TI-82 from high school math teachers, however, both calculators are very similar in both their capabilities and operation. A detailed technical description of each calculator is contained in Appendix A.

2. The physical layout of the keyboards and the syntax of the programming languages used by each calculator are essentially identical. This allows for the development of a single programming manual which may be used for both calculators.
3. Both calculators contain a link cable which enables programs to be transferred between calculators of the same type. This feature is important because it enables high school students to exchange programs by simply linking their calculators together. In addition, programs stored in a TI-82 calculator may be passed directly to a TI-83 calculator.
4. The link cable system also allows both calculators to be connected to a microcomputer which enables the students to store copies of their programs in files on the computer. The students also have the capability, when appropriate software is used, to download calculator programs from the internet. A substantial library of public domain software is already in existence, and

many of these programs are available for downloading from internet sites to a microcomputer and then via the link cable directly into a calculator.

5. The Texas Instruments' Calculator Based Laboratory (CBL) has become a very popular tool in high school math and science classes for gathering experimental data. Both the TI-82 and TI-83 have the capability to link to the CBL via cable to retrieve and process experimental data collected by the CBL.
6. Texas Instruments Inc. has established a more extensive and comprehensive support system for educators using calculators than any of the other manufacturers. This is the most important reason for the widespread popularity of the TI series of calculators, and it offers many benefits to educators.
7. Although there are differences between the programming instructions available on the TI-82 and TI-83 calculators and other TI calculators such as the TI-85, TI-86, and TI-92, there exists sufficient similarity between the languages that a student who learned to program a TI-82 could easily adapt that skill to the programming of other TI calculators like the TI-85 or TI-92.

It was for these reasons that the TI-82 and TI-83 calculators were chosen as the focus of this project.

## 2. The Format of the Manual

It was decided that the format of the manual should be determined only after a review of existing high school level computer science textbooks. It was assumed that many of the students who might want to learn how to program a graphics calculator would be students who had completed a previous high school computer programming class. Because there is a basic format that appears to be common to most high school computer programming textbooks, it was felt that students learning to program a graphics calculator would be more comfortable utilizing a manual which was written in a format similar to the textbooks they had used in previous programming courses.

There are, however, some inherent difficulties in attempting to define a common format in computer programming textbooks, particularly when different programming languages are considered. Certain programming commands which are included in one programming language may not be available in another language, so while similarities may exist between the format of textbooks written in one language, it may be expected that differences in format will occur with textbooks using a different language. Even when these differences are considered, most textbooks still follow a somewhat similar format in introducing computer programming languages.

The computer languages available at the high school level which bear the closest resemblance to the programming language used on the TI-82 and TI-83 calculators are BASIC and Pascal. Both of these languages enjoy widespread use at the high school level, and it is reasonable to assume that many, if not most, students with previous

programming experience would be familiar with the syntax of one, or both, of these programming languages. For this reason, the investigation into possible formats for the manual centered on textbooks describing BASIC and Pascal. The textbooks reviewed were; Standard BASIC Programming with True BASIC (Catlin, 1992), Fundamentals of Pascal (Nance, 1990), Computer Science with Pascal (Mandell & Mandell, 1985), Running MS-DOS QBASIC (Halvorson & Rygmyr, 1991), QBASIC (Baumann & Mandell, 1992). The guidebooks provided with each calculator by Texas Instruments Inc. were also reviewed to examine their format (Texas Instruments, 1993, 1994, 1996).

The order in which the various programming operations appeared in the textbooks followed a generally consistent pattern. The fundamental statements required to initiate programming operations were presented first. Then the statements involving input and output operations such as PRINT, INPUT, and READ were described. Next statements used to control the flow of a program including IF-THEN and SELECT-CASE were introduced. Loop control instructions such as FOR-NEXT, DO-WHILE, and DO-UNTIL were presented next. These were usually followed by statements like FUNCTION and SUBROUTINE which allow the programmer to create sub-programs. Data storage structures including arrays, records, and files were generally presented last.

In addition to the order of presentation, there was a consistent pattern to the method of presentation. Most textbooks presented short program segments in which the statements that made up the program were printed first, followed by a simulated display window in which the output corresponding to the program was presented. This format has been successfully used by textbook publishers for many years to offer students an

easy to follow description of both the combination of statements that make up a program and the output that results when the program is executed.

As a result of this textbook review, the decision was made to present the calculator programming instructions in an order similar to that found in most textbooks. The format would also include short program segments which included the program statements and a simulated display window showing the resulting output. After presenting a description of each of the various programming instructions available on the graphing calculator, the final section of the programming manual would include several complete calculator programs demonstrating to the students how the individual programming instruction might be combined into programs to solve specific mathematical problems.

### 3. Selection of Programming Instructions

There are more than one hundred functions and instructions that are available to anyone desiring to write a program for the TI-82 and TI-83 graphics calculators. The goal of this study was to produce an introductory programming manual which presented only the essential commands a beginning student would need in order to write basic calculator programs. If this goal was to be obtained in a reasonably short introductory manual, then careful attention had to be paid to which programming instructions should be included in the manual. The final selection of which instructions were included was primarily based on the experience gained by the author in writing over one hundred TI-82 and TI-83 calculator programs. This experience enabled the author of this study to select the

instructions which would be essential to a beginning student programmer while eliminating instructions which could be added after the programmer gained experience in writing programs through the use of the manual.

The following programming instructions involving input and output operations were selected for inclusion in the final version of the manual.

1. **ClrHome** - Used to clear the home screen.
2. **Pause** - Used to temporarily delay execution of a program.
3. **Stop** - Used to permanently stop execution of a program.
4. **DispGraph** - Used to display a graphics screen.
5. **Disp** - Used to print information on a text screen.
6. **Output** - Used to format output on a text screen.
7. **Input** - Used to enter data into a program.
8. **Prompt** - Used to enter data into a program.

The following instructions involving control of the flow of a program were also included.

1. **If – Then** - Used to make decisions within a program.
2. **If – Then – Else** - Used to make decisions within a program.
3. **Lbl – Goto** - Used to control the flow of a program.
4. **Menu** - Used to control the flow of a program.
5. **Prgm – Return** - Used to execute sub-programs.



The following loop control instructions were selected to be included in the manual.

1. **For** - Used to control looping operations.
2. **While** - Used to control looping operations.
3. **Repeat** - Used to control looping operations.

The selection of a few instructions which are particularly useful in writing mathematics programs include the following instructions.

1. **Rand** - Used to generate random numbers.
2. **fMin** - Used to find where the minimum value of a function occurs.
3. **fMax** - Used to find where the maximum value of a function occurs.
4. **nDeriv** - Used to evaluate the derivative of a function.

Finally, the following instructions which allow the programmer to add information to a graphics display are especially useful.

1. **ClrDraw** - Used to clear the graphics screen.
2. **Line** - Used to draw straight lines on a graphics display.
3. **Horizontal** - Used to draw horizontal lines on a graphics display.
4. **Vertical** - Used to draw vertical lines on a graphics display.
5. **Circle** - Used to draw circles on a graphics display.
6. **Tangent** - Used to draw the tangent line to a given function.
7. **Text** - Used to print text information on a graphics display.

8. **DrawInv** - Used to draw the inverse of a given function.
9. **Pt – On** - Used to plot a point on a graphics display.

The instructions presented in the preceding list will allow a beginning student programmer to write reasonably sophisticated calculator programs which take full advantage of the graphing capabilities of both the TI-82 and TI-83 calculators. As the students gain experience, they can make reference to the calculator guidebook to utilize the remaining instructions which were not included in this manual.

#### 4. Creation of the Sample Programs

Two objectives were to be met in the creation of the sample programs to be included in the final version of the manual.

Because the primary goal of this study was to produce a programming guide specifically targeted for high school students, the first objective was to insure that the mathematics included in the sample programs was restricted to the level of mathematics commonly available to a typical high school student. The mathematics curriculum found in a typical high school ranges from a beginning algebra course at the freshman level through calculus at the senior level. Because the manual was designed to provide instruction to students at each grade level, care was taken to provide a variety of sample programs which insured that the instruction would be of value to students in beginning algebra as well as students in calculus. It was anticipated that senior level students taking calculus would derive the most benefit from the manual because they would understand

not only the programming instruction but also the mathematics involved in the trigonometry and calculus sample programs. Students in the lower grades would still find the manual beneficial because a number of lower level sample programs involving mathematics no higher than algebra were also included. The possibility also existed that even though a lower level student might not fully understand the mathematics in a particular sample program, that same student could still benefit from the example by noting the effects of the various programming instructions used in the sample program.

The second objective was to utilize the widest possible variety of programming instructions within the sample programs. Each of the instructions selected for inclusion in the manual may be found in at least one of the sample programs, and in several cases programs were designed to show alternative uses of the instructions. In certain instances, the same problem was solved in two or more ways just to show students the options that are available to them when writing programs.

In the final version of the manual, fifteen sample programs are included. A description of the problem solved by each program as well as a description of why certain programming instructions were included in each example will now be presented.

### Sample Program 1

Sample Program 1 uses a financial equation to demonstrate how a student may store a formula within a program. The program then evaluates the formula using values entered by the user.

This program demonstrates:

- Basic input and output operations.
- The **Disp** instruction to display output to the user.
- The **Prompt** instruction to enter values into the program.
- How an assignment statement may be used to store the equation within the program.
- The **ClrHome** instruction to clear the screen.

### Sample Program 2

Sample Program 2 allows the student to enter the X and Y coordinates of two points and then uses the distance formula and the midpoint formula from algebra to calculate both the distance between the points and the midpoint between the points.

This program demonstrates:

- The **Input** instruction to enter values into the program.
- How the **Pause** instruction may be combined with the **ClrHome** instruction to display output on more than one screen.
- The **Frac** instruction to display program output in fractional form.
- The **round** instruction to control the number of decimal places displayed in the output.

- The **Output** instruction for more precise control over the positioning of output on the screen.

### Sample Program 3

Sample Program 3 uses a random number generator to create a game in which the student attempts to guess which number between 1 and 100 the calculator has randomly selected. After the student enters a guess, the calculator will state whether the guess was too high or too low. This process is repeated until the student guesses the correct answer. The program will then display the number of guesses needed to determine the correct answer.

This program demonstrates:

- How to use the **int** and **rand** instructions to generate random integers within a specified range.
- The use of the **Lbl** and **Goto** instructions to control the flow of a program.
- How the **If** instruction may be used to make decisions within a program.
- The use of a “counter” variable to sum up the number of incorrect answers.

### Sample Program 4

Sample Program 4 presents a second solution to the guessing game posed in Sample Program 3. To the user, the input and output screens appear identical, however,

this program uses a “while” loop to control the flow of the program. Sample Program 4 and Sample Program 5 have been included to show students that more than one program structure may be used to solve the same problem.

This program demonstrates:

- The use of the **While** instruction to control the flow of a program. The “while” loop will repeat while a specific condition is true.
- How an **If - Then - Else** instruction may be used to make decisions within a program.
- The use of a priming input statement to “prime” the while loop.

#### Sample Program 5

Sample Program 5 presents a third solution to the guessing game posed in Sample Program 3. To the user the input and output screens appear identical, however, this program uses a “repeat” loop to control the flow of the program.

This program demonstrates:

- The use of the Repeat instruction to control the flow of a program. The “repeat” loop will repeat until a specified condition is true.

### Sample Program 6

Sample Program 6 uses a random number generator to simulate flipping a coin and then displays the resulting number of heads and tails. The user controls the number of times the coin is tossed. The program will also display the final percentages for heads and tails.

This program demonstrates:

- The use of the **For** loop instruction in which the upper limit of the loop is a variable to be determined by the user.
- How a random number generator may be used to model experimental data.
- The use of the **round** and **Output** instructions to precisely control the position of output displayed on the screen.

### Sample Program 7

Sample Program 7 allows the user to enter a function into a program while the program is running. The program switches from the text screen to the graphics screen and graphs the given function.

This program demonstrates:

- How the user may enter a given function into the program and store it in one of the **Y** variables so its graph may be displayed.

- The use of the **Zstandard** instruction to change from the text mode to the graphics mode and display the graph of a function.

### Sample Program 8

Sample Program 8 demonstrates how one program may call another program during execution. This program also shows how a “setup” program may be used to restore the default settings of the calculator prior to the execution of a second program.

This program demonstrates:

- How the **prgm** instruction may be used to have one program call a second program.
- How execution returns from the “called” program to the main program.
- How the default settings of the calculator may be restored within a program.

### Sample Program 9

Sample Program 9 allows the user to draw any size triangle on the screen by entering the coordinates of the three vertices of the triangle. This is the first sample program to make use of the programming instructions within the “draw” menu. These instructions are useful in placing text and figures on the screen to complement the graphing capabilities of the calculator.



This program demonstrates:

- How the **Line** instruction may be used to draw lines at any position on the screen.
- How the user may enter values into the “window” size variables from within a program.

### Sample Program 10

Sample Program 10 shows how the calculator may be used to graph conic sections. The program first allows the user to enter values into the equation of an ellipse in standard form. The program then graphs the ellipse and gives the user the option to zoom in or out in order to clearly view the ellipse.

This program demonstrates:

- The importance of coordinate geometry in writing programs which display graphics.
- How a conic section may be broken down into two individual functions for graphing purposes.
- How the “draw” menu instructions may be used to enhance the graphical presentation.
- The use of a zoom in or out option within a program.

### Sample Program 11

Sample Program 11 presents an example involving trigonometry in which the user is allowed to enter an angle in either the degree or radian mode. The program then displays all six basic trigonometric functions for the given angle. The program will anticipate any trigonometric functions which are undefined and will display the word “Undefined” in these cases.

This program demonstrates:

- The use of the basic trigonometric functions **Sin**, **Cos**, and **Tan**.
- The **Fix** instruction to limit the number of decimals displayed in the output.
- The use of conditional **Output** instructions to display various options as determined by the program.
- How the **Radian** and **Degree** instructions may be used within the program to control the mode of the calculator.

### Sample Program 12

Sample Program 12 presents a simple method of storing and displaying text information. This program stores the names and telephone numbers of individuals and then displays the names and numbers when the program executes.

This program demonstrates:

- How the **Disp** instruction may be used to store and display text information.
- How the **Pause** and **ClrHome** instructions may be used in combination to allow the user to control the pace of a program.

### Sample Program 13

Sample Program 13 allows the user to perform a graphical simulation of projectile motion. The user enters the initial height and the initial velocity of an object and the desired length of time for the model to run. The program then graphically models the object's height as a function of time with a continuous display of both height and time. Finally, the user is given the option to change the time period and restart the model.

This program demonstrates:

- How the calculator may be used by science students to bring equations to life by modeling real world situations.
- The use of the **fMax** instruction to automatically scale the graph so the entire projectile motion is displayed on the graph.
- The use of the **Text** instruction to combine both graphical and text information on the same screen.
- How the **Pt-On** instruction may be used as an alternative method of graphing a function.

### Sample Program 14

Sample Program 14 is a calculus program which combines both the computational and graphics features of the calculator in approximating a Reimann sum.. The user is allowed to enter a function and the program then graphically and numerically displays the Reimann sum over a given interval. The user is given the option of determining how many rectangles will be used to approximate the Reimann sum. The program then offers the user an opportunity to change the lower and upper limits of the interval and the number of rectangles used in the approximation.

This program demonstrates:

- How the user may enter a function into a “**Y**” variable for graphing within a program.
- The use of the **Xmin** and **Xmax** variables to control the width of the graphics screen.
- How to evaluate a function at a given value of X from within a program.
- How the **fMin** and **fMax** instructions may be used to automatically scale the graphics screen to maximize use of the display area.
- How the **Line** instruction from the “draw” menu may be combined with the graph of a function to increase the effectiveness of the display as a teaching and learning tool.

- How the user may be given the option to re-run the program and change key parameters.
- The importance of coordinate geometry in mathematical programming involving graphics.

### Sample Program 15

Sample Program 15 is a calculus program which allows the user to enter a function and a specific X coordinate. Then the program determines the equation of the tangent line to the function at the specified X value and displays both a graph of the tangent line and the equation of the tangent line in slope-intercept form. Finally, the program determines the equation of the normal line to the function at the specified X value and display both the graph of the normal line and the equation of the normal line in slope-intercept form.

This program demonstrates:

- The use of the **nDeriv** instruction to determine the slope of the tangent line at a specific value of X.
- How a program can change a linear equation from point-slope form into slope-intercept form for graphing purposes.
- How the equation of the normal line may be determined from the equation of the tangent line.

- The use of the **Text** instruction to display both text and graphical information on the same screen.

## 5. Evaluating the Effectiveness of the Manual

Two different methods were selected to evaluate the effectiveness of the manual. The first method involved an evaluation of the manual by high school students. The second method involved an evaluation of the manual by the educational marketing staff at Texas Instruments Inc. A detailed description of each method of evaluation is now presented.

### 1. Evaluation by high school students

The first method involved the creation of a questionnaire which would be completed by high school students evaluating early versions of the manual. The responses obtained from these questionnaires were used to improve later versions of the manual. The purpose of the questionnaires used in this study was solely to obtain specific suggestions from high school students in order that the final version of the manual would contain the most effective presentation of an independent study programming guide. It was never the intention of this study to perform a detailed statistical analysis of the results of the questionnaires.

A sample of the questionnaire used in this study may be found in Appendix C. A description of each of the eight questions included on the questionnaire is now presented.

**Questions #1 and #2:**

#1: What grade level are you in this school year?

#2: What math course are you taking now or will be taking next semester?

It was anticipated that a senior in high school who was enrolled in calculus would be able to derive the greatest benefit from this manual because this student should be able to understand the mathematical content of all of the examples in the manual. These questions were included primarily to determine if the examples contained in the manual provided adequate instruction for students at the lower grade levels. If students at the lower levels experienced problems in using the manual, then additional lower level examples could be included.

**Questions #3 and #4:**

#3: Prior to using this calculator programming manual, have you had any previous experience in computer programming?

#4: If you have had any previous programming experience, did you have difficulty adapting those programming skills to the graphing calculator?

These questions were included as a check on how effective the format used in the manual was in providing instruction in programming. It was felt that a student with previous programming experience would have an advantage in learning to program a graphing calculator, however one of the goals of this project was to create an manual which would be useful to students with no previous programming experience. If students with no previous experience indicated that they had difficulty using the manual, then adjustments would have to be made to either the format or the content of the manual.

**Questions #5 and #6**

#5: Did you find the manual easy, reasonable, or difficult to use?

