DATATRANS--A USER-FRIENDLY FILE TRANSFER UTILITY

GRADUATE PROJECT

BY
CHARLES B. SWANNER

November 19, 1994

COMMITTEE MEMBERS

Dr. Herbert R. Haynes,
Chairman

Dr. Roy S. Ellzey,
Member

Mr. David C. Perkins,
Member
ABSTRACT

The project is the design and implementation of a file transfer utility program called DataTrans. DataTrans copies selected records from an input to an output file, reformatting each record according to user-specified field locations. Reformatting is described in terms of fixed-length columnar fields. DataTrans provides an easy-to-use graphical user interface as the means for defining and saving data transfer information. The define and edit functions allow the user to describe the formats of one input file and one output file. The output of these functions is a binary file containing the description of transfer. This format information, along with the names of the files containing the data to be transformed and where resulting data are to be placed, provide input to the transformation function. This transformation function performs one file transfer from the input format to the output format given in the transfer definition. A list function provides a convenient transfer definition check and correction method. A viewing function allows the user to check transfer output, including error and rejected record logs, without leaving DataTrans. DataTrans is written using Visual Basic 3.0, in and for the Microsoft Windows 3.1 operating environment. DataTrans has the look and feel associated with a typical MS-Windows application.
# TABLE OF CONTENTS

Abstract ........................................................................................................... i
Table of Contents ........................................................................................... ii
Introduction ..................................................................................................... 1
Environment .................................................................................................... 2
Background and Rationale ........................................................................... 3
Narrative .......................................................................................................... 4
Procedure ....................................................................................................... 9
Results and Conclusions ............................................................................. 15
Bibliography ................................................................................................. 17
Appendix
   A ................................................................................................................. User's Guide
   B ............................................................................................................... Reference Guide
   C ................................................................................................................. Program Listing
   D ............................................................................................................... Exhibits
INTRODUCTION

DataTrans copies selected records from an input to an output file, reformatting each record according to user-specified field locations. Reformattting is described in terms of fixed-length columnar fields. A field is a datum, like social security number, or address. In addition to field locations, the user may specify that an input field is not empty or does not contain blanks. During a transfer, only lines of input whose fields satisfy these specifications are written to the output file. A set of these field descriptions is called a "transfer definition." DataTrans provides a means for creating, changing, correcting, saving, and executing transfer definitions in an organized manner. This lowers the skill level and work effort which would otherwise be required. DataTrans facilitates the transfer of information produced by one program into a form that can provide input to another. This project is the initial release of DataTrans, together with its User's and Reference Guides. On-line contextual help is available.

The organization of the DataTrans menu reflects the three-stage process of defining, performing, and checking the results of one data transformation. The input and output files of the data transfer stage are ASCII text files. The data transfer definition is stored in a binary format.

In addition to field location information, a transfer definition contains two fields, If Not All Filled Ignore Line and If Null Ignore Line, which allow the transfer stage to reject records not satisfying these criteria. For example, the transfer stage may reject lines not containing a nine-character social security number in the correct columns. This mechanism can be used to distinguish report header, total, and other non-detail lines, so that they are not transferred.

In the third stage the user may view and, optionally, print text files. The view selection consists of a scrolling list box to be used primarily for checking the output data. However, record rejections, the log describing the transfer process, or the input data may also be viewed and printed, without leaving the DataTrans environment.

The typographical conventions used in the document are as follows:

<table>
<thead>
<tr>
<th>Formatting convention</th>
<th>Type of information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bold type</strong></td>
<td>menu selections, program name</td>
</tr>
<tr>
<td><em>Italic type</em></td>
<td>command button captions</td>
</tr>
<tr>
<td>Monospaced font</td>
<td>list, check box, button labels</td>
</tr>
</tbody>
</table>
ENVIRONMENT

DataTrans operates in an MS-Windows environment using version 3.1 or greater. The programming language used is Visual Basic 3.0. The program is designed to work with an 80386 processor or better on a system with a minimum of four megabytes of memory, and a monitor and video card capable of 640 by 480 or greater resolution. 800 x 600 resolution or greater is recommended. A minimum of two megabytes of disk storage is required.
**BACKGROUND AND RATIONALE**

