ABSTRACT

Algebra Maker and Manipulator is a teaching utility targeted toward math teachers. It is a menu-driven system that allows a teacher the ease of producing and manipulating algegraic math problems. Its main features include full-screen editing of text, graphics design and editing of mathematical symbols, file management of the problems, and printing facilities.

The system brings together the typing and drawing capabilities necessary to create an algebraic problem. It also maintains a filing system that alleviates the need for year-to-year redrawing of problems.

TABLE OF CONTENTS

INTRODUCTION

Society's emphasis on computers has two general effects on education. One is that educators are asked to teach students about computers; the other is that educators are encouraged to use them to improve learning in the classrooms. Teachers can improve learning in the classroom by using the computer to automate the creation and filing of tests. This allows a more feasible means of individualized instruction and diversified teaching.

Currently, most teaching utilities are computerized gradebooks, spelling checkers, and data bases of test questions for textual content areas such as history. Some software -- not necessarily targeted at educators -- provides mathematical formula and figure editing, while other software provides file or data base management of numerical or textual data. There seems to be little current software that combines the features of an editor and a file management system for mixed data types.

The general approach to the Algebra Maker and Manipulator was governed by learning the following basic techniques, with indicated references:

- designing bit image graphics characters; (1)
- directly accessing video memory on an IBM PC; (7)(8)
- writing and reading to/from video memory; (7)(8)
- menu design and handling
- file I/O in C

ENVIRONMENT OF THE PROJECT

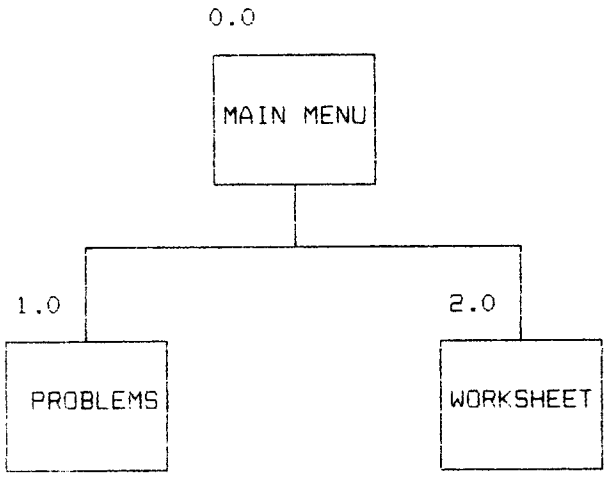
The project was implemented for the IBM PC and compatibles. Development took place using a Tandy 1000 with 640k RAM, two floppy disk drives, and a 20 Megabyte hard disk.

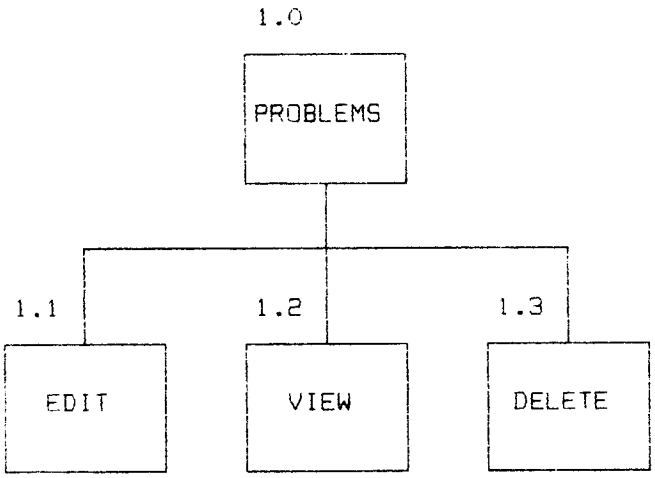
Programs were implemented with the C programming language and Microsoft C 5.0 Optimizing Compiler.