#6: How helpful would it be to you if additional sample programs were included in the manual?

These questions were included to determine if there was a correlation between the students grade, mathematics level, and level of previous programming experience and the ease or difficulty they experienced in attempting to use the manual. If students experienced difficulty in using the manual, then one solution would be to add additional sample programs for added clarity.

**Question #7**

#7: How valuable do you think your ability to write programs for your graphics calculator will be during your high school and college career?

This question was included to survey student's attitudes concerning the importance they place on programming skills, both now and in their immediate future. Although this question did not influence the design of the manual, it was felt that an understanding of student's attitudes towards learning programming skills would be meaningful in future research relating programming to the mathematics curriculum.

**Question #8:**

#8: Please suggest any improvements you think would make this manual easier to use.

This final question was included to give students an open opportunity to make any suggestions to improve the final version of the manual.



Early versions of the manual, including the questionnaire, were distributed to high school students over a nine month period. The initial group of students consisted of approximately 30 ninth and tenth grade students attending a summer math academy at Oklahoma State University in 1996. Students attending the academy received copies of the manual along with a TI-83 graphics calculator for the duration of the academy. The students received no formal instruction on programming the calculators, but were given the programming manual as an extracurricular activity to be used as an independent study guide. At the completion of the academy, they returned the questionnaires containing their responses.

During the 1996-97 school year, students at Stillwater High School in Stillwater, Oklahoma, participated in the study on a voluntary basis. Students with an interest in learning how to program their graphics calculators were given copies of the manual and a TI-83 calculator and asked to evaluate how effective the manual was as an independent study guide. A total of 30 students in varying grade levels from sophomores in algebra through seniors in calculus participated in this evaluation. They also returned completed questionnaires as part of their evaluation.

As each group of students completed their evaluation, the comments provided on the questionnaires were used to make revisions and improvements to early versions of the manual.

## 2. Evaluation by the staff of Texas Instruments

The second method of evaluation consisted of a thorough review of the manual by members of the Texas Instruments educational marketing and programming staff. A copy of the manual was presented to Texas Instruments and they were asked to distribute the manual to all staff members in a position to suggest revisions or improvements to the manual. The TI staff spent approximately six weeks reviewing the manual and returned a technical critique of the programming presentation as well as a general critique of the format of the manual.

The results of both methods of evaluation are contained in the results section of this document.

## 6. Distribution of the Manual

Several decisions were made which would insure that the final version of the manual would be available to the largest possible audience of high school students. First, the selection of the TI-82 and TI-83 calculators meant that the programming manual would be written for the calculators most commonly used at the high school level. This should insure that it would appeal to the largest group of high school students currently using graphics calculators.

The decision was also made to produce a manual which was free of all copyright restrictions. Teachers and students using the manual would be encouraged to make additional copies of the manual for distribution at their school. To allow further distribution, the manual would be provided either free of charge or at the cost of making

copies of the manual. It was believed that this approach would relieve the financial burden to school systems utilizing the manual.

Finally, two additional steps were taken to provide easy access to students and teachers desiring to obtain a copy of the manual. First, a copy of the manual was placed in a web site on the internet so students with internet access could download free copies of the manual (<http://www.math.okstate.edu/~aichele/GrCalcs/TGC.html>). Second, Texas Instruments Inc. included a reference to the manual in their "TI Cares" newsletter containing educational support material for their graphics calculators (Appendix D). Students and teachers may order copies of the manual directly from Texas Instruments Inc., for copying costs only, by calling 1-800-TI-CARES.

The steps described above should insure the maximum distribution of the manual to its intended audience.

## CHAPTER IV

### RESULTS

The two methods of evaluation discussed in the previous chapter both contributed to the final design of the programming manual. The results of the student questionnaire primarily influenced the number and type of sample programs included in the final version of the manual. The results of the critique by the staff at Texas Instruments Inc. improved the technical presentation of the programming instructions and eliminated minor errors and omissions in the sample programs thus insuring the accuracy of the examples presented in the manual.

#### Results of the Student Questionnaires

Approximately 60 students completed questionnaires after participating in the evaluation of the programming manual. A summary of the students' responses to the questions will now be presented along with the effect those responses had on the final design of the manual.

The response to questions #1 and #2 regarding grade level and mathematics curriculum level showed that the manual was evaluated by a relatively even distribution

of students throughout each of the grade levels from nine through twelve. This even distribution of students, and the responses obtained, indicated that the manual received a thorough evaluation from students at each of the levels of its intended audience.

As anticipated, students at the upper level seemed to have an easier time making full use of all the examples in the manual. Students who experienced difficulty understanding certain sample program tended to be students at the lower levels who had not yet completed a mathematics course covering the material present in a specific example. Generally, the problems they experienced were a result of not being familiar with the mathematics presented rather than not being able to follow the programming instructions utilized in the example. As a result of the suggestions for improving the manual by the students reviewing early versions of the manual, the number of sample programs was extended from ten to fifteen in the final version of the manual and each of the five new sample programs was targeted for students at the lower levels. After the five new programs were added, lower level students reviewing a later version of the manual appeared to find the later version much more beneficial in providing programming instruction.

It was also anticipated that students who had completed a previous programming course would have an easier time learning to program the graphics calculator, and the responses indicated that this indeed was the case. However, students with no previous programming experience indicated that they also found the manual easy to use and very instructive as a programming guide. The most common suggestion for improving the manual was to include additional sample programs at the lower level and this suggestion was implemented in the final version of the manual.

The responses to the final question concerning the importance that high school students placed on the ability to write calculator programs showed a wide variation in the importance students attached to this skill. Many felt that programming skills would be very beneficial during both high school and college, while others felt that only minimal importance should be attached to the ability to program a calculator. It should be remembered that this manual is intended to be used as an extracurricular activity by students who have an active interest in learning to program their calculators. In spite of the variation in interest demonstrated by the students, it is anticipated that there still exists a large number of students who will benefit from this programming manual.

The overall results of the student questionnaires indicated that the final version of the manual did an effective job as an independent study guide in providing programming instruction to students at all grade levels in high school, including students with wide variations in both their mathematics and previous programming experience.

#### Results of the Critique by the Texas Instruments Staff

The educational marketing staff at Texas Instruments ( TI ) also returned a critique of a preliminary version of the manual. The focus of the critique by TI staff was to conduct a thorough examination of each of the programming instructional examples and sample programs to identify any errors in the programs and to make sure the programs performed as they were expected to. All errors and omissions identified by the TI staff were corrected in the final version of the manual. In addition, the TI staff members offered several suggestions for improving the accuracy and the presentation

format of the manual and many of their suggestions were also included in the final manual. The efforts of the TI staff are acknowledged and greatly appreciated.

### The Final Version of the Programming Manual

The final version of the programming manual designed to serve as an independent study guide for high school students may be found in Appendix B. To obtain a copy of the manual, students and teachers can find ordering information for the manual in the TI-Cares newsletter contained in Appendix D. A copy of the manual has also been stored in a website on the internet and may be downloaded by students with internet access using the address <http://www.math.okstate.edu/~aichele/GrCalcs/TGC.html> .

## CHAPTER V

### SUMMARY AND RECOMMENDATIONS

#### Summary

The goal of this study was to develop a programming manual which could be used as an independent study guide by high school students desiring an introduction to programming on the TI-82 and TI-83 graphics calculators. The final version of the manual which was developed in this study appears to have satisfied that goal. This conclusion is based on two primary evaluations.

First, several preliminary versions of the manual were tested and critiqued over a nine month period by a variety of students ranging in grade level from ninth grade students through college level students. The students who took part in the evaluation had widely varied mathematical backgrounds ranging from Algebra 1 in high school through college level mathematics. There was also a wide variation in computer programming experience ranging from students who had no previous programming experience to students who had completed at least two programming courses. An examination of the questionnaire completed by students who evaluated the programming manual confirmed the following:



1. Almost every student who reviewed the manual stated that they found it to be a very helpful introduction to programming a graphics calculator.
2. Students who had completed a previous programming course appeared to find the transition to programming a graphics calculator easier than students who had no previous programming experience.
3. Because the sample programs in the manual ranged from Algebra 1 through Calculus, students who had completed higher level math courses such as Calculus found it easier to understand the mathematical methods demonstrated in the advanced sample programs. This difference was anticipated at the beginning of the study and was, therefore, expected.

A review of the student critiques of early versions of the manual found that the most common criticism suggested that additional examples should be added to the manual to add clarity to several of the programming instructions. As a result of these comments, several new examples and sample programs were added to the final version of the manual to clarify any problem areas identified by the students. Students reviewing the final version of the manual appeared to find both the quantity and content of the examples presented satisfactory to insure their understanding of the programming instructions.

The second evaluation was based on a thorough review of the manual by members of the staff of Texas Instruments Inc. Several members of the Texas Instruments educational marketing and programming staff spent about six weeks conducting a detailed review of the manual. Several specific points involving programming errors and omissions were identified by the Texas Instruments reviewers,

and each error or omission was corrected in the final version of the manual. The reviewers also made many pertinent suggestions involving changes in the format of the manual which, when incorporated into the final version, resulted in a more orderly presentation of the material. The efforts of the Texas Instruments staff in improving the final version of the manual are very much appreciated.

The fact that Texas Instruments Inc. has endorsed the final manual is evidenced by the following:

1. A reference to the manual has been included in the reference list of published graphing calculator material in the Spring 1997 TI - CARES educational support programs newsletter which is distributed nationally.
2. Texas Instruments Inc. has further agreed to handle both the publishing and distribution of the manual nationally by having persons interested in obtaining a copy of the manual call 1 - 800 - TI - CARES . Texas Instruments will keep copies of the manual in stock and will mail them, at cost, to individuals or groups who place an order.

### Recommendations

This study was designed to produce an independent study guide which could be used to provide programming instruction to students in an extracurricular setting. This study made no attempt to define the role of programming within the mathematics curriculum. Past objections to including programming within the mathematics curriculum have included the fact that the limited time available to complete the core

mathematics material leaves no time to add additional instruction in programming. In recent years, however, there has been a significant change in the manner in which high schools schedule classes which may permit the inclusion of programming within the mathematics curriculum. This change involves a “block schedule” in which high schools complete courses under a semester arrangement similar to many college schedules. The block schedule allows students to increase the number of mathematics courses they complete prior to graduation from high school (Kramer, 1997). Many high schools have used the block schedule to add new mathematics course to their curriculums. One recommendation is to investigate the possibility of adding a new course providing instruction in mathematical programming and modeling using graphics calculators within a block schedule.

Researchers investigating the role of technology in mathematics education must be careful to consider the appropriate role of technology. It is important that the technology should not be allowed to drive the curriculum. The appropriate role of technology is one in which the technology enhances the mathematical discussion without diverting the focus away from the mathematical principles being discussed.

Finally, as the infusion of technology into mathematics instruction continues and the distinction between disciplines of mathematics and computer science becomes increasing blurred, it is recommended that new investigations be initiated to determine the proper role of programming within the mathematics curriculum.

## BIBLIOGRAPHY

- Aichele, D., Hopfensperger, P., Leiva, M., Mason, M., Murphy, S., Schell, V., Vheru, M. (1998). Geometry: Explorations and applications. Boston, MA: Houghton - Mifflin.
- Baumann, S. K., & Mandell, S. L. (1992). QBasic. St. Paul, MN: West.
- Best, G. W., & Penner, D. A. (1994). Using the TI-82 to explore precalculus and calculus. Andover, MD: Venture.
- Brueningsen, C., Bower, B., Antinone, L., & Brueningsen, E. (1994). Real-world math with the CBL system. Dallas, TX: Texas Instruments Inc.
- Brueningsen, C., & Kraviec, W. (1994). Exploring physics and math with the CBL system. Dallas, TX: Texas Instruments Inc.
- Camp, J.S., & Marchionini, G. (1984). Programming and learning: Implications for mathematics education. Computers in Mathematics Education: NCTM 1984 Yearbook. (pp. 118-126). Reston, VA: NCTM.
- Catlin, A. (1992). Standard basic programming with True Basic (2<sup>nd</sup> ed.). Englewood Cliffs, NJ: Prentice-Hall.
- Demana, F., Waits, B. K., & Clemens, S. R. (1992). College algebra: A graphing approach (2<sup>nd</sup> ed.). Reading, MA: Addison-Wesley.
- Demana, F., Waits, B. K., & Clemens, S. R. (1992). College algebra and trigonometry: A graphing approach (2<sup>nd</sup> ed.). Reading, MA: Addison-Wesley.
- Demana, F., Waits, B. K., & Clemens, S. R. (1994) Precalculus mathematics: A graphing approach (3<sup>rd</sup> ed.). Reading, MA: Addison-Wesley.
- Educational Testing Services (1996) A student guide to the AP calculus courses and examinations. Princeton, NJ: Author.
- Farrell, M.A. (1994). A bridge to the classroom: Implementing the NCTM standards. Dedham, MA: Janson.

- Finney, R. L., Thomas, G. B., Demana, F., & Waits, B. K. (1993) Calculus: A graphing approach. Reading, MA: Addison-Wesley.
- Halacy, D. S. (1962). Computers: The machines we think with. New York, NY: Harper and Row.
- Halvorson, M., & Rygmyr, D. (1991). Running MS-DOS QBasic. Redmond, WA: Microsoft Press.
- Hatfield, L. L. (1984). Toward comprehensive instructional computing in mathematics. Computers in Mathematics Education: NCTM 1984 Yearbook. (pp. 1-9) . Reston, VA: NCTM.
- Kelly, B. (1992). Programming the TI-81 and TI-85 graphics calculators to explore mathematics. Burlington, Ontario: Author.
- Kramer, S. L. (1997). "What we know about block scheduling and its effect on math instruction, Part 1." Bulletin: National Association of Secondary School Principals, 81(586), 18-42.
- Larson, R. E., Boswell, L., & Stiff, L. (1995). Geometry: An integrated approach. Lexington, MA: D. C. Heath.
- Larson, R. E., Boswell, L., Kanold, T., & Stiff, L. (1996). Passport to algebra and geometry: An integrated approach. Evanston, IL: McDougal-Littell.
- Larson, R. E., Hostetler, R. P., & Edwards, B.H. (1994). Calculus (5<sup>th</sup> ed.). Lexington, MA: D. C. Heath.
- Larson, R. E., Hostetler, R. P., & Edwards, B. H. (1993). College algebra: A graphing approach. Lexington, MA: D. C. Heath.
- Larson, R. E., & Hostetler, R. P. (1997). Precalculus ( 4<sup>th</sup> ed.) Boston, MA: Houghton-Mifflin.
- Larson, R. E., & Hostetler, R. P. (1997). Trigonometry ( 4<sup>th</sup> ed.). Boston, MA: Houghton-Mifflin.
- Larson, R. E., Kanold, T., & Stiff, L. (1997). Algebra 1: An integrated approach. Evanston, IL: McDougal-Littell.
- Larson, R. E., Kanold, T., & Stiff, L. (1997). Algebra 2: An integrated approach. Evanston, IL: McDougal-Littell

- Leiva, M. A., & Brown, R.G. (1997). Algebra 1: Explorations and Applications. Evanston, IL: McDougal-Littell.
- Leiva, M. A., & Brown, R.G. (1997). Algebra 2: Explorations and Applications. Evanston, IL: McDougal-Littell.
- Lucas, J. F., & Lucas, C. A. (1992). A guided tour of the TI-85 graphics programmable calculator. New York, NY: Ardsley House.
- Lund, C., & Anderson, E. (1996). Introduction to the TI-92: 37 experiments in precalculus and calculus. Urbana, IL: Mathware.
- Mandell, S. L., & Mandell, C. J. (1985). Computer science with Pascal for advanced placement students. St. Paul, MN: West.
- Nance, D. W. (1990). Fundamentals of Pascal: Understanding programming and problem solving (2<sup>nd</sup> ed.). St. Paul, MN: West.
- National Council of Teachers of Mathematics. (1989). Curriculum and evaluation standards for school mathematics. Reston, VA: Author.
- National Council of Teachers of Mathematics. (1991). Professional standards for teaching mathematics. Reston, VA: Author.
- National Council of Teachers of Mathematics. (1995). Assessment standards for school mathematics. Reston, VA: Author.
- Nichols, S. D. (1995). CBL explorations in calculus for the TI-82. Erie, PA: Meridian Creative Group.
- Papert, S. (1980). Mindstorms. New York, NY: Basic Books.
- Papay, K., Serum, L., & Donnelly, J. (1996). "Predicting the orbit of satellites with a TI-85 calculator." The Science Teacher, 63(4), p.33.
- Rich, N., Rose, J., & Gilligan, L. (1996). Mastering the TI-92: Explorations from algebra through calculus. Cincinnati, OH: Gilmar.
- Ruthven, K. (1992). Personal technology and classroom change: A British perspective. Calculators in Mathematics Education: NCTM 1992 Yearbook. (pp. 91-100). Reston, VA: NCTM.
- Shumway, R. J. (1984). Young children, programming, and mathematical thinking. Computers in Mathematics Education: NCTM 1984 Yearbook. (pp. 127-134). Reston, VA: NCTM.

- Smith, S. (1984). Microcomputers in middle school. Computers in Mathematics Education:NCTM 1984 Yearbook. (pp. 135-145) . Reston, VA: NCTM.
- Stwertka, A. (1987). Recent revolutions in Mathematics. New York, NY: Franklin Watts.
- Teachers Teaching with Technology (1997). <http://www.ti.com/calc/docs/shrt.htm>.
- Texas Instruments Inc. (1993). Introduction to programming on the TI-82 [Brochure]. Dallas, TX: Author.
- Texas Instruments Inc. (1994). Introduction to programming on the TI-85 [Brochure]. Dallas, TX: Author.
- Texas Instruments Inc. (1996). Introduction to programming on the TI-83 [Brochure]. Dallas, TX: Author.
- Texas Instruments Inc. (1993). TI-82 graphics calculator guidebook. Dallas, TX: Author.
- Texas Instruments Inc. (1996). TI-83 graphics calculator guidebook. Dallas, TX: Author.