**DataTrans** provides a foundation for the development of a commercially-viable file transfer utility. The utility fills a gap in commercial software packages, which are focused on the transfer of data into more esoteric formats rather than the **DataTrans** concept of data transfer into a different fixed-field format. For example, data transfer utilities like Data Junction or those provided with Word Perfect perform functions such as translation of a Microsoft Word formatted document into one in Word Perfect format. This basic implementation of **DataTrans** allows the definition of, as well as the storage of, the definition of, the translation of a columnar file format into a different columnar format. In addition, this translation process may reject a line of input data that does not satisfy user-specified criteria. Data can be transformed into a format which can provide input to one of the more specific translators. That translator can then transform the data into the format needed by the target application. **DataTrans** can also take a report that has been saved as a file and convert that report into an input format to be used by another application. When a previously defined format is needed, **DataTrans** provides access to that format in an organized, easily retrievable manner.

Primary design concerns were that **DataTrans** be developed and operated in an environment providing a common user interface and that the environment chosen continue to be developed and supported. **DataTrans** was originally proposed for the MS-DOS environment. Borland's Turbo-Vision product was used. In 1992 Borland discontinued development of Turbo-Vision and curtailed or discontinued their development of all MS-DOS based projects. The chair of this project committee suggested that Visual Basic be substituted. Program development began again using Visual Basic for MS-DOS. It was found that the window frame controls were incomplete and that memory control was inadequate. By the middle of 1994 it appeared that other major software developers were curtailing their development in the MS-DOS environment in favor of one using MS-Windows. Microsoft had no plans for a new version of the MS-DOS based Visual Basic. Based on these problems and observations, Visual Basic for MS-Windows was selected. MS-Windows and Visual Basic for Windows have provided a viable development environment without compromising the original design concerns.
NARRATIVE

The execution of the program is divided into three stages. Each stage corresponds to a menu bar title and each function of that stage corresponds to a submenu selection under that title. The first stage, Define, allows the user to create, modify, save, delete, list, and recall descriptions of the input expected and the output produced by one transfer. For each field, the user defines the input beginning and ending columns and the output beginning and ending columns. Optionally, the user may specify whether this field's transfer process should look at the input data field for two possible criteria. If the user checked If Null Ignore Line a line of input will be rejected if only blanks are found in this input data field. In a like manner, checking If Not All Filled Ignore Line will cause an input line to be rejected if any blank characters appear in the input data field.

The second stage, Transfer, allows the user to recall a transfer definition, enter transfer definition parameters, define data file names, and perform a data transfer. The user may recall a stored definition or use one that was specified in the definition stage. This definition, along with an input file specification and an output file specification, makes up the input necessary to create the output file in the format described in the definition of transfer. The user may change the name of the input, output, transfer definition and, optionally, the error log and/or the rejected records file. In the parameters menu selection, a user may specify that he wants the transfer process to be suspended if some of the output is going to be overwritten by another output field. He may also specify whether that output should be overwritten. Regardless of the value of these parameters, an overwrite condition will cause the contents of the field that will be overwritten and the overwriting field to appear in the error log.

In the third stage, View, the user may view and/or print the contents of a text file on disk. The user will probably be most interested in checking the results of a data transformation by viewing the output file contents. However, the user may also want to view the contents of the error log file, rejected records file, or input file.

The program can handle up to twenty field descriptions. The field descriptions are stored in an array. Each field is associated with a data entry form on the screen. If the array index value is n, the form number associated with that array element is Form(n). For example, the array element accessed by the value 1 is associated with Form1. These entry forms cascade on the screen in approximately one-third inch increments, provided the video resolution is at least 800 x 600. Deleting a field description is accomplished by placing a zero in any column location and saving the file.

Define

The first item on the main menu is Define. This corresponds to the first major stage of the program and consists of several submenu selections on a pull-down list which perform the functions involved in the definition of the fields to be included in the transfer process. The
Define menu label heads the following menu selections:

New   Edit   Store   Delete   List   Retrieve

Selecting New causes the array of field definitions to be cleared of any previous values. Once the array is initialized, the first data entry form (Fig. 1 - Appendix D) is loaded (causing the implicit load of the background window form MDIForm1.frm) and the user is able to define a field's column locations. Column numbering begins with one. If the If Not All Filled check box is marked, a blank appearing in the input field will cause a data line not to be transferred to the output file. The If Null Ignore Line check box will cause a data line not to be transferred if the input data field is all blank.