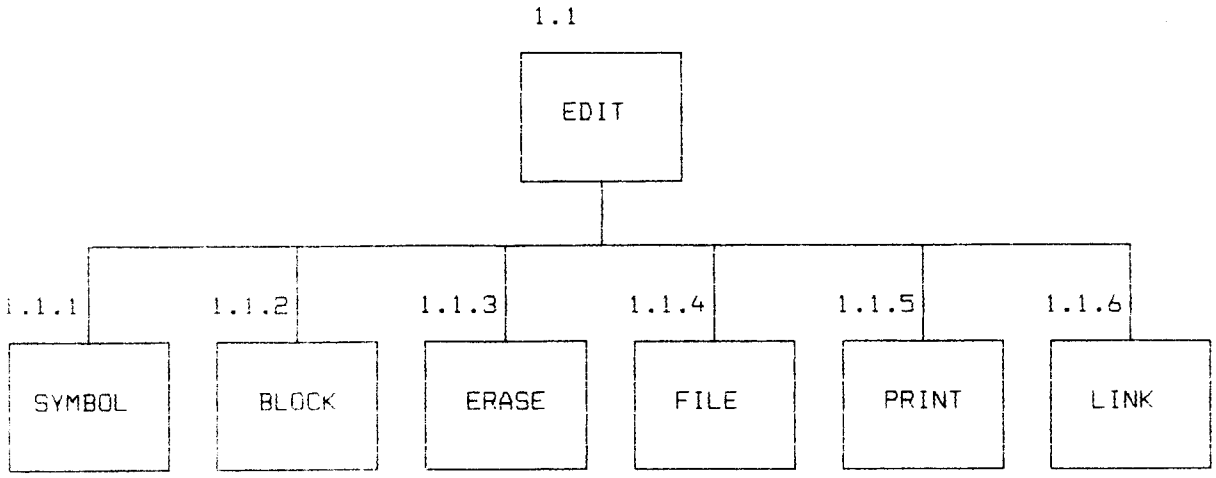
Printing was implemented for the IBM Proprinter and printers with compatible control sequences.

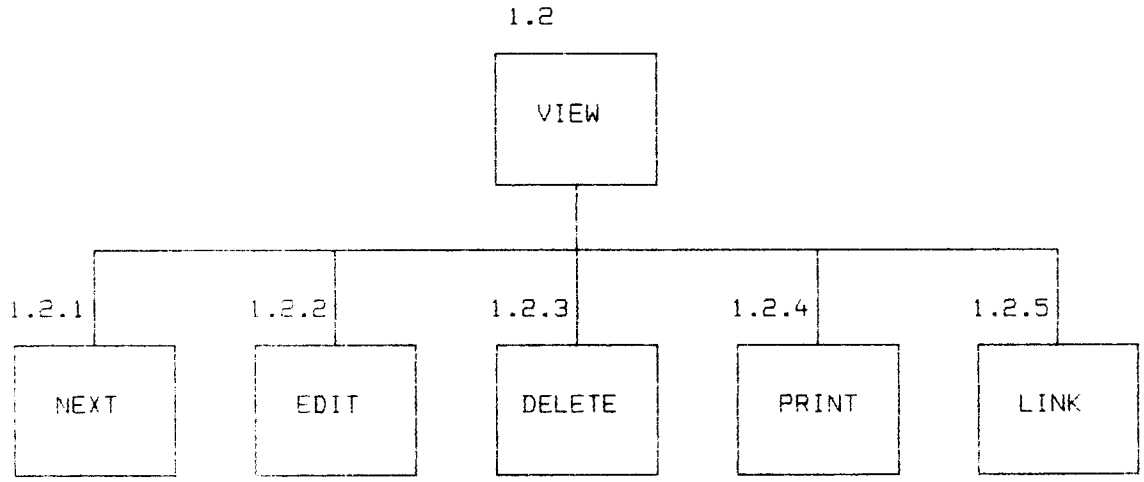
MODULE STRUCTURE CHARTS

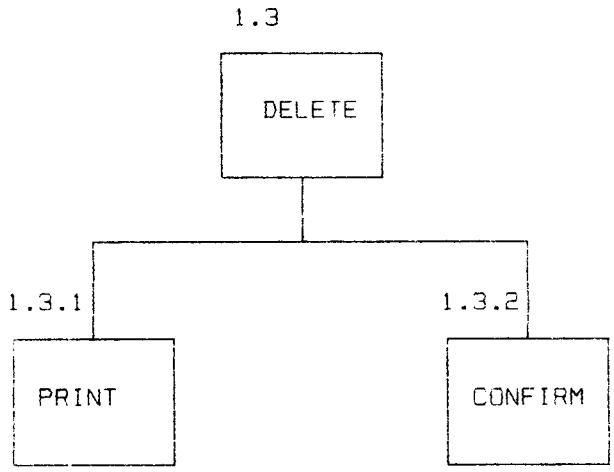
This is a self contained module that serves all of the other modules.