## APPENDIXES



APPENDIX A

CALCULATOR TECHNICAL SPECIFICATIONS

TI - 82 TECHNICAL SPECIFICATIONS

- 8 - line by 16 character display.
- Advanced functions accessed by pull-down display menus.
- Operates on lists of numbers as well as single numbers.
- Number calculated to 14-digit accuracy and displayed with 10 digits plus a 2-digit exponent.
- Graphs rectangular functions, parametric expressions, polar expressions, and recursively defined sequences.
- Up to 10 graphing functions defined, saved, graphed, and analyzed at one time.
- Simultaneous graphing of more than one function.
- Sequence graphing mode shows both time series and cobweb/stair-step plot.
- 13 interactive zoom features.
- Function evaluation table shows numeric evaluation of functions in table format.
- Interactive evaluation of function values, roots, maximums, minimums, integrals, and derivatives.
- Matrix operations including inverse, determinant, transpose, augment, and elementary-row operations.
- List-based one- and two-variable statistical analysis, including median-median line, linear, logarithmic, exponential, power, quadratic polynomial, cubic polynomial, and quartic polynomial regression models.
- Box-and-whisker plots, histograms, scatter diagrams, and regression equation graphs.

- Programming capability with the number of programs limited only by available memory.
- 28K bytes of available memory.
- Link capabilities for data transfer through I/O port.
- Unit-to-unit link included.
- Overhead projector unit available.
- Accessories allow information to be transferred to a computer and to be printed or stored on disk.
- Compatible with Calculator-Based Lab (CBL) and Calculator-Based Ranger (CBR) systems to allow analysis of real world data.
- Calculator poster and keyboard transparency available.
- One year limited warranty.
- Volume Purchase Program - proofs of purchase redeemable for additional products.

## TI - 83 TECHNICAL SPECIFICATIONS

- 8-line by 16 character display.
- Advanced functions accessed through pull-down display menus.
- Real and complex numbers calculated to 14-digit accuracy and displayed with 10 digits plus a two digit exponent.
- Graphs 10 rectangular functions, 6 parametric expressions, 6 polar expressions, and 3 recursively-defined sequences.
- Up to 10 graphing functions defined, saved, graphed, and analyzed at the same time.
- Sequence graphing mode shows time series plot, cobweb/stair-step plot, and phase plots.
- User-defined list names. Lists up to 999 elements.
- 14 interactive zoom features.
- Function evaluation table shows numeric evaluation of functions in table format.
- Interactive analysis of function values, roots, maximums, minimums, integrals, and derivatives.
- 7 different graph styles for differentiating the look of each graph drawn.
- Horizontal and vertical split-screen options.
- Matrix operations including inverse, determinant, transpose, augment, reduced row echelon form, and elementary row operations. Convert matrices to lists and vice versa.

- List-based one-and two-variable statistical analysis including logistical, sinusoidal, median-median, linear, logarithmic, exponential, power, quadratic polynomial, cubic polynomial, and quartic polynomial regression models.
- 3 statistical plot definitions for scatter plots, xy-line plots, histograms, regular and modified box-and-whisker plots, and normal probability plots.
- Advanced statistical features including 9 hypothesis testing functions, 6 confidence interval functions, and one-way analysis of variance.
- Supplementary workbooks available.
- 15 probability distribution functions including Normal, Student-t, Chi-Square, Binomial, and Poisson.
- Business functions including Time-Value-of-Money (TVM), cash flows, and amortization. Full screen interactive editor for solving TVM problems.
- Interactive equation solver editor for solving for different variables in an equation.
- Alphabetic catalog of all calculator operations in one menu.
- Programming capability with the number of programs limited only by available memory.
- 27.3K bytes of available memory.
- Link capabilities for data transfer through I/O port. Transfer all data between two TI-83's and from a TI-82 to a TI-83, and lists L1-L6 from a TI-83 to a TI-82.
- Overhead projector unit available (same LCD as the TI-82).
- TI-Graph Link accessory allows information to be transferred to a computer and to be printed or stored to disk.

- Compatible with Calculator-Based Laboratory (CBL) and Calculator-Based Ranger (CBR) systems to allow analysis of real-world data.
- Calculator poster and keyboard transparency available.
- Volume Purchase Program - proofs of purchase redeemable for additional products.

APPENDIX B

FINAL VERSION OF THE PROGRAMMING MANUAL

**The High School Student's Guide to  
Programming the TI - 82 and TI - 83  
Graphics Calculators**

**Jim Bowen  
Stillwater High School  
Stillwater, Oklahoma**



## **Acknowledgments**

Special thanks to Steve DeBauge, Lenda Hill, and Guy Harris of Texas Instruments for their insightful comments and suggestions for improving the final version of this manual.

Thanks also to students in the Mu Alpha Theta math club at Stillwater High School and students in the Pi Mu Epsilon math club at Oklahoma State University for their evaluations of preliminary versions of this manual. Many of their suggestions have been included in the final version of the manual.

Jim Bowen  
Stillwater High School  
Stillwater, Ok.  
1997

## To The Students

The purpose of this manual is to assist high school students in writing original programs on the TI - 82 and TI - 83 graphics calculators. All the examples presented in this manual will work on both calculators, however the TI - 83 has a few additional features that the TI - 82 does not have. In certain instances these new features allow the TI - 83 programmer to perform operations not available to the TI - 82 programmer. This manual has been careful to include only techniques which work the same on both calculators.

Creating original programs on graphics calculators is similar to writing programs in other programming languages. Students who have completed a previous programming course in a language such as BASIC or PASCAL will recognize many similarities between the languages. The primary difference between editing programs on a graphics calculator and a normal desktop computer involves the manner in which the program statements are entered. On a desktop computer you generally type each program statement letter by letter, similar to using a word processor. Most of the program instructions on a graphics calculator, however, are entered by selecting the appropriate instruction from various menus within the calculator. In one sense, this makes programming the calculator easier because it minimizes the typing required. Initially, however, it makes programming the calculator more frustrating because you must learn in which menu to search to find a particular instruction, and the location of these instructions is not always obvious. As you gain experience programming the graphics calculator, you will quickly become adept at locating the required instruction within the various menus.

This manual assumes that you have already been introduced to, and are familiar with, the basic operations of the calculator. This manual will attempt to extend that basic knowledge by presenting descriptions of the programming instructions available on the calculators. This will include the use of numerous examples and sample programs to show you how these individual instructions may be combined into complete programs. This will allow you to expand the capabilities of your calculator well beyond the built in functions.

You should be advised, however, that this manual is intended to complement the calculator guidebook, not replace it! You will want to keep your calculator guidebook handy because you will probably make frequent reference to it to recall details about specific calculators functions as you write your programs. Particularly useful is the "menu map" located near the back of the guidebook. This "menu map" shows the menu location of every instruction available on the calculator.

# **INTRODUCTION**

**Throughout this manual, the following format has been applied:**

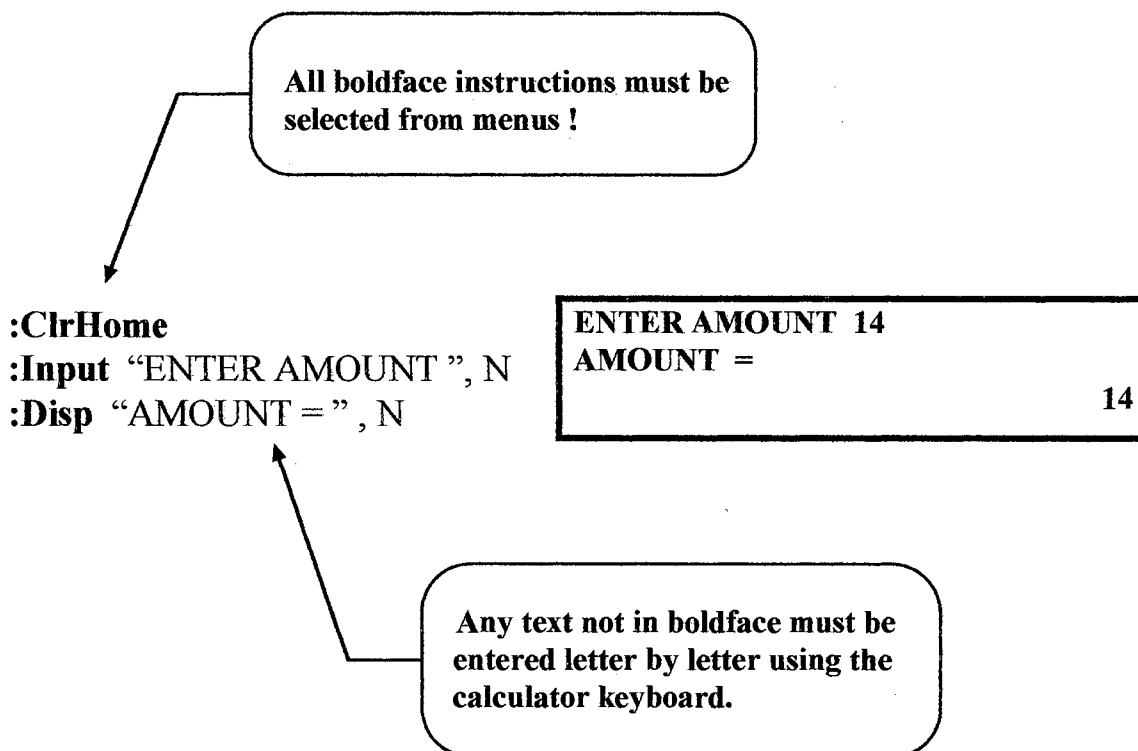
First, the programming instructions you enter into the calculator are shown on the left side of the page. The output to the screen when the program is executed is shown in the shaded area on the right side of the page.

**IMPORTANT:**

Notice that some text is in **boldface type** while other text is in normal type.

Instructions which are presented in **boldface type** represent instructions which must be selected from menus. You do not type them in letter by letter !

Any text which is not in **boldface** must be entered letter by letter by using the [ ALPHA ] key and choosing the appropriate alphabetic character from the calculator keyboard.



## To Enter and Run Programs on the Graphics Calculator

### To Create a New Program

Assume you wanted to write a simple program named **DEMO** that just printed the word “HELLO” on the home screen.

Follow these steps:

- (1) Hit the [PRGM] key to enter the program mode.
- (2) Then hit the [▶] key twice to highlight [NEW] for a new program, then hit [ENTER].
- (3) Type DEMO to name the program, then hit [ENTER].
- (4) Now enter your program statements:
  - (a) Hit the [PRGM] key to view the programming menus.
  - (b) Hit [▶] to highlight [I/O].
  - (c) Hit [▼] to highlight **8: ClrHome**, then hit [ENTER] to move the **ClrHome** instruction from the menu into your program.
  - (d) Hit [ENTER] to start a new line beginning with a : .
  - (e) Hit [PRGM] to view the programming menus.
  - (f) Hit [▶] to highlight [I/O].
  - (g) Hit [▼] to highlight **3: Disp**, then hit [ENTER] to move the **Disp** instruction from the menu into your program.
  - (h) Lock the calculator in the alpha mode by hitting the [2<sup>nd</sup>], [ALPHA] keys. This enables you to type letters from the keyboard.
  - (i) Now type “HELLO”, then hit [ENTER] to complete this statement.
- (5) The program is now finished, so hit [2<sup>nd</sup>], [QUIT] to exit the program mode. The program is automatically stored under the name **DEMO**.

### To Run the Program DEMO

- (1) Hit the [PRGM] key to enter the program mode.
- (2) Hit [▼] to highlight the program **DEMO**, then hit the [ENTER] key to load the program.
- (3) Now hit [ENTER] again to run the program.
- (4) The program and output should have looked as follows :

```
PROGRAM: DEMO
:ClrHome
:Disp "HELLO"
```

<b>HELLO</b>	Done
--------------	------

## To Edit Existing Programs on the Graphics Calculator

The only time you choose “NEW” on the program menu is when you initially create a new program.

If you want to make changes to an existing program, follow these steps :

- (1) Hit the **[PRGM]** key to enter the program mode.
- (2) Hit the **[▶]** key once to highlight **[EDIT]** . This puts you in the editing mode to make changes to existing programs.
- (3) Hit the **[▼]** key to highlight the program name you want to edit, then hit the **[ENTER]** key.
- (4) You should now see a listing of the program, and you can make whatever changes you like.
- (5) After making the required changes, hit **[2<sup>nd</sup>]** , **[QUIT]** and the new version of your program is automatically stored under the same program name.

**tip:** If you make typing mistakes while editing, you can use the **[DEL]** key, or the **[2<sup>nd</sup>]** , **[INS]** keys to make corrections.

**tip:** To add a new line between existing lines in a program, first use the arrow keys to position the cursor where you want the new line to begin, then hit **[2<sup>nd</sup>]** , **[INS]** , **[ENTER]** . This will open a space for you to insert the new line.

**tip:** There may be times when you will want to save a modified copy of a program under a different name while preserving the original program. To do this, first create a new program and choose a new name for the program. Then, on the first line of the new program, hit **[2<sup>nd</sup>]** , **[RCL]** , **[PRGM]** , **[▶]** , **[▶]** , to highlight **[EXEC]** . Then use **[▼]** to highlight the name of the original program and hit **[ENTER]** . This moves a copy of the entire original program into the new program . Now you can change the copy program and save it under the new name while preserving the original program under the old name.

# **PROGRAMMING INSTRUCTIONS**

## The PRGM CTL and I / O menus

When writing programs, there are two primary menus that contain most of the calculator's basic programming instructions. Both of these programming menus are accessed by hitting the **[PRGM]** key while editing a program. The two menus are the control menu **[CTL]**, and the input / output menu **[I / O]**. There are some minor differences between the menus on the **TI - 82** and the **TI - 83**, however the discussions in this manual will only involve menu items that are common to both calculators. The menu items that will be discussed in this manual are reproduced here for your reference.

### CTL menu

**: If**  
**: Then**  
**: Else**  
**: For (**  
**: While**  
**: Repeat**  
**: End**  
**: Pause**  
**: Lbl**  
**: Goto**  
**: Menu (**  
**: prgm**  
**: Return**  
**: Stop**

### I / O menu

**: Input**  
**: Prompt**  
**: Disp**  
**: DispGraph**  
**: DispTable**  
**: Output (**  
**: ClrHome**  
**: ClrTable**

Generally, if an instruction involves input or output operations it will be found in the **[I / O]** menu. If an operation involves control of the flow of the program, it will be found in the **[CTL]** menu.

These are the two primary programming menus, but remember, when writing programs you may use almost every instruction or function available in all menus on the calculator. This is another good reason to keep the calculator guidebook "menu map" handy when writing programs !



## **ClrHome Instruction**

The **ClrHome** instruction clears the text screen and places the cursor in the upper left hand corner.

Note: The **ClrHome** instruction is found in the [ PRGM ], [ I/O ] menu.

## **Pause Instruction**

The **Pause** instruction is used to temporarily delay execution of a program so you will have time to view text or graphs screens. When you are ready to continue, just hit the [ ENTER ] key.

Note: The **Pause** instruction is found in the [ PRGM ], [ CTL ] menu.

## **Stop Instruction**

The **Stop** instruction permanently stops the execution of a program and displays the home screen. It may be placed at any point in the program where you want the program to end.

Note: The **Stop** instruction is found in the [ PRGM ], [ CTL ] menu.

## **DispGraph Instruction**

The **DispGraph** instruction displays the current graph. It is frequently used to switch the calculator from text mode to graphics mode within a program.

Note: The **DispGraph** instruction is found in the [ PRGM ], [ I/O ] menu.

## **Disp** - The Display Instruction

This instruction is used to print either text or numeric information on the screen, or to return to the home screen from another display mode such as graphics.

### EXAMPLES:

**tip:** Prior to using the **Disp** instruction you might elect to use the **ClrHome** instruction so you begin with a clean home screen.

1. To print text information, just enclose the message you want to print within double quotes:

```
:ClrHome
:Disp "THIS IS A DEMO"
```

**THIS IS A DEMO**

2. A space in a text message is obtained by hitting [ ALPHA ] , [ 0 ] .

**tip:** Instead of hitting the [ ALPHA ] key prior to each character, you can lock the calculator in the alpha mode by hitting [ 2<sup>nd</sup> ] , [ ALPHA ] .

3. The maximum number of characters you can print across the screen is 16.
4. If the text message is longer than one line you must use multiple **Disp** instructions to print it. For example, to print "This sentence requires more than one line":

```
:ClrHome
:Disp "THIS SENTENCE"
:Disp "REQUIRES MORE"
:Disp "THAN ONE LINE"
```

**THIS SENTENCE  
REQUIRES MORE  
THAN ONE LINE**

5. To print a blank line, use two successive quotes in a **Disp** instruction:

```
:ClrHome
:Disp "THIS SENTENCE"
:Disp " "
:Disp "REQUIRES MORE"
:Disp " "
:Disp "THAN ONE LINE"
```

**THIS SENTENCE  
REQUIRES MORE  
THAN ONE LINE**

**Disp - Continued:**

**tip:** To keep from splitting words while printing, don't allow the trailing quote to extend more than one character beyond the initial quote. If a word does extend beyond the initial quote, then end the first part of the message between words and continue the message on the next line using a new **Disp** instruction.

6. You can display both text and numeric information using a single **Disp** instruction by placing a comma between each value:

```
:ClrHome
:7.54 → N
:Disp "THE VALUE IS", N
```

```
THE VALUE IS
7.54
```

7. You may include an arithmetic expression within a **Disp** instruction. The expression will be evaluated before the final value is displayed:

```
:ClrHome
:Disp "THE VALUE
IS", (7+9)/3
```

```
THE VALUE IS
5.333333333
```

8. You can control the number of decimal places by using the **Round** function within a **Disp** instruction:

```
:ClrHome
:8/3 → N
:Disp N
:Disp Round (N,3)
```

```
2.666666667
2.667
```

9. If you want to display a result in fractional form, use the **▸ Frac** function within the **Disp** instruction:

```
:ClrHome
:8/3 → N
:Disp N
:Disp N ▸ Frac
```

```
2.666666667
8/3
```

**Disp - Continued:**

10. You can fix the number of decimals to be displayed throughout a program by placing the instruction **Fix #** at the beginning of your program.

```

:Fix 2
:ClrHome
: 3.5678765 → A
: 234.6758878 → B
: 7.878912 → C
:Disp A , B , C
:Float

```

	3.57
	234.68
	7.88

**Note:** To enter the **Fix** instruction you must be in the program edit mode. Then hit **[MODE]**, **[▼]** to **FLOAT**, **[▶]** to **2**, then hit **[ENTER]**.

**Note:** The calculator will remain fixed at 2 decimals even after your program is finished executing unless you reset the mode back to **Float** at the end of the program.