Four command buttons are available: Next, Previous, List, and Done. When one of them is selected, the field definition data is stored in the transfer definition array record in the element with an index corresponding to the current form number. Next loads the data entry form for the current index entry plus one. Previous returns the user to the current form minus one. As an example, selecting Next when form two is active will cause form three to be loaded for entry, while selecting Previous will invoke the unloading of form two, returning the user to data entry form one. Should the user select List, the definition browser form (Fig. 2 - Appendix D) will present a list where the user can check the transfer definition in one window. The list will contain one field definition per line, rather than one per form. The data fields corresponding to If Null Ignore Line and If Not All Filled Ignore Line will be listed as Y, if the box was checked, or N if it was not.

When the user exits the data entry form or the browse list form, he is prompted for a file name under which the transfer information may be saved to disk. The browser has two command buttons, Done and Editor. Done closes all open forms and returns the user to the main menu form. Editor closes all open entry forms and reopens the forms up to the form number corresponding to the line number the user was on when Editor was chosen. For example, the user specifies input columns 1-3, output columns 41-43, and Not Empty. Next is selected, and the process continues, until he is Done. Choosing List, the user checks his work on one screen. He makes corrections by going to the line item in the list box that needs correcting and choosing Editor. The data entry form corresponding to that line number appears. Making the corrections, he chooses List and returns to the browser. The process may be repeated until a correct definition is obtained. Selecting Done causes the user to be prompted, through the common dialog list box, for a file name that the contents of the transfer definition array may be stored under. The file name extension will default to "def," although the user may override this.

Input: User-supplied by data entry.

Output: The transfer definition array describing the input and output fields for this data transformation and a binary data file reflecting those formats.

The Edit function performs in a manner similar to that of New with the exception that the user is first presented with a common dialog list box requesting that an existing file name be entered.
The transfer definition array is then loaded with the definition from this file. The user can correct or modify this information. If a file specification has previously been entered the dialog box will default to that name. The interface is identical to that of New except that corrections or additions are being made to existing data. When the Done selection is made, the user is prompted to save his work through the common dialog interface mechanism.

Input: A text file describing the input and output fields for one data transformation and user-supplied data entry.

Output: The transfer definition array describing the input and output fields for this data transformation and a new or an updated binary file reflecting those formats. A default file name under which transfer definition changes may be saved.

The Store selection allows a transfer definition to be given a name and stored as a binary file under that name on disk through the use of the common dialog interface.

Input: The transfer definition array contents and a user-supplied file name.

Output: A binary file describing the input and output formats for one data transformation.

Choosing Delete causes the user to be prompted, through the common dialog interface, for a file name to be deleted from the disk. The user will then be asked if he is certain that he wishes to delete the file, and, if he does, the file is deleted from the disk.

Input: A user-supplied file name.

Output: A call to the system deleting the directory entry for the current description.

The List function is similar to that of Edit. The user is presented with the browser form, rather than the first data entry form, so that he may go directly to the entry form needing correction. He places the cursor on the line that needs correction in the list, activates Editor, and is presented with the form which needs correcting. Making corrections and choosing Done, he is prompted for a filename so that the work may be saved.

Input: Same as the Edit function.

Output: Same as the Edit function.

The Retrieve submenu choice will prompt for the name of a transfer definition file using the common dialog control. If an existing file name is given, the file contents will be loaded into the transfer definition array. In addition, that file name will become the default transfer definition file name.

Input: User-supplied name of file whose contents are to be retrieved.
Output: The filled transfer definition array and a default definition file name.

Transfer

The second major heading on the main menu is Transfer. This corresponds to the second major component of the program.

Transfer menu bar activation causes a pull-down menu to appear with the following selections:

Recall Definition Parameters Data Files Go

The Recall Definition menu selection allows the user to activate a transfer definition. Operation is similar to the Retrieve submenu selection discussed above.

Input: User-supplied name of file whose contents are to be recalled.

Output: The filled transfer definition array and a default definition file name.

Parameters allows the user to alter the default values for two check boxes (Fig. 3 - Appendix D). The two check boxes, Stop Transfer On Error and Overwrite Fields if multiply defined, provide input for the Go option discussed below.