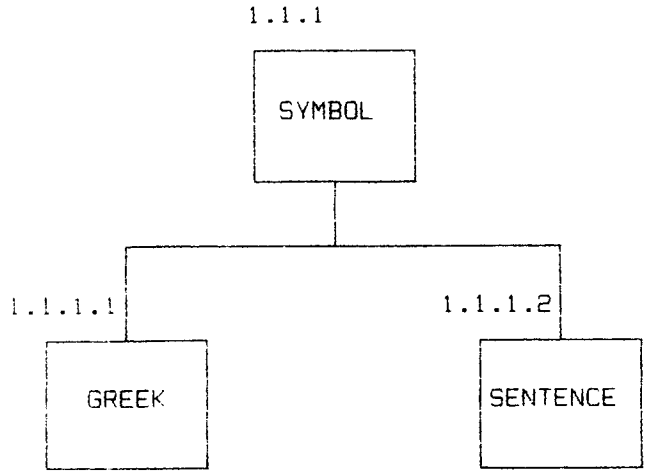


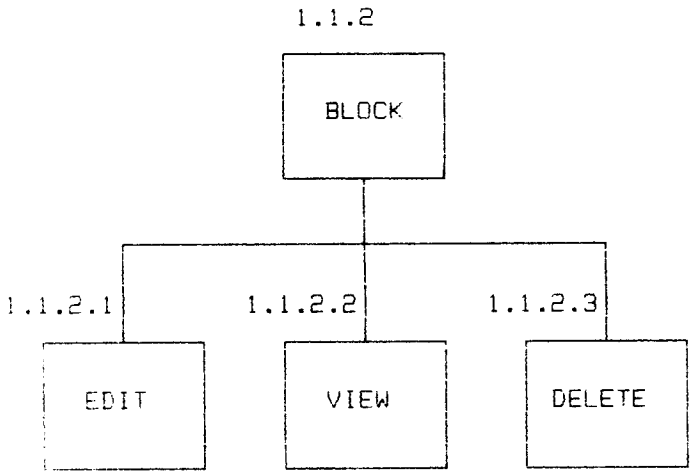


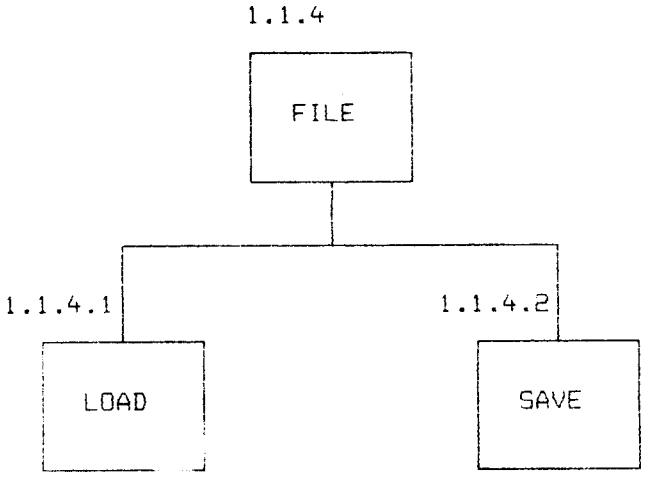

1.1
EDIT
1.1.1
SYMBOL
1.1.2
BLOCK
1.1.3
ERASE
1.1.4
FILE
1.1.5
PRINT
1.1.6
LINK

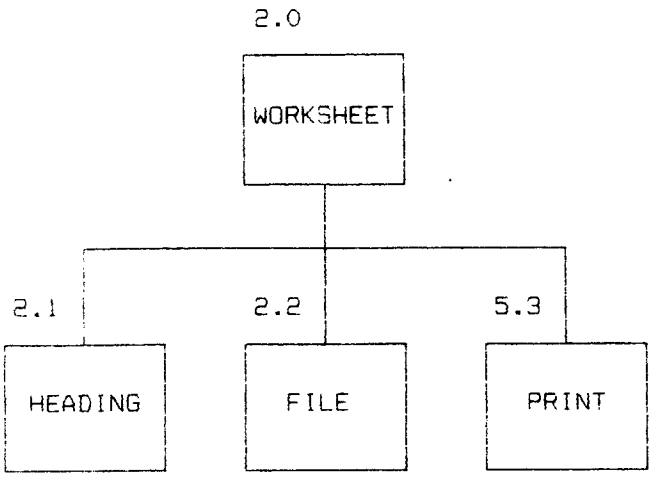

1.2
VIEW
1.2.1
NEXT
1.2.2
EDIT
1.2.3
DELETE
1.2.4
PRINT
1.2.5
LINK


1.3
DELETE
1.3.1
PRINT
1.3.2
CONFIRM


1.1.1
SYMBOL
1.1.1.1
GREEK
1.1.1.2
SENTENCE


1.1.2
BLOCK
1.1.2.1
EDIT
1.1.2.2
VIEW
1.1.2.3
DELETE


1.1.4
FILE
1.1.4.1
LOAD
1.1.4.2
SAVE


2.0
WORKSHEET
2.1
2.2
5.3
HEADING
FILE
PRINT

FILE DESCRIPTIONS

The Algebra Maker and Manipulator system uses two files, a problem file and a worksheet file.