11. If you want the output to be displayed horizontally instead of vertically, you can enclose the output in braces { } which prints the values in list form:

```

:ClrHome
: 46 → A
: 8 → B
: 234 → C
:Disp { A , B , C }

```

	{ 46 8 234 }
--	--------------

## Output - The Output Instruction

The **Output** instruction gives you more control over the positioning of output on the screen than the **Disp** instruction. You can specify the exact row number and column number where you want the output to begin. Remember, you have 8 horizontal rows down the screen and 16 vertical columns across the screen.

The format of the instruction is: **Output ( row #, col #, value )**

### EXAMPLES:

1. To place the words "HI THERE" near the center of the screen:

```
:ClrHome
:Output (4, 7, "HI")
:Output (5, 5, "THERE")
```

**HI**  
**THERE**

2. The **Output** instruction may be used to display both text and numeric information on the same line:

```
:ClrHome
:95 → M
:17 → F
:Output (3, 1, "HER AGE IS ")
:Output (3, 12, F)
:Output (5, 1, "HIS SCORE IS ")
:Output (5, 14, M)
```

**HER AGE IS 17**

**HIS SCORE IS 95**

**tip:** When using two Output instructions to print on the same line, count the number of characters in the first Output instruction and this will tell you in which column to begin the second Output instruction.

## Input - The Input Instruction

The **Input** instruction allows the user to enter values into the calculator and store those values in specified variables.

The format of the instruction is either: **Input** "TEXT",variable  
or **Input** variable

### EXAMPLES:

1. In the most basic form, allow the user to enter a number, store that number in the variable "N", and then display the result:

```
:ClrHome  
:Input N  
:Disp N
```

? 7.2	7.2
-------	-----

Notice that the **Input** instruction displays a "?" and waits for the user to enter a value.

2. Normally, when you use an **Input** instruction you will want to display a message telling the user what to enter. This message is called a "prompt", and there are several ways to display a prompt:

- (A) The prompt may be enclosed within quotes in the **Input** instruction. Include a comma before the variable name.

```
:ClrHome  
:Input "ENTER A VALUE ", N  
:Disp N
```

ENTER A VALUE 5.2	5.2
-------------------	-----

Notice the question mark has been replaced by a flashing cursor.

- (B) The prompt may be placed in a **Disp** instruction prior to the **Input** instruction.

```
:ClrHome  
:Disp "ENTER A VALUE"  
:Input N  
:Disp N
```

ENTER A VALUE	57300
? 5.34e4	

Notice the question mark now prints on the line following the prompt.

**Input - Continued:**

3. The **Input** instruction also allows the user a method of assigning functions to the **Y=** variables within a program. For example, suppose you wanted to assign the function  $X^2 - 9$  to the variable **Y1**. To do this, use **Y1** as the variable in your **Input** instruction and make sure you enclose " $X^2 - 9$ " within quotes as you enter it in the program.

**Notes:**

- (a) To select **Y1**, hit [ 2<sup>nd</sup> ], [ **y-vars** ], [ **Function** ], and then select **Y1**.  
 (b) To select =, hit [ 2<sup>nd</sup> ], [ **TEST** ], and select =.

**:ClrHome**

**:Disp** "ENTER FUNCTION"

**:Disp** "(WITHIN QUOTES)"

**:Input** "Y1 =", Y1

<b>ENTER FUNCTION</b> <b>(WITHIN QUOTES)</b> <b>Y1 = "X<sup>2</sup> - 9"</b>
--

Remember, you must enclose the function within quotes as you enter it !

Now hit the [ **Y=** ] key and you will see that  $X^2 - 9$  has been assigned to **Y1**.

## Prompt - The Prompt Instruction

The **Prompt** instruction is similar to the **Input** instruction. It also allows the user to enter values into the calculator and store the values in variables. One or more variables may be included within a single **Prompt** instruction. The **Prompt** instruction displays the variable name, including =? , when executed. However, unlike the **Input** instruction, you cannot include a prompt message to the user within a **Prompt** instruction.

The format of the instruction is: **Prompt** var1, var2, ...

### EXAMPLES:

- The primary difference between the **Prompt** and **Input** instructions is that the **Prompt** automatically displays the variable name while the **Input** instruction does not:

```
:ClrHome
:Input "ENTER A VALUE ", N
:Prompt N
```

ENTER A VALUE 17  
N=? 17

Notice the difference between the two instructions.

- You may use one or more variables ( separated by commas ) in a single **Prompt** statement:

```
:ClrHome
:Prompt A, B, C
```

A=? 4  
B=? 5  
C=? 6

- You may also use the **Prompt** instruction to assign values to the built-in calculator variables. Suppose you want to change the **Window** variables to +/- 20:

**Note:** To select the variables, hit [ vars ], [ window ], [ Xmin ] etc.

```
:ClrHome
:Prompt Xmin, Xmax
:Prompt Ymin, Ymax
```

Xmin =? -20  
Xmax =? 20  
Ymin =? -20  
Ymax =? 20



## If - Then Instruction

The **If - Then** instruction allows a program to make decisions and then branch to alternative courses of action.

The format of the **If - Then** instruction is :

```

: If condition
: Then
:   statement 1
:   statement 2
:   etc.
: End

```

**Note:** **If**, **Then**, and **End** are all separate statements and you must select each one individually from the [ PRGM ], [ CTL ] menu.

The **If - Then** instruction works by first evaluating the condition to see if it is true or false. If the condition is true, the block of statements between **Then** and **End** are executed. If the condition is false, the block of statements between **Then** and **End** are skipped, and execution of the program resumes at the first statement following the **End** statement.

The condition is evaluated using the following relational operators :

=	<b>Equal to</b>
≠	<b>Not equal to</b>
>	<b>Greater than</b>
≥	<b>Greater than or equal to</b>
<	<b>Less than</b>
≤	<b>Less than or equal to</b>

**Note:** To select relational operators, hit [ 2<sup>nd</sup> ], [ TEST ]

**If - Then - Continued:****EXAMPLES:**

1. Write a program which will allow you to enter your age and then tell you if you are old enough to drive:

```

:ClrHome
:Input "ENTER AGE ", A
:If A ≥ 16
:Then
:Disp "YOU CAN DRIVE"
:Disp "A CAR"
:End
:Disp " "
:Disp "FINISHED"

```

```

ENTER AGE 18
YOU CAN DRIVE
A CAR

FINISHED

```

Now observe what happens when your age is less than 16.

```

ENTER AGE 14

FINISHED

```

Notice that the block of statements between **Then** and **End** are only executed if the condition is true, but the word "FINISHED" prints in either case.

2. You may use compound conditions in **If - Then** statements by using Boolean operators. The Boolean operators are :

```

AND - True only if both parts are true
OR  - True if at least one part is true
XOR - True if only one part is false
NOT - True if the original condition is false

```

**Note:** To select the Boolean operators hit  
[ 2<sup>nd</sup> ], [ TEST ], [ ▶ ], [ LOGIC ]

We will now modify example 1 to offer a discount to students who are old enough to drive and who have a 3.0 GPA:

**If - Then - Continued:**

EXAMPLE 2 continued:

```

:ClrHome
:Input "ENTER AGE ", A
:Input "ENTER GPA ", G
:If A ≥ 16 and G ≥ 3.0
:Then
:Disp "YOU CAN DRIVE"
:Disp "WITH A DISCOUNT"
:End
:Disp " "
:Disp "FINISHED"

```

```

ENTER AGE 18
ENTER GPA 3.5
YOU CAN DRIVE
WITH A DISCOUNT

FINISHED

```

Now run the program again and observe what happens if one of the conditions is false.

3. In situations where only a single statement is to be executed if the condition is true, then you may omit the **Then** and **End** statements. If the condition is true, the single statement following the **If** instruction is executed. If the condition is false, execution skips to the second statement following the **If** instruction.

```

:ClrHome
:Input "ENTER YOUR AGE ", A
:If A ≥ 16
:Disp "YOU CAN DRIVE"
:Disp "FINISHED"

```

```

ENTER YOUR AGE 18
YOU CAN DRIVE
FINISHED

```

Now observe what happens when your age is less than 16.

```

ENTER YOUR AGE 14
FINISHED

```

## If - Then - Else Instruction

The **If - Then - Else** instruction is similar to the **If - Then** instruction but there is one important additional feature. The **If - Then - Else** instruction executes one block of statements if the condition is true, and it executes a different block of statements if the condition is false.

The format of the **If - Then - Else** instruction is :

```

:If condition
:Then
: statements if true
:Else
: statements if false
:End
```

The **If - Then - Else** instruction works by first evaluating the condition to see if it is true or false. If the condition is true, the first block of statements ( the statements between **Then** and **Else** ) are executed. If the condition is false, the second block of statements ( the statements between **Else** and **End** ) are executed.

### EXAMPLE:

1. Modify the previous **If - Then** example by using **If - Then - Else** to tell if :
  - (a) You are old enough to drive
  - (b) You are too young to drive

```

:ClrHome
:Input "ENTER AGE ", A
:If A ≥ 16
:Then
:Disp "YOU CAN DRIVE"
:Disp "A CAR"
:Else
:Disp "YOU ARE TOO"
:Disp "YOUNG"
:End
:Disp " "
:Disp "FINISHED"
```

```

ENTER AGE 14
YOU ARE TOO
YOUNG

FINISHED
```

Now run the program again and observe what happens if the condition is true.

## Lbl and Goto Instructions

The **Lbl** and **Goto** instructions are used together to control the possible branches that the flow of a program may follow.

The format of the **Lbl** instruction is :     **Lbl label**

The format of the **Goto** instruction is :     **Goto label**

**Note:** In both instructions, label is a single letter or number.

The **Lbl** and **Goto** instructions are found in the [ PRGM ], [ CTL ] menu.

When the program encounters a **Goto** instruction, the flow of the program immediately branches to the line where the specified label is located.

### EXAMPLES:

1. One common use of the **Lbl** and **Goto** instructions is to allow the user to run a program more than once. To do this, put the **Lbl** instruction at the beginning of the program, and put the **Goto** instruction in an **If - Then** instruction near the end of the program.

```

:Lbl A
:Disp " "
:Input "ENTER A VALUE ", X
:Disp X
:Disp "RUN AGAIN ? "
:Disp "(1) YES"
:Disp "(2) NO"
:Input "ENTER 1 OR 2: ", C
:If C = 1
:Goto A
:Disp " "
:Disp "FINISHED"

```

```

ENTER A VALUE 8
8
RUN AGAIN ?
(1) YES
(2) NO
ENTER 1 OR 2: 1

ENTER A VALUE 9
9
RUN AGAIN ?
(1) YES
(2) NO
ENTER 1 OR 2: 2

FINISHED

```

Notice if the user enters a 1, control of the program immediately transfers from the **Goto A** instruction to the **Lbl A** instruction and the entire program is executed again. If the user enters a 2, then the **Goto A** instruction is skipped and the program ends.

Lbl and Goto - Continued:

2. Programs are often written which contain several parts and the user is given the choice of which part to run. The **Lbl** and **Goto** instructions may be combined with the **If - Then** instruction to set up a menu to control which part of a program is executed :

```

:ClrHome
:Disp "PLEASE CHOOSE:"
:Disp "(1) PART A"
:Disp "(2) PART B"
:Disp "(3) PART C"
:Input "ENTER 1, 2, OR 3:", C
:Disp ""
:If C = 1
:Goto 1
:If C = 2
:Goto 2
:If C = 3
:Goto 3
:
:Lbl 1
:Disp "THIS IS PART A"
:Goto 4
:
:Lbl 2
:Disp "THIS IS PART B"
:Goto 4
:
:Lbl 3
:Disp "THIS IS PART C"
:
:Lbl 4
:Disp ""
:Disp "FINISHED"

```

```

PLEASE CHOOSE:
(1) PART A
(2) PART B
(3) PART C
ENTER 1, 2, OR 3: 1

THIS IS PART A

FINISHED

```

```

PLEASE CHOOSE:
(1) PART A
(2) PART B
(3) PART C
ENTER 1, 2, OR 3: 2

THIS IS PART B

FINISHED

```

```

PLEASE CHOOSE:
(1) PART A
(2) PART B
(3) PART C
ENTER 1, 2, OR 3: 3

THIS IS PART C

FINISHED

```

**Note:** You must insert a **Goto 4** instruction at the end of each part to prevent following parts from also being executed.

## Menu Instruction

Since menus are so frequently used to control the flow of programs, the TI - 82 includes a separate **Menu** instruction.

The format of the **Menu** instruction is :

**Menu** ( "TITLE" , "TEXT 1" , label 1 , "TEXT 2" , label 2 , ... )

where: "TITLE" - This is the title of the menu.  
 "TEXT 1", "TEXT 2", etc. - These are the menu choices.  
 label 1, label 2, etc. - These are the corresponding labels to branch to.

**Note:** The maximum number of menu choices is seven.

### EXAMPLE:


This example will show how one **Menu** instruction can be used to replace several of the **If - Then** instructions used in the previous example menu :

```

:ClrHome
:Menu ( "PLEASE CHOOSE:", "PART A", 1, "PART B", 2, "PART C", 3 )
:
:Lbl 1
:Disp "THIS IS PART A"
:Goto 4
:
:Lbl 2
:Disp "THIS IS PART B"
:Goto 4
:
:Lbl 3
:Disp "THIS IS PART C"
:
:Lbl 4
:Disp " "
:Disp "FINISHED"

```

**PLEASE CHOOSE:**  
**1: PART A**  
**2: PART B**  
**3: PART C**

Now choose **PART B** by either:  
 (a) **Hitting 2 on the keyboard**  
 (b) Use [  ] to highlight **PART B**,  
 and hit [ **ENTER** ]

**THIS IS PART B**

**FINISHED**

## For Loop Instruction

Although it may be used in several ways, the **For** loop is generally used to repeat a block of statements a fixed number of times.

The format of the **For** loop is :

```
:For ( var, initial, final, stepsize )
:
:   one or more statements
:
:End
```

where: **var** - This is the loop control variable.  
**initial** - This is the initial value of the loop control variable.  
**final** - This is the final value of the loop control variable.  
**stepsize** - This is the incremental value.

**Note:** The **For** instruction is found in the [ PRGM ], [ CTL ] menu.

The **For** loop works by first setting the loop control variable equal to the initial value. The loop then repeats all the statements between **For** and **End**, incrementing the control variable with each pass through the loop. The amount of the increment is given by the stepsize value. The looping action finally stops when the value of the control variable exceeds the final value. Then execution of the program resumes at the first statement following the **End** statement.

**Note:** The stepsize is optional and if omitted it is assumed to be +1

### EXAMPLES:

1. Display the word "HELLO" five times using a **For** loop:

```
:ClrHome
:For ( A , 1, 5 )
:Disp "HELLO"
:End
```

```
HELLO
HELLO
HELLO
HELLO
HELLO
```

Notice that when the stepsize is omitted it becomes 1 by default !



**For loop - Continued:**

2. You can print the loop control variable as well:

```

:ClrHome
:For ( A, 1, 5 )
:Disp A
:End

```

1
2
3
4
5

3. The stepsize controls the increment value:

```

:ClrHome
:For ( K, 1, 2, .2 )
:Disp K
:End

```

1
1.2
1.4
1.6
1.8
2

4. You can run the loop backwards using a negative stepsize:

```

:ClrHome
:For ( M, 5, 1, -1 )
:Disp M
:End

```

5
4
3
2
1

5. You can allow the user to control the number of times the loop is repeated by using input variables instead of constants in the **For** instruction:

```

:ClrHome
:Input "ENTER FIRST ", F
:Input "ENTER LAST ", L
:Input "ENTER STEP ", S
:ClrHome
:For ( N, F, L, S )
:Disp N
:End

```

ENTER FIRST 3
ENTER LAST 5
ENTER STEP .5

3
3.5
4
4.5
5



## While Loop Instruction

The **While** loop is another looping structure that repeats a block of statements within a loop.

The format of the **While** loop is :

```
:While condition
:
: statements if true
:
:End
```

**Note:** The **While** instruction is found in the [ PRGM ], [ CTL ] menu.

The **While** loop works by first evaluating the condition to see if it is true or false. While the condition is true, the block of statements inside the loop are executed repeatedly. When the condition becomes false, the looping action stops and the program transfers execution to the first statement following the **End** statement.

The **While** loop is generally used in situations where you may not know in advance exactly how many times the loop will be executed.

### EXAMPLES:

1. A **While** loop may be used to count by performing the following steps:
  - (a) First, set the counter variable ( **C** ) equal to an initial value.
  - (b) Increment the counter each time through the loop.
  - (c) Stop when the counter exceeds a specific final value.

Now use a **While** loop to print the numbers from 1 to 5 :

```
:ClrHome
: 1 → C
:While C ≤ 5
:Disp C
:C + 1 → C
:End
:Disp "FINISHED"
```

	1
	2
	3
	4
	5
<b>FINISHED</b>	

**While Loop Continued:**

2. When using **While** loops, you must include a statement within the loop that changes the value of the loop control variable. If this statement is omitted the program may enter an infinite loop which would run forever without stopping. Using the previous example, observe the effect of omitting the statement :  $C + 1 \rightarrow C$  .

<b>:ClrHome</b>	1
<b>:1 <math>\rightarrow</math> C</b>	1
<b>:While C <math>\leq</math> 6</b>	1
<b>:Disp C</b>	1
<b>:End</b>	1
	etc

Notice that if there is no statement to change the value of  $C$  , the condition will always be true and the loop will never end !

**Note:** To interrupt a program while it is running, hit the **[ON]** key.

3. A **While** loop may be used to force the user to repeat an Input instruction until a correct value is entered:

<b>:ClrHome</b>	<b>5 + 2 = 8</b>
<b>:Input " 5 + 2 = ", A</b>	<b>WRONG</b>
<b>:While A <math>\neq</math> 7</b>	<b>TRY AGAIN</b>
<b>:Disp "WRONG"</b>	<b>5 + 2 = 4</b>
<b>:Disp "TRY AGAIN"</b>	<b>WRONG</b>
<b>:Input " 5 + 2 = ", A</b>	<b>TRY AGAIN</b>
<b>:End</b>	<b>5 + 2 = 7</b>
<b>:Disp "CORRECT"</b>	<b>CORRECT</b>

The first **Input** instruction in this program is called a "priming" **Input** instruction because it "primes the loop". The **Input** statement inside the loop is repeated until the user enters the correct answer.

## Repeat Loop Instruction

The **Repeat** loop is another looping structure that repeats a block of statements within a loop.