Input: User-supplied preferences for the check boxes.

Output: Parameters that will be used during processing of data transfers.

The Data Files selection allows for user specification of the files used during a data transfer.

Input: File names for input, output, data transfer, error log, and record rejection files. May be user-supplied or default values.

Output: The user-supplied data file names that the program will use as defaults in any other function requiring the corresponding file name parameter.

The Go option causes an actual data transformation. If the transfer definition and data files have been specified, the user is presented with a message box displaying these names and asked if he wishes to begin the transfer using these values. A Yes response to the message box interaction will begin the transfer process. Otherwise the user may enter, through the common dialog interface, appropriate files names for this transfer. If the files have been properly named and the user has not chosen to cancel the operation the transfer process begins. The log and discard files will be updated with exception information as it occurs. An error message will be displayed if the process should stop, allowing the user to cancel or continue the process. When the end of the input file is reached a message displaying the names and location of the data files and an indication of success will be displayed.
Input: The current transfer definition, input and output file names, transfer definition parameters, and data file names.

Output: A log file containing a record of what happened during processing, a discard file containing records rejected because they did not fit the parameter or input field specifications, an error message box should the process terminate prematurely, a message box displaying the success information, and the data contained in the output file.

View

This third major menu heading contains no submenu selections. The user is prompted for a file name; a message box asks if the file is to be printed also; and the contents of the file are displayed in a scrollable window. The window contents are limited by their Visual Basic implementation to 32K bytes. Since the entire file normally would not need to be viewed to check the results of a transformation, the view is limited to the first five hundred lines of the file. The entire file, however, will be printed. Done, the only command selection available, returns the user to the main menu.

Input: A file specification and whether the file should be output to the printer also.

Output: The contents of the file to the screen and, if the user chooses, the printer.
PROCEDURE

In this section, an introductory paragraph is followed by a description of the declarations section and the main menu form load procedure. This is followed by detailing each submenu selection of the MainMenu form grouped by main menu heading. A discussion of pertinent form attributes is then followed with a short narrative of the event handlers. The section is completed with a discourse on individual procedures in the code-only module, Main.

Visual Basic 3.0 Professional Edition was used to create DataTrans. Several forms were created using the Form Designer tool available in Visual Basic. These forms are MainMenu, MDIForm1, Form(n) (where n varies from 1 to MaxRecords), ParmForm, ListForm, and Viewer. Pictures of the data entry form (Form(n)), the browse list form (ListForm), and the parameters form (ParmForm), are listed in Appendix D. The properties and code sections of the forms are appear in the Program Listing in Appendix C. These custom form definitions contain a description of the forms' look and default settings as well as procedure code that is executed when some event occurs in that form.

In addition to these custom forms, the built-in common dialog and message box forms were used. The message box is a window with a title, a message, a possible icon, and an answer. The message box is useful for displaying results, asking the user if he wishes to continue, etc. The common dialog control provides a method whereby the user may enter a file specification through a standard interface. The program module Main contains the definition and declaration sections, as well as the procedures not called directly as event handlers. The MainMenu form provides event handlers that act when one of the Define, Transfer, View, or Exit submenu selections is activated.

The constants, types and variables contained in the declarations section of Main provide the program data available at load time. The number of forms and size of the transfer record array are declared with the global constant MaxRecords (twenty in this version). Constants are defined for window sizing and placement, as well as for common dialog control, message box, and help messaging. The transfer information is stored in the Tran_Rec array, whose elements are of the Trans_Rec_Type, which is declared as a record containing six integers and a character. The records are stored on disk in a binary file in the form of one record of this type for each field. Four of the integers are for the input and output beginning and ending columns.

The remaining two integers are treated as logical variables which tell the Go option of DataTrans whether it is looking for the If Null Ignore Line or If Not All Filled Ignore Line attributes in this field. The corresponding transfer record components, NOT_EMPTY and ALL_FILLED, along with the input data, determine whether a line of input is valid output. The character field meant to hold the data type of the field being transferred is currently unused. A global array of these records is declared to be MaxRecords long. CurrentWindow will contain the form number corresponding to the record number in the array being edited. EditMode will be "True" if the user choose Edit or List. LeftField and TopField are used for the placement of the next data entry form in relation to its parent form MDIForm1. The Saved variable keeps track of whether a definition has been saved. The
OpenWins array of integers keeps track of which windows are open. InputDataFile, OutputDataFile, TransferDefFile, ErrorLogFile, and RejectedRecordFile are variables used to track the names of the Input, Output, Transfer Definition, Error Log, and Rejected Records file names, respectively. The StopOnTransferError and OverWriteFields variables allow the user to modify program behavior during a data transfer. This behavior will be discussed during the description of the data transfer process. These variables' default values may be modified by the user in the Parameters section of the Transfer menu selection.