The files are accessed in binary mode, which differs from text mode in newline and EOF translation. In text mode, a newline is translated into the carriage return-linefeed (CR/LF) combination before being written to disk. Likewise, the CR/LF on the disk is translated back into a newline when the file is read by the C program. However, if a file is opened in binary mode, as opposed to text mode, these translations will not take place. In text mode the special character 1A hex (26 decimal) is inserted after the last character in the file to indicate EOF. Both modes keep track of the total length of the file and will return an EOF when this length has been reached. Binary mode does not recognize 1A hex as an EOF. Binary mode only returns an EOF when the total length of the file has been reached.

The files in this system are represented in binary format which differs from text format in number representation. Binary format stores numbers in the same format used to store numbers in memory. Since video memory is being stored, this method of number representation was the deciding factor in choosing binary files over text files.

Both the problem file and the worksheet file use fixed length records.

PROBLEM-FILE DESCRIPTION

The first item stored in a problem record is the HIGHY. In high resolution, the y values range from 0-199, with 0 being at the top of the screen and 199 at the bottom of the screen. HIGHY is the highest y value of a row with a pixel turned on. The HIGHY is used to reduce load time and print time. HIGHY is stored as a CHAR and uses one byte of storage.

After the HIGHY, each byte of the screen image is stored as a CHAR. The last print row of an edit screen contains the status bar displaying the current line, column, and mode. The status bar is not stored. Therefore, a screen image needs 15360 bytes -- 640 x 192.

WORKSHEET-FILE DESCRIPTION

The first item stored in a worksheet record is the HEADER, or title, to be printed. The HEADER field is a string of seventy-eight CHARS which uses seventy-eight bytes. Following the HEADER are the problem numbers of the problems in the worksheet. A worksheet may contain a maximum of fifty problem numbers. A problem number is stored as an INT which uses two bytes; consequently, the worksheet problem numbers use a total of one hundred bytes.

SYSTEM DEVELOPMENT

Development of the system began with the design of a character font and the routines to display a character of the font on the screen.

The character font is defined in font.txt. Each character is eight pixels high and eight pixels wide. The program font.c reads font.txt and produces the include file font.h .

A character is placed on the screen by copying to video memory the rows of pixels which compose the character. Put.c contains the routines to put characters on the screen.

Once a character could be put on the screen, development proceeded with the design of the menu data structures in menustr.h and the routines to process a menu. There are three different types of menus: a menu bar, a pulldown, and a popup. Make.c handles the making of menus. Main.c controls the menu traversal through one simple loop that switches on the current menu bar level. Menu traversal is based on the level numbers assigned in levels.h . These level numbers are also reflected in the module structure charts. Dynamic memory allocation is used to save and restore the images under the menus. The entire system was implemented with "stubs" before each actual module was written.

After implementing the framework of the system, functional modules were developed one at a time. The logical framework of the modules is represented in the module structure charts.

The scope of each module is described below:

HELP: This module can be entered from any depth in the menu-driven system by pressing F1. The point of call determines the help provided.

MAIN: This module allows the user to process PROBLEMS or WORKSHEETS.

PROBLEMS: This module allows the user to EDIT, VIEW, or DELETE problems.

WORKSHEET: This module allows the user to define the worksheet HEADING, load or save a worksheet from the FILE, and PRINT the active worksheet.

EDIT: This module allows full screen editing with the options SYMBOL, BLOCK, ERASE, PRINT, and LINK; and options to load or save a problem from the problem FILE.

VIEW: This module allows the user to view the NEXT problem, chain to the EDIT module, chain to the DELETE module, PRINT the problem, and LINK the problem into the active worksheet.

DELETE: This module allows the user to PRINT the problem and CONFIRM a deletion of the problem.

SYMBOL: This module allows the user to incorporate into a problem symbols which cannot be typed from the keyboard.

BLOCK: This module allows the user to perform the block operations of SET, MOVE, and ERASE on blocks of a problem.