The format of the **Repeat** loop is :

```

:Repeat condition
:
: statements if false
:
:End

```

**Note:** The **Repeat** instruction is found in the [ PRGM ], [ CTL ] menu.

The **Repeat** loop is similar to the **While** loop except it waits until the **End** statement is encountered before it evaluates the condition. If the condition is false, the block of statements inside the loop are repeatedly executed until the condition becomes true. If the condition becomes true, the looping action stops and the program transfers execution to the first statement following the **End** statement.

There is one primary difference between the **While** and **Repeat** loops. The **While** loop is known as a “pre-test” loop which means it evaluates the condition before executing the loop. The **Repeat** loop is known as a “post-test” loop which means it evaluates the condition after executing the loop and encountering the **End** statement. The effect of this difference is that the statements within a **Repeat** loop will always be executed at least once regardless of whether the **Repeat** condition is true or false, whereas it is possible that the statements within a **While** loop may never be executed at all !

### EXAMPLES:

1. A **Repeat** loop may also be used to count. This example will be similar to the previous **While** loop example 1 except that a **Repeat** loop will replace the **While** loop.

Use a **Repeat** loop to print the numbers from 1 to 5 :

```

:ClrHome
: 1 → C
:Repeat C > 5
:Disp C
: C + 1 → C
:End

```

	1
	2
	3
	4
	5

## prgm and Return Instructions

The **prgm** and **Return** instructions are used together to allow one program to be called and executed from another program.

The following example shows how the **prgm** and **Return** instructions work :

```
PROGRAM: ONE
:ClrHome
:Disp " THIS IS ONE "
: prgmTWO
:Disp " BACK TO ONE "
```

```
PROGRAM: TWO
:Disp " THIS IS TWO "
:Return
```

**Note:** You must not include a space in the **prgmTWO** statement.

In this example, program ONE will call program TWO . The user initially runs program ONE . When the **prgm** instruction is encountered, program TWO is called and execution is transferred to the first line of program TWO. After program TWO is executed, the **Return** instruction in program TWO transfers execution back to program ONE at the line following the **prgm** instruction. The remainder of program ONE is then executed.

The output from the example above is :

```
THIS IS ONE
THIS IS TWO
BACK TO ONE
```

Students with previous programming experience will recognize this structure as being similar to subroutines or sub-programs in other languages.

**tip:** The **Return** statement is optional. If the **Return** statement is omitted, an implied return is considered to exist at the end of the called program, and execution will return to the main program.

## rand Instruction

The **rand** instruction generates random numbers between 0 and 1. One of the main uses of the **rand** instruction is to program games on the calculator.

Generally, you will need random integers ranging between specific high and low integer values. You can use the following formula to generate random integers between chosen high and low integer values:

$$\text{: int ( rand * range ) + low \# } \rightarrow R$$

where: **int** - converts a decimal number to an integer  
**rand** - generates a random number between 0 and 1  
**range** - the range of numbers = ( high # - low # ) + 1  
**low #** - the lowest possible integer in the range  
**R** - variable used to store the random integer

**Note:** The **int** instruction is found in the  
[MATH], [▶], [NUM], [▼], [int] menu.

The **rand** instruction is found in the  
[MATH], [▶], [▶], [▶], [PRB], [▼], [rand] menu.

### EXAMPLE:

Suppose you wanted to write a program to simulate the roll of two dice. In this case the high integer would be 6 and the low integer would be 1. The range is :

$$\text{range} = (\text{high \#} - \text{low \#}) + 1 = (6 - 1) + 1 = 6$$

This program will simulate rolling two dice 5 times:

```
:ClrHome
:For ( X , 1 , 5 )
: int ( rand * 6 ) + 1 → A
: int ( rand * 6 ) + 1 → B
:Disp { A , B }
:End
```

{ 3 5 }
{ 1 6 }
{ 4 4 }
{ 4 2 }
{ 5 6 }

**Note:** Placing the output within { } will display them in list form.

## fMin and fMax Instructions

The **fMin** instruction returns the value of **X** which yields the minimum value of a given function on a specified interval. The **fMax** instruction returns the value of **X** which yields the maximum value of a given function on a specified interval.

The format of the **fMin** or **fMax** instruction is:

**fMin ( EXP , X , L , R )**  
**fMax ( EXP , X , L , R )**

where: **EXP** = An expression or **Y** variable.  
**X** = The independent variable.  
**L** = The left endpoint of the range of **X**.  
**R** = The right endpoint of the range of **X**.

### EXAMPLE:

1. One of the most beneficial uses of the **fMin** and **fMax** instructions allows the calculator to automatically determine the values for **Ymin** and **Ymax** to insure a given function is entirely contained within the viewing window for specified values of **Xmin** and **Xmax**. This automatic scaling feature may be demonstrated by the following:
  - a. First, use **fMin** to determine the **X** value where the function minimum occurs.
  - b. Then evaluate the function at that **X** value and store the result in **Ymin**.
  - c. Now use **fMax** to determine the **X** value where the function maximum occurs.
  - d. Then evaluate the function at that **X** value and store the result in **Ymax**.

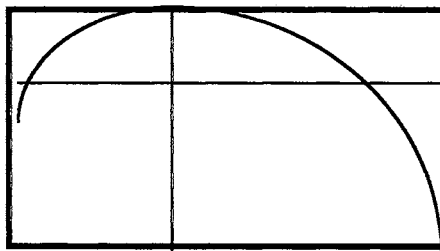
```

:ClrHome
:Disp "ENTER FUNCTION"
:Disp " ( IN QUOTES )"
:Input " Y1 = ", Y1
:Prompt Xmin
:Prompt Xmax
:ClrHome
:Disp " SCALING GRAPH "
:Disp " PLEASE WAIT "
:fMin( Y1 , X , Xmin , Xmax )
→ A
:Y1 ( A ) → Ymin
:fMax( Y1 , X , Xmin , Xmax )
→ B
:Y1 ( B ) → Ymax
:ClrHome
:DispGraph

```

**ENTER FUNCTION**  
**( IN QUOTES )**  
Y1 = " 3 - X<sup>2</sup> "  
Xmin = ? -3  
Xmax = ? 4

**SCALING GRAPH**  
**PLEASE WAIT**





## The Draw Menu Instructions

The instructions contained in the **DRAW** menu are particularly useful for adding information to the graphics screen of the calculator. By utilizing these instructions within a program, you can draw lines, circles, and other figures, including text messages on a graphics screen.

The **DRAW** instructions are contained in the [2<sup>nd</sup>], [DRAW] menu.

### **ClrDraw** Instruction (DRAW MENU)

The **ClrDraw** instruction is used to clear any existing drawings from the graphics screen. It performs a similar function to the **ClrHome** instruction for the home screen.

## Line Instruction (DRAW MENU)

The **Line** instruction is used to draw or erase lines on the graphics screen.

The format of the **Line** instruction is:

To draw a line:

**: Line ( X1, Y1, X2, Y2 )**

To erase a line:

**: Line ( X1, Y1, X2, Y2, 0 )**

The **Line** instruction draws a line from the point ( **X1, Y1** ) to the point ( **X2, Y2** ). This instruction may also be used to erase an existing line between the points by adding a "0" after the last coordinate.

### EXAMPLE:

1. Allow the user to enter the coordinates of two points ( **X1, Y1** ) and ( **X2, Y2** ) and then draw a line between the two points. Have the program pause until the user hits **[ENTER]**, then erase the line.

```

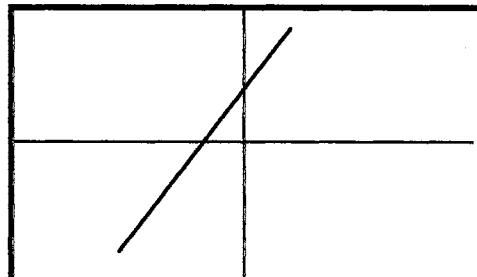
:ClrHome
:Disp "ENTER ( X1, Y1 )"
:Disp "AND ( X2, Y2 )"
:Input " X1 = ", A
:Input " Y1 = ", B
:Input " X2 = ", C
:Input " Y2 = ", D
:ClrDraw
:DispGraph
:Line ( A, B, C, D )
:Pause
:Line ( A, B, C, D, 0 )

```

```

ENTER ( X1, Y1 )
AND ( X2, Y2 )
X1 = - 6
Y1 = - 8
X2 = 3
Y2 = 9

```



## Horizontal and Vertical Line Instructions ( DRAW MENU )

These instructions are used to draw horizontal or vertical lines across the entire graphics window.

The format of the instructions are :

### **: Horizontal A**

This instruction draws a horizontal line at  $Y = A$ .

### **: Vertical B**

This instruction draws a vertical line at  $X = B$ .

### **EXAMPLE:**

1. Allow the user to enter a **Y** position for a horizontal line and an **X** position for a vertical line, and then draw these lines.

```

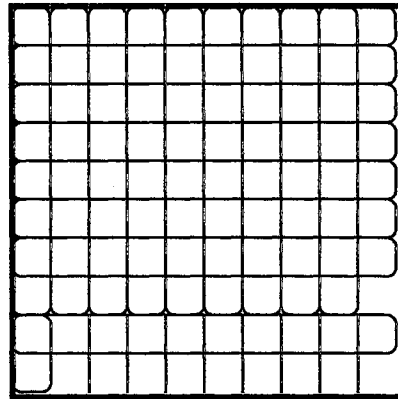
:ClrHome
:Disp " ENTER Y VALUE"
:Input " Y = ",H
:Disp " ENTER X VALUE"
:Input " X = ",V
:ClrDraw
:DispGraph
:Horizontal H
:Vertical V
  
```

<b>ENTER Y VALUE</b>
<b>Y = - 5</b>
<b>ENTER X VALUE</b>
<b>X = 4</b>


**Horizontal and Vertical - Continued:**

2. Draw a grid on the graphics screen by placing the **Horizontal** and **Vertical** line instructions inside a “**For Loop**”.

```
:ClrDraw  
:ZStandard  
:DispGraph  
:For ( P, -10, 10, 2 )  
:Horizontal P  
:Vertical P  
:End
```



## Circle Instruction ( DRAW MENU )

The **Circle** instruction allows the user to specify the ( **X**, **Y** ) coordinates of the center of a circle and the radius, **R** , and then draws the circle on the graphics screen.

The format of the **Circle** instruction is :

**:Circle ( X , Y , R )**

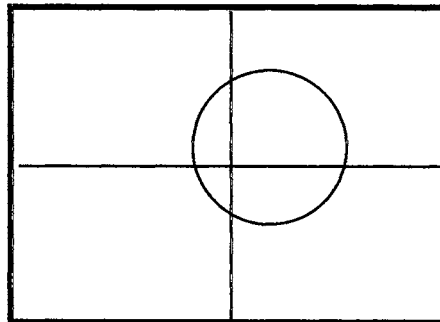
where:    **X** = **X** coordinate of the center.  
               **Y** = **Y** coordinate of the center.  
               **R** = the radius of the circle.

### **EXAMPLE:**

1. Allow the user to enter the ( **X**, **Y** ) coordinates of the center of a circle and the radius **R** , and then draw the circle on the graphics screen.

```
:ClrHome
:Disp " CENTER = ( A , B )
"
:Prompt A
:Prompt B
:Disp " RADIUS = R "
:Prompt R
:ClrDraw
:ZSquare
:DispGraph
:Circle ( A , B , R )
```

```
CENTER = ( A , B )
A = ? 2
B = ? 3
RADIUS = R
R = ? 5
```



**Note:** Use **ZSquare** to prevent the circle from appearing distorted!

## Tangent Instruction ( DRAW MENU )

The **Tangent** instruction draws the tangent line to a given function at a specified **X** coordinate.

The format of the **Tangent** instruction is :

### **:Tangent ( Exp , P )**

where: **Exp** = An expression or **Y** function variable in terms of **X**.

**P** = The **X** coordinate of the point on the function where the tangent line is to be drawn.

### **EXAMPLE:**

1. Allow the user to enter a function **Y1** and an **X** value. Then graph the function and the tangent line at the specified value of **X**.

```

:ClrHome
:Disp "ENTER FUNCTION "
:Disp "( IN QUOTES )"
:Disp ""
:Input " Y1 = ", Y1
:ClrHome
:Disp "NOW ENTER THE "
:Disp "X COORDINATE OF "
:Disp "THE TANGENT "
:Input " X = ", A
:ClrDraw
:DispGraph
:Tangent ( Y1 , A )

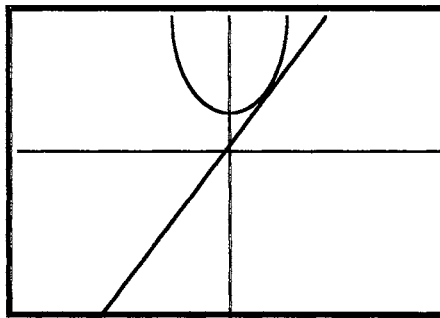
```

**ENTER FUNCTION  
( IN QUOTES )**

**Y1 = " X^2+4 "**

**NOW ENTER THE  
X COORDINATE OF  
THE TANGENT**

**X = 1**



**Text Instruction ( DRAW MENU )**

The **Text** instruction allows the user to display text information on the graphics screen. The **Text** instruction may also include expressions and built-in calculator functions.

The format of the **Text** instruction is :

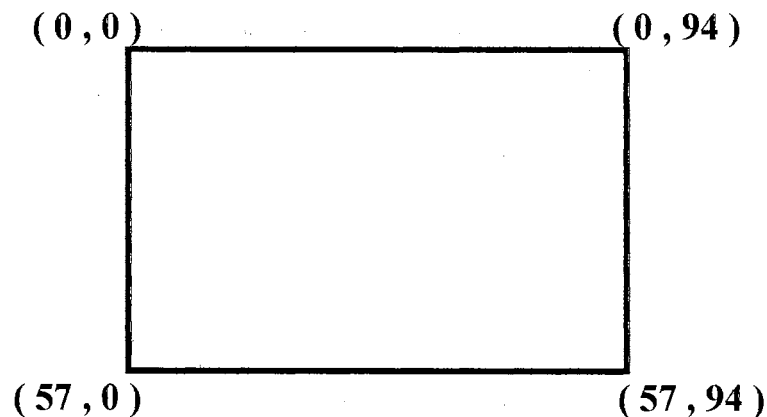
**:Text ( R, C, " TEXT " )**

or

**:Text ( R, C, " TEXT 1 ", EXP 1, " TEXT 2 ", EXP 2, . . . )**

where: **R** = the row number  
**C** = the column number  
**TEXT** = the text message  
**EXP** = a valid expression or built-in calculator function

The ranges of values for **R** and **C** are :  $0 \leq R \leq 57$   
 $0 \leq C \leq 94$



**Note:** You must choose **R** and **C** values carefully to insure that the message does not extend outside the viewing window !

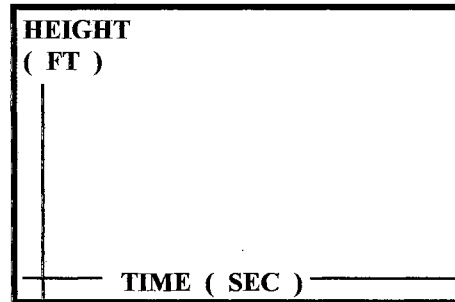
Text - Continued:EXAMPLES:

1. The **Text** instruction is frequently used to label an axis while graphing.

```

:ClrHome
:ClrDraw
:0 → Xmin
:10 → Xmax
:0 → Ymin
:10 → Ymax
:Text ( 0 , 0 , " HEIGHT " )
:Text ( 8 , 0 , " ( FT ) " )
:Text ( 57 , 40 , " TIME ( SEC ) " )
:DispGraph

```



2. The **Text** instruction may also be used to display combinations of text messages and the results of evaluating expressions and built-in functions. For example, allow the user to enter two integers **A** and **B**, and then use the **Text** instruction to print the sum and the quotient of the two integers, and round off the result to two decimal places.

```

:ClrHome
:Input " ENTER A " , A
:Input " ENTER B " , B
:ClrDraw
:AxesOff
:Text ( 0 , 0 , " THE ANSWERS ARE : " )
:Text ( 10 , 0 , " THE SUM OF " , A , " AND " , B , " IS " , A + B )
:Text ( 20 , 0 , A , " DIVIDED BY " , B , " IS " , round ( A / B , 2 ) )

```

The figure shows a rectangular window with a black border. Inside, the text is displayed as follows: "THE ANSWERS ARE:" on the first line, "THE SUM OF 20 AND 3 IS 23" on the second line, and "20 DIVIDED BY 3 IS 6.67" on the third line. The text is left-aligned and uses a monospaced font.

**Note:** The **AxesOff** instruction is used to turn off the graphics axis so the screen looks like the home screen.



## DrawInv Instruction ( DRAW MENU )

The **DrawInv** instruction draws the inverse of an expression or **Y** - variable. The original function must be expressed in terms of **X**.

The format of the **DrawInv** instruction is :

**:DrawInv EXP**

where: **EXP** = An expression or **Y** - variable  
expressed in terms of **X**.

### EXAMPLE:

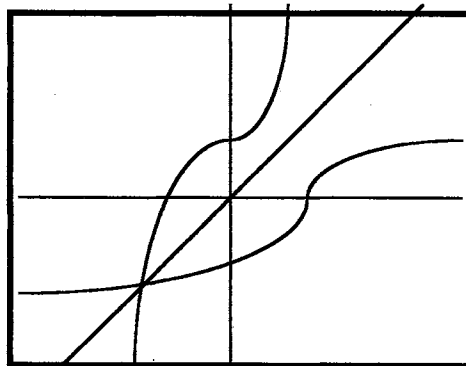
1. Allow the user to enter a function and store that function in **Y1**. Then graph:
  - (a) The original function.
  - (b) The  $45^\circ$  line  $Y = X$ .
  - (c) The inverse of the original function using **DrawInv**.

```

:ClrHome
:Disp "ENTER FUNCTION"
:Disp "( IN QUOTES )"
:Disp ""
:Input " Y1 = ", Y1
:ClrHome
:" X " → Y2
:ZSquare
:DrawInv Y1
  
```

**ENTER FUNCTION  
( IN QUOTES )**

**Y1 = " X ^ 3 + 3 "**



## Pt - On Instruction ( DRAW MENU )

The **Pt - On** instruction plots an individual point on the graphics screen at a given ( X , Y ) coordinate.

The format of the **Pt - On** instruction is :

**:Pt - On ( X , Y )**

where: **X** = The **X** coordinate of the point.

**Y** = The **Y** coordinate of the point.

### **EXAMPLE:**

- One use of the **Pt - On** instruction is to allow the user to select the spacing between individual points when plotting a graph. First, allow the user to enter a function and store it in **Y1**. Then, use the **FnOff** instruction to turn off **Y1** so the calculator will not automatically graph it. ( **FnOff** is found in the [ Y - VARS ] menu )

Then allow the user to enter **Xmin** , **Xmax** , and the  $\Delta X$  spacing between each point. Put the program in a loop from **Xmin** to **Xmax** using the given stepsize and evaluate the function at each **X** value to obtain the corresponding **Y** value. Then use the **Pt - On** instruction to plot each point.

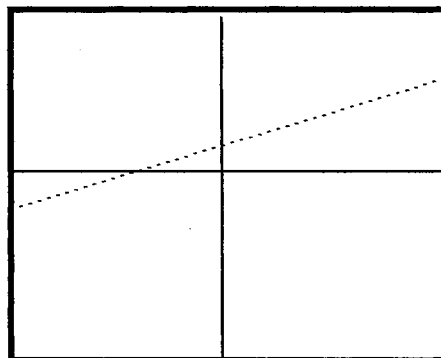
```

:ClrHome
:Disp " ENTER FUNCTION "
:Disp " ( IN QUOTES )"
:Input " Y1 = ", Y1
:Disp ""
:Prompt Xmin
:Prompt Xmax
:Disp " SPACING BETWEEN "
:Input " POINTS = ", S
:FnOff
:DispGraph
:For ( A , Xmin , Xmax , S )
:Y1 ( A ) → B
:Pt - On ( A , B )
:End

```

**ENTER FUNCTION**  
**( IN QUOTES )**  
**Y1 = ".5 X + 2"**

**Xmin = - 10**  
**Xmax = 10**  
**SPACING BETWEEN**  
**POINTS = 1**



## nDeriv Instruction

The **nDeriv** instruction returns the first derivative of an expression or **Y**- variable with respect to a given variable at a specific point.

The format of the **nDeriv** instruction is :

**:nDeriv ( EXP , VAR , PT )**

where: **EXP** = An expression or **Y** - variable.

**VAR** = With respect to **VAR**.

**PT** = At the point **PT**.

### EXAMPLES:

1. Allow the user to enter a function and store it in **Y1**. Then allow the user to enter a value for **X**. Have the program print the slope of the tangent line to the function **Y1** at the specified **X** value.

```

:ClrHome
:Disp "ENTER FUNCTION "
:Disp " ( IN QUOTES )"
:Disp ""
:Input " Y1 = ", Y1
:Disp ""
:Disp "ENTER X VALUE "
:Input " X = ", A
:nDeriv ( Y1 , X , A ) → M
:ClrHome
:Disp " THE SLOPE OF THE "
:Disp " TANGENT LINE AT "
:Output ( 3 , 1 , " X = " )
:Output ( 3 , 5 , A )
:Output ( 5 , 1 , " IS M = " )
:Output ( 5 , 4 , M )

```

```

ENTER FUNCTION
( IN QUOTES )

```

```

Y1 = " X ^ 2 "

```

```

ENTER X VALUE
X = 2

```

```

THE SLOPE OF THE
TANGENT LINE AT
X = 2

```

```

IS M = 4

```

**nDeriv - Continued:**

2. This example will demonstrate how the **nDeriv** instruction may be used to graph both the first and second derivatives of a function entered by the user. The original function will be stored in **Y1**, the first derivative will be stored in **Y2**, and the second derivative will be stored in **Y3**.

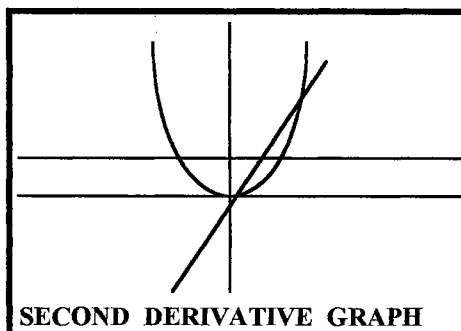
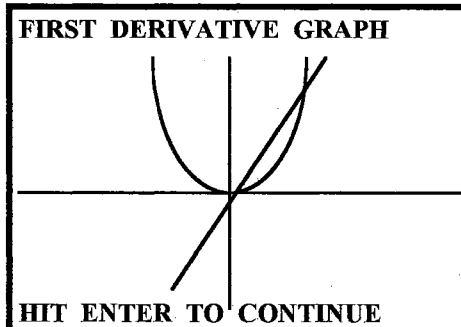
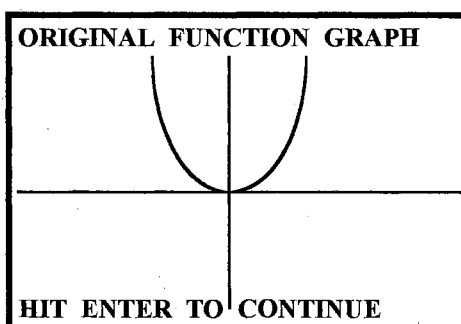
```

:ClrHome
:Disp "ENTER FUNCTION"
:Disp "( IN QUOTES )"
:Disp ""
:Input " Y1 = ", Y1
:" nDeriv ( Y1 , X , X )" → Y2
:" nDeriv ( nDeriv ( Y1 , X , X ) ,
  X , X )" → Y3
:FnOff
:ClrHome
:ClrDraw
:Text ( 0 , 0 , " ORIGINAL F
  UNCTION GRAPH ")
:FnOn 1
:Text ( 57 , 0 , " HIT ENTER
  TO CONTINUE ")
:Pause
:ClrDraw
:Text ( 0 , 0 , " FIRST DERIV
  ATIVE GRAPH ")
:FnOn 2
:Text ( 57 , 0 , " HIT ENTER
  TO CONTINUE ")
:Pause
:ClrDraw
:Text ( 57 , 0 , " SECOND DER
  IVATIVE GRAPH ")
:FnOn 3

```

**ENTER FUNCTION  
( IN QUOTES )**

**Y1 = " X ^ 2 "**



**Note:** The **FnOn** instruction turns on each **Y** function as it is needed.

# **SAMPLE PROGRAMS**

## The TI - 82 and TI - 83 Setup Programs

Sometimes you may run a program and the program will produce unexpected results or perhaps it will not run at all. One reason this may happen is that the normal default setting in the calculator have been changed by some previous operation. These setup programs reset the values of the internal calculator variables to their default settings, and this will help insure that your programs will produce the expected output. It is suggested that you enter the setup program into your calculator and run it before running your own programs, or whenever you want to return the calculator to its default settings.

### **PROGRAM: SETUP82**

**:ClrHome**  
**:Normal**  
**:Float**  
**:Radian**  
**:Func**  
**:Connected**  
**:Sequential**  
**:FullScreen**  
**:RectGC**  
**:CoordOn**  
**:GridOff**  
**:AxesOn**  
**:LabelOff**  
**:FnOff**  
**:PlotsOff**  
**:ClrDraw**  
**:ClrTable**  
**:Zstandard**  
**:ClrHome**

### **PROGRAM: SETUP83**

**:ClrHome**  
**:Normal**  
**:Float**  
**:Radian**  
**:Func**  
**:Connected**  
**:Sequential**  
**:Real**  
**:Full**  
**:RectGC**  
**:CoordOn**  
**:GridOff**  
**:AxesOn**  
**:LabelOff**  
**:ExprOn**  
**:FnOff**  
**:PlotsOff**  
**:ClrDraw**  
**:ClrTable**  
**:Zstandard**  
**:ClrHome**

**Note:** Refer to the “menu map” in the guidebook to locate each of these instructions.

**Note:** **Sample Program 8** will demonstrate how the setup program may be linked to your programs so it will be automatically executed each time you run one of your programs.

**Note:** These setup programs are not “fixed”. You may add or remove any of the instructions you choose to suit your programming needs !

## Sample Program 1

This program will demonstrate how a formula may be stored in a program and then be evaluated using input from the user.

A common problem in an Algebra class is to use the formula for simple interest to determine the future value of money.

The formula for simple interest is :  $A = P ( 1 + R ) ^ T$

where : **A** = Final Amount  
**P** = Initial Principal  
**R** = Yearly Interest Rate ( in decimal form ! )  
**T** = Time ( in years )

### Programming Plan of Attack :

- (1) First, display the formula so the user can see it.
- (2) Allow the user to enter **P**, **R**, and **T**.
- (3) Change **R** into decimal form by dividing by 100.
- (4) Evaluate the formula and store the result in **A**.
- (5) Clear the screen and display the result.

Assume : **P** = \$1000  
**R** = 8.3 %  
**T** = 15 years

### **PROGRAM: MONEY**

```
:ClrHome
:Disp "A=P (1+R) ^ T "
:Disp " "
:Prompt P
:Prompt R
:Prompt T
:R / 100 → R
:P ( 1+R ) ^ T → A
:ClrHome
:Disp "FINAL AMOUNT = ", A
```

**A=P (1+R) ^ T**

**P =? 1000**

**R =? 8.3**

**T =? 15**

**FINAL AMOUNT =**

**3306.944067**

## Sample Program 2

If you are given the coordinates of two points (  $X_1, Y_1$  ) and (  $X_2, Y_2$  ) you can find the midpoint between the two points and the distance between the two points by using the following formulas:

$$\text{Midpoint} = \left( \frac{X_1 + X_2}{2}, \frac{Y_1 + Y_2}{2} \right)$$

$$\text{Distance} = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

Write a program which will allow the user to enter values for  $X_1, Y_1, X_2, Y_2$  and then calculate and display the coordinates of the midpoint and the distance between the two points.

### PROGRAM: POINTS

```

:ClrHome
:Disp "ENTER THE"
:Disp "COORDINATES"
:Disp " "
:Input "X1= ", A
:Input "Y1= ", B
:Input "X2= ", C
:Input "Y2= ", D
:( A + C ) / 2 → M
:( B + D ) / 2 → N
:√((C - A)^2 + (D - B)^2) → E
:ClrHome
:Disp "MIDPOINT IS"
:Disp "X ="
:Disp M ► Frac
:Disp "Y ="
:Disp N ► Frac
:Pause
:ClrHome
:Disp "DISTANCE ="
:Disp round(E, 3)
:Disp "OR IN √ FORM ="
:Output ( 4, 1, "√" )
:Output ( 4, 3, E ^ 2)

```

**ENTER THE  
COORDINATES**

X1= 2  
Y1= 9  
X2= -8  
Y2= -6

**MIDPOINT IS**

X = -3  
Y = 3/2

**DISTANCE =**

18.028  
**OR IN √ FORM =**  
√ 325



### Sample Program 3

This problem will be solved in three different ways to give you a chance to compare loops created by **Goto**, **While**, and **Repeat** instructions.

Write a guessing game program which will generate a random number between 1 and 100. Then allow the user to guess the number. Tell the user if the guess is too high or too low, and then let them guess again until they get it right. Also, tell the user how many guesses it took to find the number. Then allow a friend to try the game and see who takes the least number of guesses.

In the programs that follow : **R** = The random number (  $1 \leq R \leq 100$  )  
**G** = The user's guess  
**T** = The number of guesses

The first program, **GUESS1**, will solve the problem using a **Goto** loop:

```
:PROGRAM: GUESS1
:ClrHome
: 0 → T
: int ( rand * 100 ) + 1 → R
:Lbl A
:Input "ENTER GUESS ", G
: T + 1 → T
:If G > R
:Disp "TOO HIGH "
:If G < R
:Disp "TOO LOW "
:If G ≠ R
:Goto A
:Disp " "
:Disp "CORRECT:"
:Disp "GUESSES =", T
```

```
ENTER GUESS 50
TOO HIGH
ENTER GUESS 30
TOO LOW
ENTER GUESS 37
TOO LOW
ENTER GUESS 44
TOO HIGH
ENTER GUESS 41

CORRECT:
GUESSES =
```

5

**Note:** You might want to review the discussion on the random number instruction, **rand**, in the instruction section !

### Sample Program 4

This program will solve the same problem presented in **Sample Problem 3** using a **While** loop:

```

:PROGRAM: GUESS2
:ClrHome
:int ( rand * 100 ) + 1 → R
:Input "ENTER GUESS ", G
: 1 → T
:While G ≠ R
:If G > R
:Then
:Disp "TOO HIGH "
:Else
:Disp "TOO LOW "
:End
:Input "ENTER GUESS ", G
: T + 1 → T
:End
:Disp " "
:Disp "CORRECT:"
:Disp "GUESSES =", T

```

```

ENTER GUESS 50
TOO LOW
ENTER GUESS 85
TOO LOW
ENTER GUESS 95
TOO HIGH
ENTER GUESS 91
TOO HIGH
ENTER GUESS 87
TOO LOW
ENTER GUESS 89
TOO LOW
ENTER GUESS 90

CORRECT:
GUESSES =

```

7

**Sample Program 5**

This program will solve the same problem presented in **Sample Program 3** using a **Repeat** loop:

```
:PROGRAM: GUESS3  
:ClrHome  
:int ( rand * 100 ) + 1 → R  
: 0 → T  
:Repeat G = R  
:Input "ENTER GUESS ", G  
:If G > R  
:Disp "TOO HIGH"  
:If G < R  
:Disp "TOO LOW"  
: T + 1 → T  
:End  
:Disp " "  
:Disp "CORRECT:"  
:Disp "GUESSES =", T
```

```
ENTER GUESS 50  
TOO HIGH  
ENTER GUESS 10  
TOO LOW  
ENTER GUESS 36  
TOO LOW  
ENTER GUESS 42
```

```
CORRECT:  
GUESSES =
```

4

## Sample Program 6

If you flip a coin repeatedly it should come up heads 50 % of the time and tails 50 % of the time. However, if you only flip the coin twice, there is a distinct possibility that you might get 100 % heads or 100 % tails.

Write a program which demonstrates that the greater the number of flips, the more likely the final percentages will approach 50 % each.

### Programming Plan of Attack:

- (1) First, allow the user to enter the number of flips ( **F** ).
- (2) Set up a **For** loop from 1 to **F**.
- (3) Let 1 = heads and 2 = tails and use a random number generator from 1 and 2.
- (4) Count the number of heads ( **H** ) and tails ( **T** ).
- (5) When the loop finishes, display the number of heads and tails.  
Include the % heads ( **C** ) and the % tails ( **D** ).

```

:PROGRAM: FLIP
:ClrHome
:Disp "HOW MANY FLIPS?"
:Input F
:ClrHome
:Output ( 1 , 1 , " FLIP NO. " )
:Output ( 3 , 1 , " RESULT = " )
:0 → H
:0 → T
:For ( N , 1 , F )
:int ( rand * 2 ) + 1 → R
:If R = 1
:Then
: H + 1 → H
:Else
: T + 1 → T
:End
:Output ( 1 , 10 , N )
:Output ( 3 , 9 , R )
:End
:ClrHome
: ( H / F ) * 100 → C
: ( T / F ) * 100 → D
:round ( C , 0 ) → C
:round ( D , 0 ) → D
:Output ( 1 , 1 , "TOTAL = " )
:Output ( 1 , 8 , F )

```

```

:Output ( 2 , 1 , "HEADS = " )
:Output ( 2 , 8 , H )
:Output ( 3 , 1 , "TAILS = " )
:Output ( 3 , 8 , T )
:Output ( 5 , 1
,"PERCENTAGES:")
:Output ( 6 , 1 , "HEADS  " )
:Output ( 6 , 7 , C )
:Output ( 7 , 1 , "TAILS  " )
:Output ( 7 , 7 , D )

```

HOW MANY FLIPS ?  
50

FLIP NO. 34 etc  
  
RESULT 1 etc

TOTAL = 50  
HEADS = 23  
TAILS = 27  
PERCENTAGES:  
HEADS 46  
TAILS 54

## Sample Program 7

This program will demonstrate how to enter a function within a program and how to switch from text to graphics mode.

Allow the user to enter a function and then graph that function on a **10 by 10** graph scale. When the program finishes execution, leave the graph on the screen so the user can perform other operations on it.

### Programming Plan of Attack:

- (1) Allow the user to enter the function and assign it to the graph variable  $Y_1$ .  
**Note:** Remember, the function must be enclosed within quotes as you enter it!  
 (  $Y_1$  is found in the [2<sup>nd</sup>], [Y-VARS], 1: Function,  $Y_1$  menu )
- (2) Display a message to the user telling them what they will see.
- (3) Display the graph and end the program.

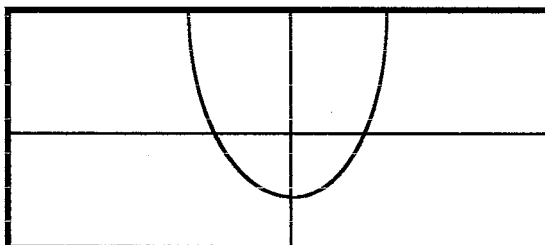
```
PROGRAM: GRAPHFN
:ClrHome
:Disp "ENTER FUNCTION"
:Disp "( IN QUOTES )"
:Disp ""
:Input " Y1 = ", Y1
:ClrHome
:Disp " HIT ENTER TO SEE "
:Disp " GRAPH ON A "
:Disp " 10 BY 10 SCALE "
:Disp ""
:Disp " THEN YOU CAN "
:Disp " TRACE, ZOOM, ETC "
:Pause
:ClrHome
:ZStandard
```

ENTER FUNCTION  
( IN QUOTES )

$Y_1 = "X^2 - 6"$

HIT ENTER TO SEE  
GRAPH ON A  
10 BY 10 SCALE

THEN YOU CAN  
TRACE, ZOOM, ETC



**Note:** You will get an error message if you forget to put quotes around the function !

**tip:** Instead of using **DispGraph**, use **ZStandard**. **ZStandard** will both display the graph and zoom to a 10 by 10 scale !

## Sample Program 8

This program will demonstrate how one program can call another program.

This example will use the same problem presented in Sample Program 7, however, one additional instruction will be added at the beginning of Sample Program 7. This new instruction will call the TI - 82 setup program before executing Sample Program 7. By calling the setup program, you will automatically reset all the system variables to their default values before your main program runs, thus assuring that your program will produce the output you expected it to. It is possible that the system variables may have been changed to a setting that is not compatible with the program you are attempting to run, and the setup program will reset these variables to prevent possible problems.