SET: This module allows the user to SET a block. The function expects the cursor to be at the top left corner of the block. The cursor keys are used to highlight the block.

MOVE: This module allows the user to MOVE a block by pressing cursor keys. The MOVE function expects a block to have been SET. The MOVE function moves a block half of a print position (4 pixels) in the specified direction.

Also, implementation of the edit module required the design and programming of specific key functions.

The scope of the function of each such key is described below:

<home> - beginning of current row on screen.

<ctrl><home> - beginning of first row of document.

<end> - end of current row on screen.

<del> - delete character under cursor.

<bksp> - move cursor left when in replace mode or delete character to left of cursor when in insert mode.

<ins> - toggle between insert and replace mode.

<esc> - escape from current activity or menu level.

<alt><e> - entry of single character exponent.

<ctrl><-> - underscore mode.

<F5> - parenthesis mode.(for adjustable size parentheses)

<F6> - bracket mode. (for adjustable size brackets)

<ctrl><r> - radical mode. (for adjustable size radicals)

<ctrl><u> - undo the last mode command.

<F10> - activate the edit menu bar.

A SAMPLE SESSION WITH AMM

Note: User input from the keyboard is placed in angle brackets <>. Press ESC to escape the current activity or menu level. Press <ENTER> to select a menu option. Press F1 for help.

1. Insert the system disk in drive A. Turn on the computer or reboot the system.

2. Press any key at the greeting screen and the main menu appears with the HEADING option highlighted. To initiate an active worksheet, the first step is to choose this option and enter a heading.

3. To edit a problem, move the cursor to the EDIT option of the PROBLEM module and press <ENTER>.

4. Become familiar with the keys implemented in EDIT by hacking or consulting the help screen. A bell sounds if you press a key that is not implemented. Watch the status bar for information concerning key combinations. Press F10 to display the EDIT menu.

5. Now, edit a formula to demonstrate some of the various functions. Press F10 and select the ERASE option. This erases the entire screen of your work.

6. Move the cursor over and down several times to give yourself some room to work. The formula you are going to create is the Quadratic formula.

7. Begin typing: <X><space><=><space><-><b><space>. You need the symbol ± . Press F10 and select this sentence symbol. The symbol appears where you were typing.

Continue typing: <space> <b>.

You need an exponent of 2. Hold down <alt> and press <e>. Type <2>. An upraised 2 appears after the b.

Continue typing: <space> <-><space><4><a><c>.

Now you need a radical. Move the cursor over to the b^2. Hold down <ctrl> and press <r>. Use the right cursor key to highlight the text to be under the radical. Press <ENTER> when you have finished highlighting the text to be underscored. A radical appears. However, you did not leave enough room for the radical. Hold down <ctrl> and press <u>. The screen is restored. Press <insert> and insert a space before the text to go under the radical. Press <insert> again to return replace mode. Position the cursor at the b^2 and make the radical again.

Now, you need to move the radical portion up to form a numerator. Position the cursor over the top left position that includes a portion of the radical. Press F10 and select the BLOCK SET option. Move the cursor right and down to highlight the block containing the radical. Press <ENTER> when you are done. You have returned to the BLOCK menu. Select the MOVE option. Press the up arrow once to move the block up. Press <ENTER> when you are done. The block moves up half of a print position.

Now, you need to underscore the radical portion. Position the cursor over the bottom left position that includes a portion of the radical. Hold down <ctrl> and press <->. Move the cursor right to highlight the block to be underscored. Press <ENTER> when you are done. The block is underscored.

Now, type the denominator: <2><a>.

You are done!

You can save the problem by selecting the FILE SAVE option of the edit menu. You can link the problem into the active worksheet by selecting the LINK option.

FINAL RESULTS OF THE PROJECT

The final system resulting from this project is a menu-driven system that allows a teacher the ease of producing and manipulating algebraic math problems. Algebra Maker and Manipulator is a user-friendly graphics editor and file management system.

The menu-system and the menu traversal technique run smoothly and the editing functions are efficient. Both include error handling routines for user input.

The file I/O is still not at a marketable state. Further study of the C I/O library routines and data storage techniques is needed to determine a better way of achieving decent response time and smaller file sizes.

More Greek and sentence symbols and editing functions should be implemented for more versatility.

A major revision which should significantly improve the system is a data base management system instead of a file management system. This would allow the user more power over the organization and categorization of problems.

Another area that needs further study concerns printer control. Presently, the output is not satisfactory for duplicating from spirit masters or thermal masters.