**Note:** The `prgm` instruction is found in the [PRGM], [CTL] menu.

```

PROGRAM: GRAPHFN
:prgmSETUP82
:Disp " ENTER FUNCTION "
:Disp " ( IN QUOTES )"
:Disp " "
:Input " Y1 = ", Y1
:ClrHome
:Disp " HIT ENTER TO SEE "
:Disp " GRAPH ON A "
:Disp " 10 BY 10 SCALE "
:Disp " "
:Disp " THEN YOU CAN "
:Disp " TRACE, ZOOM, ETC "
:Pause
:ClrHome
:ZStandard

```

```

PROGRAM: SETUP82
:ClrHome
:Normal
:Float
:Radian
:Func
:Connected
:Sequential
:FullScreen
:RectGC
:CoordOn
:GridOff
:AxesOn
:LabelOff
:FnOff
:PlotsOff
:ClrDraw
:ClrTable
:Zstandard
:ClrHome

```

**Note:** Notice that even though the **Return** statement at the end of program **SETUP82** has been omitted, an implied return exists and execution still returns to the calling program.

## Sample Program 9

This program will demonstrate how the **Line** instruction from the [ DRAW ] menu may be used to draw a triangle on the screen.

Allow the user to enter the coordinates of the three vertices of a triangle (  $X_1, Y_1$  ), (  $X_2, Y_2$  ), (  $X_3, Y_3$  ). Then allow the user to set the size of the graph window by selecting the four window values  $X_{MIN}$ ,  $X_{MAX}$ ,  $Y_{MIN}$ ,  $Y_{MAX}$ .

The format of the **Line** instruction is : **Line (  $X_1, Y_1, X_2, Y_2$  )**

**Note:** This instruction will draw a line between points (  $X_1, Y_1$  ) and (  $X_2, Y_2$  ).

The **Line** instruction is found in the [ 2<sup>nd</sup> ], [ DRAW ] menu.

The window variables are found in the [ VARS ], [ WINDOW ] menu.

### Programming Plan of Attack:

- (1) Allow the user to enter the coordinates of the three vertices: (  $A, B$  ), (  $C, D$  ), (  $E, F$  )
- (2) Then allow the user to enter the four graph window values.
- (3) Then use three separate **Line** instructions to draw the three lines to form a triangle.

```

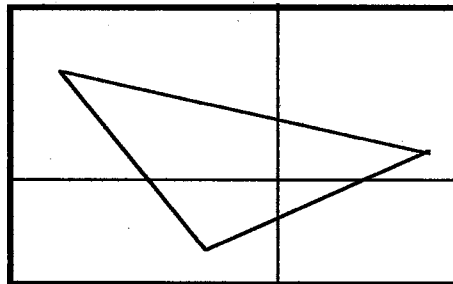
PROGRAM: DRAWTRI
:ClrHome
:Disp "ENTER THE "
:Disp "COORDINATES OF "
:Disp "THE VERTICES "
:Disp " ( X1,Y1 ) , ( X2,Y2 ) "
:Disp " AND ( X3,Y3 ) "
:Input " X1 = ", A
:Input " Y1 = ", B
:Input " X2 = ", C
:Input " Y2 = ", D
:Input " X3 = ", E
:Input " Y3 = ", F
:ClrHome
:Disp "ENTER WINDOW "
:Disp "VALUES "
:Input " XMIN = ", XMIN
:Input " XMAX = ", XMAX
:Input " YMIN = ", YMIN
:Input " YMAX = ", YMAX
:DispGraph
:Line ( A, B, C, D )
:Line ( C, D, E, F )
:Line ( E, F, A, B )

```

The output for Sample Program 9 :

```
ENTER THE
COORDINATES OF
THE VERTICES
( X1,Y1 ) , ( X2,Y2 )
AND ( X3,Y3 )
X1 = 10
Y1 = 5
X2 = - 3
Y2 = - 8
X3 = - 12
Y3 = 13
```

```
ENTER WINDOW
VALUES :
XMIN = - 15
XMAX = 10
YMIN = - 10
YMAX = 20
```



**Note:** The **ClrDraw** instruction clears previous drawings from the graph.  
The **ClrDraw** instruction is found in the [ 2<sup>nd</sup> ], [ DRAW ] menu.



### Sample Program 10

This program will demonstrate how the calculator may be used to graph conic sections. It will also show the importance of coordinate geometry in writing mathematical programs.

The equation of an ellipse in standard form is :

$$\frac{(X-H)^2}{A^2} + \frac{(Y-K)^2}{B^2} = 1$$

Where:  $(H, K)$  = the coordinates of the center of the ellipse.  
 $A$  and  $B$  = the lengths of the semi-major and semi-minor axes.

Write a program which allows the user to enter the coordinates of the center of the ellipse  $(H, K)$ , and values for  $A$  and  $B$ . Then graph the ellipse initially on a 10 by 10 graph scale. The graph should include both the major and minor axes. Then give the user the option to zoom in or zoom out using a scale factor of 2.

#### Programming Plan of Attack:

- (1) Because an ellipse is not a function (remember the vertical line test from Algebra!) you cannot graph it directly on a calculator because the calculator can only graph functions. You can, however, split the ellipse into two parts, a top half and a bottom half. Each of these halves is a function and each may be graphed separately to form a complete ellipse. To algebraically split an ellipse into top and bottom halves, solve the ellipse equation for  $Y$ . The resulting equation is :

$$Y = K \pm \sqrt{(B^2 - (B^2 / A^2)(X - H)^2)}$$

The top half of the ellipse is given by :

$$Y_1 = K + \sqrt{(B^2 - (B^2 / A^2)(X - H)^2)}$$

The bottom half of the ellipse is given by :

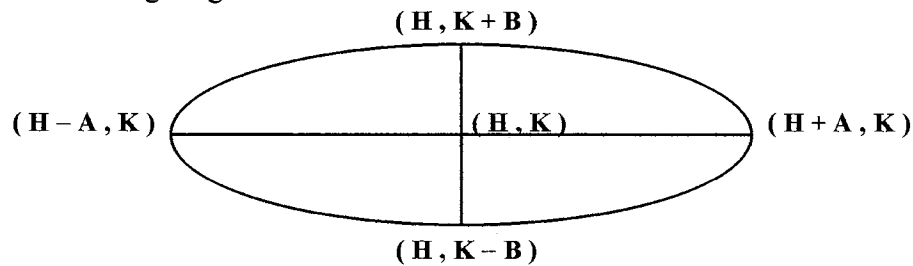
$$Y_2 = K - \sqrt{(B^2 - (B^2 / A^2)(X - H)^2)}$$

Now assign the top function to graph variable  $Y_1$  and the bottom function to  $Y_2$ .

- (2) Allow the user to enter values for  $H, K, A, B$

**Sample Program 10 - Continued:**

- (3) Use the **Line** instruction to draw the major and minor axes by referring to the following diagram:



Use: **Line ( H - A , K , H + A , K )**  
**Line ( H , K - B , H , K + B )**

- (4) Initially graph the ellipse with a 10 by 10 graph scale (  $S = 10$  ), then allow the user to zoom in or zoom out by a factor of 2.

```

PROGRAM: ELLIPSE
:prgmSETUP82
:Disp "(X - H)2 / A2 + "
:Disp " "
:Disp "(Y - K)2 / B2 = 1 "
:Prompt H
:Prompt K
:Prompt A
:Prompt B
:" K + √(B2 - (B2 / A2)(X - H
) )2 )" → Y1
:" K + √(B2 - (B2 / A2)(X - H
) )2 )" → Y2
: 10 → S
:Lbl 1
: - S → XMIN
: S → XMAX
: - S → YMIN
: S → YMAX
: DispGraph
: Line ( H - A , K , H + A , K )
: Line ( H , K - B , H , K + B )
: Pause
: ClrHome
: Disp " (1) ZOOM IN "
: Disp " (2) ZOOM OUT "
: Disp " (3) QUIT "
: Disp " "
: Input " ENTER 1, 2, OR 3: ", C
: If C = 3
: Goto 2
: If C = 1
: Then
: S / 2 → S
: Else
: S * 2 → S
: End
: Goto 1
: Lbl 2
: ClrHome

```

The output for Sample Program 10 :

$$(X - H)^2 / A^2 +$$

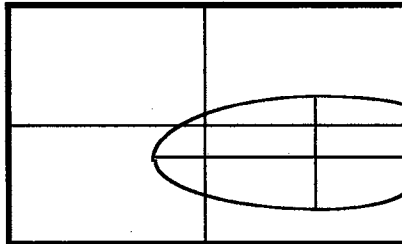
$$(Y - H)^2 / B^2 = 1$$

$$H = ? 6$$

$$K = ? -4$$

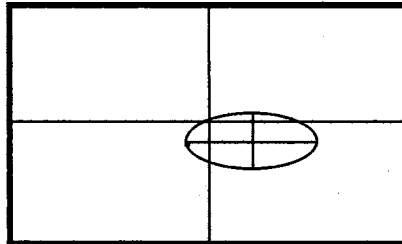
$$A = ? 9$$

$$B = ? 6$$



- (1) ZOOM IN
- (2) ZOOM OUT
- (3) QUIT

ENTER 1, 2, OR 3 : 2



- (1) ZOOM IN
- (2) ZOOM OUT
- (3) QUIT

ENTER 1, 2, OR 3 : 3

## Sample Program 11

This program will demonstrate a trigonometric application.

Write a program which will allow the user to enter an angle in either degree or radian mode, and the display all six trig functions for the given angle. Have the program anticipate the fact that some trig functions are undefined for certain angles, such as the  $\tan(90^\circ)$ . If the trig function is undefined, print "UNDEFINED" for the user.

### Programming Plan of Attack:

- (1) First, ask the user if the angle is in the **Degree** or **Radian** mode.
- (2) Allow the user to enter the angle in either mode.
- (3) Then print all six trig functions in table form. If the function is undefined, then print "UNDEFINED". Use four decimal accuracy in the output.

```

PROGRAM: TRIGFNS
:Fix 4
:ClrHome
:Disp "WHICH MODE "
:Disp " "
:Disp "(1) DEGREE "
:Disp "(2) RADIAN "
:Disp " "
:Input "ENTER 1 OR 2 : ", C
:If C = 1
:Degree
:If C = 2
:Radian
:ClrHome
:Disp "ENTER THE ANGLE "
:Prompt A
:ClrHome
:Output ( 1, 1, "SIN A=" )
:Output ( 1, 8, sin A )
:Output ( 2, 1, "COS A=" )
:Output ( 2, 8, cos A )
:Output ( 3, 1, "TAN A=" )
:If ( cos A ) = 0
:Then
:Output ( 3, 8, "UNDEFINED " )
:Else
:Output ( 3, 8, tan A )
:End
:Output ( 4, 1, "CSC A=" )
:If ( sin A ) = 0
:Then
:Output ( 4, 8, "UNDEFINED " )
:Else
:Output ( 4, 8, 1 / sin A )
:End
:Output ( 5, 1, "SEC A=" )
:If ( cos A ) = 0
:Then
:Output ( 5, 8, "UNDEFINED " )
:Else
:Output ( 5, 8, 1 / cos A )
:End
:Output ( 6, 1, "COT A=" )
:If ( sin A ) = 0
:Then
:Output ( 6, 8, "UNDEFINED " )
:Else
:Output ( 6, 8, cos A / sin A )
:End
:Float

```

**Note:** The final **Float** instruction resets the default decimal setting.

The output for Sample Program 11 :

**WHICH MODE**

- (1) DEGREE**
- (2) RADIAN**

**ENTER 1 OR 2 : 1**

**ENTER THE ANGLE**

**A = ? 60**

**SIN A = .8660**  
**COS A = .5000**  
**TAN A = 1.7321**  
**CSC A = 1.1547**  
**SEC A = 2.0000**  
**COT A = .5774**

**WHICH MODE**

- (1) DEGREE**
- (2) RADIAN**

**ENTER 1 OR 2 : 2**

**ENTER THE ANGLE**

**A = ?  $\pi / 2$**

**SIN A = 1.0000**  
**COS A = 0.0000**  
**TAN A = UNDEFINED**  
**CSC A = 1.0000**  
**SEC A = UNDEFINED**  
**COT A = 0.0000**

## Sample Program 12

This program will demonstrate a simple method of storing and displaying text information using the **Disp** instruction.

Assume you wanted to store the names and phone numbers of your friends in your calculator. Use the **Disp** and **Pause** instructions to display this information.

```

:PROGRAM: PHONE
:ClrHome
:Disp " JOHN SMITH "
:Disp " 555 - 3958 "
:Disp " "
:Disp " MARY JONES "
:Disp " 555 - 9481 "
:Disp " "
:Disp " MORE - HIT ENTER "
:Pause
:ClrHome
:Disp " ED WILLIAMS "
:Disp " (968) 555 - 2974 "
:Disp " "
:Disp " SUE BARNES "
:Disp " 555 - 3401 "
:Disp " "
:Disp " MORE - HIT ENTER "
:Pause
:ClrHome
:Disp " ERIC THOMAS "
:Disp " 555 - 5254 "

```

**JOHN SMITH**  
**555 - 3958**

**MARY JONES**  
**555 - 9481**

**MORE - HIT ENTER**

**ED WILLIAMS**  
**(968) 555 - 2974**

**SUE BARNES**  
**555 - 3401**

**MORE - HIT ENTER**

**ERIC THOMAS**  
**555 - 5254**

### Sample Program 13

This program will demonstrate how the **Text**, **fMax**, **FnOff**, and **Pt-On** instructions may be used in a program to model projectile motion.

The formula that models projectile motion is  $Y = H + V \cdot T - 16 T^2$

where: **Y** = height ( in feet )  
**H** = initial height (  $\geq 0$  )  
**V** = initial velocity  
**T** = time ( in seconds )

Display the time on the horizontal axis and the height on the vertical axis.

#### Programming Plan of Attack :

- (1) First, display the formula.
- (2) Allow the user to enter **H** and **V**.
- (3) Allow the user to enter the number of seconds (**S**) for the model to run by setting **Xmin** = 0 and **Xmax** = **S**.
- (4) Use **fMax** to automatically adjust the **Y** scale of the graph with **Ymin** = 0 and **Ymax** = the maximum value of **Y** during the given time period.
- (5) Evaluate the function in increments of 0.1 seconds and use the **Pt-On** instruction to plot each (**X**, **Y**) point.
- (6) Use the **Text** instruction to label the axes and to display the current height and time as the projectile moves.
- (7) Have the program stop when the object hits the ground. (  $Y \leq 0$  )
- (8) Finally, ask the user if they would like to change the time period and rerun the program.

```
:PROGRAM: MODEL
:ClrHome
:Disp "Y = H + V*T - 16 T ^ 2 "
:Disp " "
:Disp "H = INITIAL HEIGHT"
:Disp "V = INITIAL VELOCITY"
:Prompt H
:Prompt V
:Lbl 1
:Disp "HOW MANY SEC ?"
:Prompt S
:ClrHome
:Disp "SCALING GRAPH"
:Disp "PLEASE WAIT"
```

$$Y = H + V * T - 16 T^2$$

```
H = INITIAL HEIGHT
V = INITIAL VELOCITY
H = ? 100
V = ? 200
HOW MANY SEC ?
S = ? 12
```

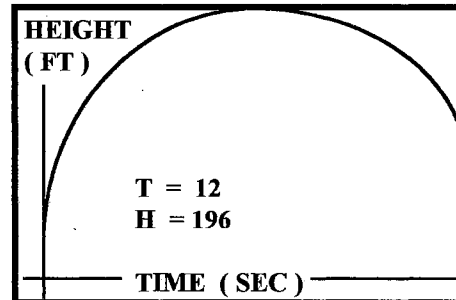
```
SCALING GRAPH
PLEASE WAIT
```

**Sample Program 13 - Continued:**

```

: S → Xmax
: " H + V*T - 16 T 2 " → Y1
:fMax ( Y1 , X , Xmin , Xmax ) → M
:Y1 ( M ) → Ymax
:FnOff 1
:0 → Xmin
:0 → Ymin
:Text ( 0 , 0 , "HEIGHT " )
:Text ( 8 , 0 , " ( FT ) " )
:Text ( 57 , 40 , " TIME ( SEC ) " )
:Text ( 35 , 30 , " T = " )
:Text ( 42 , 30 , " H = " )
:DispGraph
:For ( N , 0 , S , 0.1 )
:Y1 ( N ) → Y
:Pt-On ( N , Y )
:Text ( 35 , 40 , round ( N , 1 ) ,
" " )
:Text ( 42 , 40 , round ( Y1 ( N ) , 1 ) ,
" " )
:If Y < 0
:Goto 2
:End
:Lbl 2
:Pause
:ClrHome
:Disp " RERUN AND "
:Disp " CHANGE NUMBER "
:Disp " OF SECONDS ? "
:Disp "(1) YES "
:Disp "(2) NO "
:Input " ENTER 1 OR 2 " , R
:If R = 1
:Then
:ClrHome
:Goto 1
:End

```



```

RERUN AND
CHANGE NUMBER
OF SECONDS ?
(1) YES
(2) NO
ENTER 1 OR 2 2
DONE

```



<b>Sample Program 14</b>
--------------------------

A common problem in Calculus is to find the Riemann sum of a function by summing up the areas of rectangles between the function and the **X**-axis. This program will allow the user to find the Riemann sum of a given function and graphically display both the function and the rectangles used to obtain the sum. The program will also demonstrate an automatic scaling feature for the **Y**-axis.

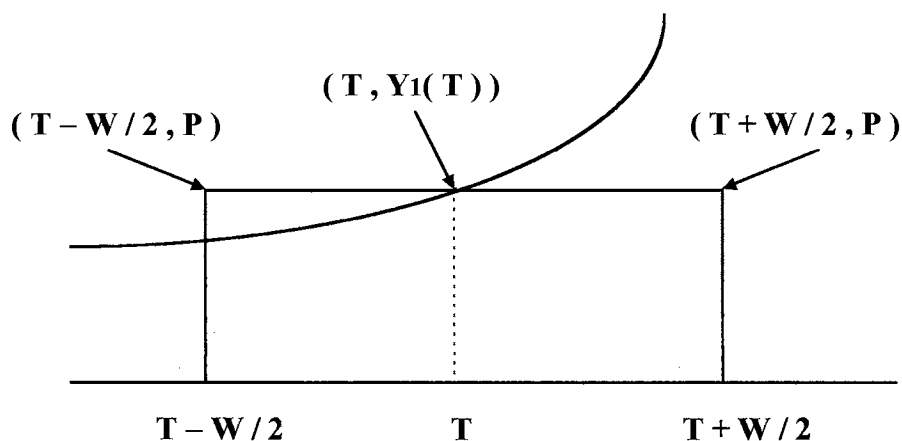
**Programming Plan of Attack:**

- (1) First, allow the user to enter a function ( **Y1** ).
- (2) Then, allow the user to enter the left ( **L** ) and right ( **R** ) endpoints of the interval.
- (3) Finally, allow the user to enter the number of rectangles ( **N** ) desired to calculate the Riemann sum.
- (4) This program will use the “ midpoint rule ” when calculating the signed area of each rectangle.
- (5) To take full advantage of the viewing window, let **Xmin = L** and **Xmax = R**.
- (6) The width ( **W** ) of each rectangle will be the same with  $W = (U - L) / N$ .
- (7) Use **fMin** and **fMax** to automatically scale the **Y**-axis, but add statements to insure that the **X**-axis will always be displayed at either the top or bottom of the window.
 

<b>:If Ymin &gt; 0</b>	<b>:If Ymax &lt; 0</b>
<b>: 0 → Ymin</b>	<b>: 0 → Ymax</b>
- (8) Put the program in a loop from **1** to **N** to calculate the area of each rectangle and draw a picture of each rectangle on the graph.
- (9) The midpoint ( **T** ) of each rectangle will be initially given by  $L + W / 2 \rightarrow T$  and subsequently by  $T + W \rightarrow T$ .
- (10) The area of each rectangle will be given by  $Y1(T) * W$  and be added to the running sum by  $S + Y1(T) \rightarrow S$ .

**Sample Program 14 - Continued:**

(11) To draw each rectangle, use the following coordinates ( with  $Y_1(T) = P$  ).



It will take three **Line** instructions to draw the rectangle:

**:Line** ( $T - W/2, 0, T - W/2, P$ )

**:Line** ( $T - W/2, P, T + W/2, P$ )

**:Line** ( $T + W/2, P, T + W/2, 0$ )

(12) To move to the next rectangle, use  $T + W \rightarrow T$ .

(13) After drawing the required number of rectangles, **Pause** the display so the user may view the graph.

(14) When the user hits **[ENTER]** switch to the home screen and display the Reimann sum.

(15) Finally, allow the user to rerun the program and change the limits or the number of rectangles using **Lbl 1** and **Goto**.

**Sample Program 14 - Continued:****PROGRAM: REIMANN**

```

:ClrHome
:Disp " ENTER FUNCTION "
:Disp "( IN QUOTES )"
:Disp ""
:Input " Y1 = ", Y1
:Lbl 1
:ClrDraw
:ClrHome
:Disp " LOWER LIMIT "
:Disp " OF X "
:Prompt L
:Disp " UPPER LIMIT "
:Disp " OF X "
:Prompt U
:Disp " HOW MANY "
:Disp " INTERVALS "
:Prompt N
:L → Xmin
:U → Xmax
:ClrHome
:Disp " SCALING GRAPH "
:Disp " PLEASE WAIT "
:fMin( Y1 , X , L , U ) → A
:fMax( Y1 , X , L , U ) → B
:Y1( A ) → Ymin
:Y1( B ) → Ymax
:If Ymin > 0
:0 → Ymin
:If Ymax < 0
:0 → Ymax
:( U - L ) / N → W
:0 → S
:L + W / 2 → T

```

```

ENTER FUNCTION
(IN QUOTES)

```

```

Y1 = " X2 + 2 "

```

```

LOWER LIMIT
OF X
L = ? -1
UPPER LIMIT
OF X
U = ? 3
HOW MANY
INTERVALS ?
N = ? 6

```

```

SCALING GRAPH
PLEASE WAIT

```

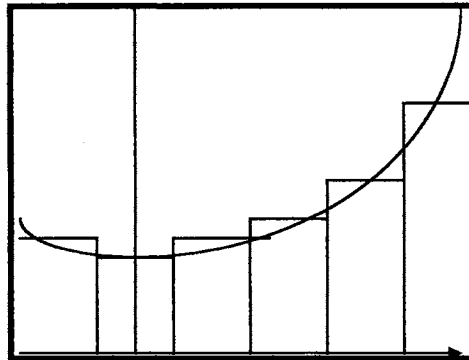
Continued...

**Sample Program 14 - Continued:**

```

:DispGraph
:For ( I, 1, N )
: Y1( T ) → P
: S + P * W → S
:Line ( T - W / 2, 0, T - W / 2, P )
:Line ( T - W / 2, P, T + W / 2, P )
:Line ( T + W / 2, P, T + W / 2, 0 )
: T + W → T
:End
:Pause
:ClrHome
:Disp " INTEGRAL ="
:Disp S
:Disp ""
:Disp " RUN AGAIN ? "
:Disp " (1) YES "
:Disp " (2) NO "
:Input R
:If R = 1
:Goto 1
:ClrHome

```



```

INTEGRAL =
17.18518519

RUN AGAIN ?
(1) YES
(2) NO
? 2

```

### Sample Program 15

This Calculus program will allow the user to enter a function and then display the tangent line equation and the normal line equation to the curve at a point specified by the user. The program will also graphically display both the tangent and normal lines.

#### Programming Plan of Attack

- (1) First, allow the user to enter the function (  $Y_1$  ).
- (2) Then allow the user to enter the value  $C$  for  $X$ . (  $X = C$  )
  - (a) If  $Y_1(C) = D$ , then the point on the original function is (  $C, D$  ).
- (3) Display the graph of the original function.
- (4) Draw the tangent line on the graph display.
- (5) Display the equation of the tangent line in slope-intercept form.
  - (a) Use the **nDeriv** function to find the slope (  $M$  ) of the tangent line at the point  $X = C$ .

- (b) The point-slope form of the tangent line equation is :

$$Y - D = M ( X - C )$$

In slope-intercept form this equation becomes :

$$Y = M * X - M * C + D$$

- (6) Pause until the user hits enter.
- (7) Then draw the normal line on the graph display.
- (8) Display the normal equation in slope-intercept form.
  - (a) If the slope of the tangent line is  $M$ , the slope of the normal line will be (  $- 1 / M$  ).
  - (b) The point-slope form of the normal line equation is :

$$Y - D = ( - 1 / M ) ( X - C )$$

- (9) Finally, if the slope of the tangent line is zero (  $M = 0$  ), then the equation of the vertical normal line will be  $X = C$ .

**Sample Program 15 - Continued:****PROGRAM: NORMTAN**

```

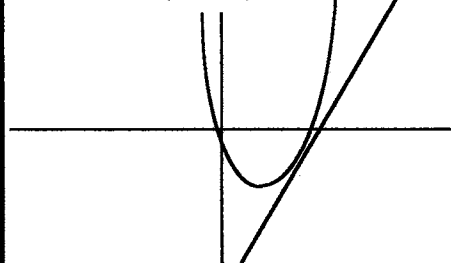
:ClrHome
:Disp "ENTER FUNCTION"
:Disp "( IN QUOTES )"
:Disp ""
:Input "Y1 =", Y1
:ClrHome
:Disp "NOW ENTER THE"
:Disp "X COORDINATE OF"
:Disp "THE POINT"
:Input "X =", C
:ClrDraw
:ZSquare
:Tangent ( Y1 , C )
:nDeriv ( Y1 , X , C ) → M
:Y1 ( C ) → D
:Text ( 1 , 1 , "TANGENT
LINE EQ" )
: ( - M * C + D ) → B
:round ( B , 2 ) → B
:Text ( 8 , 1 , "Y =", M ,
"X + ( " , B , " )" )
:Text ( 57 , 1 , "HIT ENT
ER FOR NORMAL EQ" )
:Pause
:
```

ENTER FUNCTION  
( IN QUOTES )

$$Y1 = (X - 2)^2 - 5$$

NOW ENTER THE  
X COORDINATE OF  
THE POINT  
X = 3

TANGENT LINE EQ  
 $Y = 2X + (-10)$



HIT ENTER FOR NORMAL EQ

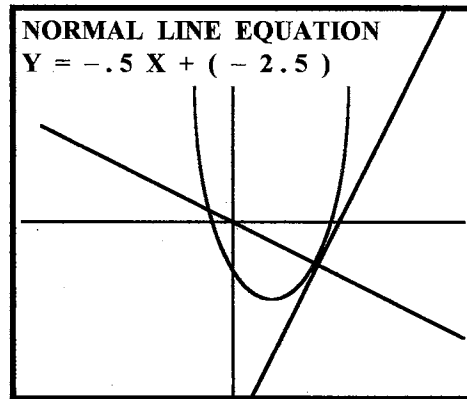
Continued...

**Sample Program 15 - Continued:**

```

:ClrDraw
:Tangent ( Y1 , C )
:Text ( 1 , 1 , "NORMAL
  LINE EQUATION" )
:If M ≠ 0
:Then
:( C / M + D ) → B
:round ( - 1 / M , 2 ) → M
:round ( B , 2 ) → B
:DrawF ( M * X + B )
:Text ( 8 , 1 , " Y = " , M ,
  " X + ( " , B , " ) " )
:Else
:Vertical C
:Text ( 8 , 1 , " X = " , C )
:End

```



APPENDIX C

STUDENT QUESTIONNAIRE



<h2 style="margin: 0;">Student Questionnaire</h2> <h3 style="margin: 0;">( Graphics Calculator Programming Manual )</h3>
--

(1) What grade level are you in this school year ?

9 \_\_\_\_\_ 10 \_\_\_\_\_ 11 \_\_\_\_\_ 12 \_\_\_\_\_ College \_\_\_\_\_

(2) What math course are you taking now or will be taking next semester ?

Algebra I \_\_\_\_\_  
 Geometry \_\_\_\_\_  
 Algebra II \_\_\_\_\_  
 Trigonometry \_\_\_\_\_  
 Calculus \_\_\_\_\_  
 Other \_\_\_\_\_

(3) Prior to using this calculator programming manual, have you had any previous experience in computer programming ? Yes \_\_\_\_\_ No \_\_\_\_\_

If Yes, please describe your previous experience:

\_\_\_\_\_  
 \_\_\_\_\_

(4) If you have had previous programming experience, did you have difficulty adapting those programming skills to the graphing calculator ? Yes \_\_\_\_\_ No \_\_\_\_\_

(5) Did you find this manual :   **Easy to use**        \_\_\_\_\_  
   **Reasonable to use**    \_\_\_\_\_  
   **Difficult to use**        \_\_\_\_\_

(6) How helpful would it be to you if more sample programs were included in this manual?                            **Not very helpful**    \_\_\_\_\_  
   **Somewhat helpful**    \_\_\_\_\_  
   **Very Helpful**        \_\_\_\_\_

(7) How valuable do you think your ability to write programs for your graphics calculator will be during your:   **High School career**    \_\_\_\_\_  
   **College career**        \_\_\_\_\_

(8) Please suggest any improvements you think would make this manual easier to use:

\_\_\_\_\_  
 \_\_\_\_\_  
 \_\_\_\_\_  
 \_\_\_\_\_  
 \_\_\_\_\_

APPENDIX D

TI CARES NEWSLETTER

## Reference List for the TI-82

---

These publications contain material relating to Texas Instruments calculators. You can order these materials directly from the [publisher](#). To add to this list, please write to us through our [feedback form](#), or send email to [ti-cares@ti.com](mailto:ti-cares@ti.com).

---

Alexander, Bob. **Exploring Quadratic Functions with the TI-83**. Students can use this book to explore quadratic functions while they learn how to use the TI-83 or TI-82. Topics include: Getting Started, Applications, Curves of Best Fit, Exploring Quadratic Graphs, Iterating Quadratic Functions, and Programming the TI-82 or TI-83. Answers are included.

Alexander, Bob. **Investigation with the TI-82 Graphics Calculator, What If? The Straight Line**. MathWare.

Ball, Stuart. **Programs for the TI-81 and TI-82 Calculators**. (A great resource book which contains 70 programs covering algebra, statistics, trigonometry, calculus and other topics. Programs have a line-by-line menu location for all key entries. A useful appendix shows sample programs in each program.)

Best, George W., and David A. Penner. **Using the TI-82 to Explore Precalculus and Calculus**. Andover, MA: Venture Publishing, 1994. ISBN 1886018065.

Bjork, Lars-Eric and Hans Brolin. **Calculus Explorations with the TI-82**. Give your students the calculator advantage by having them become familiar with calculus on the TI-82. Professor Hans Brolin of the Department of Teacher Training at Uppsala University in Sweden, one of the co-authors, has become well-known to many in the United States from presentations at NCTM meeting over the past four years. The aims of the book are to help the student to become familiar with the calculator and, by entering the programs from the book, to provide a powerful tool for studies in calculus. Many exercises and all answers included. MathWare. ISBN 0-9623629-5-6.

Bowen, Jim. **The High School Student's Guide to Programming the TI-82 and TI-83**. Call 800-TI-CARES (800-842-2737) to order.

Best, George W., and Sally Fischbeck. **AP Calculus with the TI-82 Graphics Calculator**. Andover, MA: Venture Publishing 1995. ISBN 1886018073.

Carlson, Ronald J, and Mary Jane Winter. **Algebra Experiments II, Exploring Non-Linear Functions**. Reading, MA: Addison-Wesley Publishing Company, 1993. ISBN 0-201-81525-7.

## Reference List for the TI-83

---

These publications contain material relating to Texas Instruments calculators. You can order these materials directly from the [publisher](#). To add to this list, please write to us through our [feedback form](#), or send email to [ti-cares@ti.com](mailto:ti-cares@ti.com).

---

Alexander, Bob. **Exploring Quadratic Functions with the TI-83**. Students can use this book to explore quadratic functions while they learn how to use the TI-83 or TI-92. Topics include: Getting Started, Applications, Curves of Best Fit, Exploring Quadratic Graphs, Iterating Quadratic Functions, and Programming the TI-82 or TI-83. Answers are included.

Barrett, Gloria. **Statistics with the TI-83**. Based on core concepts taught in most statistics classes, the workbook features a series of well-developed chapters that cover concepts, examples, exercises, and extended exercises. In addition, ideas for student generated data sets are also provided. Meridian Creative Group.

Barton, Ray and John Diehl. **TI-83 Enhanced Statistics**. Venture Publishing.

Best, George and David Penner. **Exploring Algebra, Precalculus and Statistics with the TI-83 Graphing Calculator**. A book that is filled with examples and exercises that explain how the TI-83 can be used to teach concepts and problem solving techniques in algebra, pre-calculus and statistics. It includes interesting exercises, probing and instructive projects, technology tips, and a through explanation of how to program the TI-83. Andover, MA: Venture Publishing, 1997. ISBN 1-886018-06-5.

Bowen, Jim. **The High School Student's Guide to Programming the TI-82 and TI-83**. Call 1-800-TI-CARES to order.

Cochner, Deborah J. and Bonnie M. Hodge. **Explorations in College Algebra Using the TI-82/TI-83: With Appendix notes for the TI-85**. Sing these unique and user-friendly workbooks, readers quickly learn to use the graphing calculator to develop problem-solving and critical-thinking skills. Hands-on applications with solutions; key charts that show which units introduce keys of the calculator; and a troubleshooting section are included. 1997, Brooks/Cole Publishing. ISBN 0-534-34228-0.

Cochner, Deborah J. and Bonnie M. Hodge. **Explorations in Precalculus Using the TI-82/TI-83: With Appendix notes for the TI-85**. Sing these unique and user-friendly workbooks, readers quickly learn to use the graphing calculator to develop problem-solving and critical-thinking skills. Hands-on applications with solutions; key charts that show which units introduce keys of the calculator; and a troubleshooting

APPENDIX E  
INSTITUTIONAL REVIEW BOARD  
APPROVAL FORM

OKLAHOMA STATE UNIVERSITY  
INSTITUTIONAL REVIEW BOARD  
HUMAN SUBJECTS REVIEW

Date: 05-06-96

IRB#: ED-96-124

Proposal Title: THE DEVELOPMENT OF AN INDEPENDENT STUDY GUIDE TO PROVIDE INSTRUCTION TO HIGH SCHOOL STUDENTS IN WRITING PROGRAMS FOR THE TI-82 GRAPHING CALCULATOR

Principal Investigator(s): Steve Marks, James G. Bowen

Reviewed and Processed as: Exempt

Approval Status Recommended by Reviewer(s): Approved

ALL APPROVALS MAY BE SUBJECT TO REVIEW BY FULL INSTITUTIONAL REVIEW BOARD AT NEXT MEETING.

APPROVAL STATUS PERIOD VALID FOR ONE CALENDAR YEAR AFTER WHICH A CONTINUATION OR RENEWAL REQUEST IS REQUIRED TO BE SUBMITTED FOR BOARD APPROVAL.

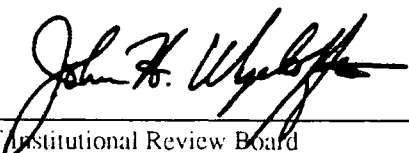
ANY MODIFICATIONS TO APPROVED PROJECT MUST ALSO BE SUBMITTED FOR APPROVAL.

---

Comments, Modifications/Conditions for Approval or Reasons for Deferral or Disapproval are as follows:

It's suggested that participating subjects be given a copy of the solicitation form to take home.

Signature:

  
\_\_\_\_\_  
Chair of Institutional Review Board

Date: May 15, 1996

2

VITA

James G. Bowen

Candidate for the Degree of

Doctor of Education

Thesis: THE DEVELOPMENT OF AN INDEPENDENT STUDY GUIDE TO PROVIDE INSTRUCTION TO HIGH SCHOOL STUDENT IN WRITING PROGRAMS FOR THE TI-82 AND TI-83 GRAPHICS CALCULATORS

Major Field: Applied Educational Studies

Biographical:

Education: Graduated from Camp Springs High School, Camp Springs, Maryland in May 1966; received Bachelor of Science degree in Mechanical and Aerospace Engineering from Oklahoma State University, Stillwater, Oklahoma in May 1971; received a Master of Science degree in Mechanical Engineering from Oklahoma State University in May 1972; received a Master of Science degree in Secondary Mathematics Education from Oklahoma State University in May 1985; completed the requirements for the Doctor of Education degree at Oklahoma State University in May 1997.

Professional Experience: T-38 Instructor Pilot, USAF, 1972-1975; Petroleum Engineer and District Manager, Halliburton Services, Indonesia, 1975-1983; Math Teacher, Stillwater High School, Stillwater, Oklahoma, 1985-Present.

Professional Membership: National Council of Teachers of Mathematics.