Execution of DataTrans begins with a form load event for the MainMenu form. The help file is defined as "DataTran.hlp." As each form load event occurs its context identification number is set so the help system can find the correct help section for the current program context. The Saved variable is "True", EditMode "False", the input, output, and transfer definition file names are empty strings, and TopField, LeftField, and CurrentWindow are zero. The rejected record file and error log file variables are set to their respective default names, "DataTran.Log" and "DataTran.Err." The StopOnTransferError and OverWriteFields flags are "False" and "True", respectively. The InitOpenWins routine in Main is called to initialize the OpenWins array variables to "False."

Define heads the submenu selections New, Edit, Store, Delete, List, and Retrieve. Choosing New causes the click event for that submenu selection to be executed. EditMode is set to "False", the Tran_Rec array is cleared of any transfer definition values it may hold, and the first entry form (Fig 1 - Appendix D) is loaded. The Edit selection causes the common dialog control to ask the user for the name of the file to edit, sets EditMode to "True", and the variable TransferDefFile to the file name returned by the common dialog control. The contents of the file are loaded into the Tran_Rec array. The first data entry form is loaded with the values of the first array record. Subsequent data entry form load events cause the values in the Tran_Rec array element corresponding to the form number being loaded to be placed on the screen for possible modification. Selecting Store causes the SaveDef routine in Main to be executed. The user is prompted for a file name where the current contents of the Tran_Rec array will be placed. After the Saved flag is set to "True" control is returned to the MainMenu form. Selecting Delete, the user is prompted for an existing file name. Providing an existing file name causes the user to be asked if he really wants to delete the file. Finally, all files are closed and the kill command is used to wipe the directory entry from the disk. The List selection brings the user into the ListForm form as a means of checking the data or quickly accessing the form or forms that need correction. The final selection on this submenu, Retrieve, sets EditMode to "True" and uses the GetTransDefFile procedure in Main to ask the user for the name of an existing file. GetTransDefFile loads the Tran_Rec array with the files contents.

The second main menu selection, Transfer, heads the submenu selections Recall Definition, Parameters, Data Files, and Go. Choosing Recall Definition is functionally identical to the Retrieve selection discussed above, with the exception that EditMode is "False" so the contents of the file are not loaded into the array. The Parameters selection loads the form ParmForm, where the user is presented with the Stop On Transfer Error and Overwrite Fields check boxes.
The Go submenu selection causes the transfer of data. The number of lines transferred is initialized to zero, the error log file and rejected record file names are set to their defaults if they have not been previously specified. If input, output, and transfer file names have been previously specified the user will be presented with the names and given the option of continuing. Choosing not to continue causes prompting for new names. The transfer file specification is obtained by calling the GetTransDefFile routine in Main. Once file specification entry is completed, the user is presented with the names and given the option of continuing.

Choosing Yes causes the transfer to begin by opening the files and reading a line from the input file into a buffer. For each field, the file's text between the input beginning and ending columns is placed in the output buffer between the output beginning and ending columns. Fields are filled from left to right, with truncation of input data occurring should the output field be smaller than the input field. If the If Not All Filled Ignore Line flag is "True" and blank exist in the input data field a flag signifying that the data line is not to be written to the output file is set to "True." A "True" value in the If Null Ignore Line flag causes the same events should the input data field be all blank. The ErrorOccurred flag is set to "True" if some of the data in the output buffer is going to be overwritten by additional data in the input string. If the ErrorOccurred flag is "True" the line number and type of error will be reported in the error log, as well as the text that may be overwritten and the text that is overwriting. If the StopOnTransferError parameter is "True" program execution will stop and a message box displaying the error and transfer field definition number will be displayed. If an overwrite error has occurred and the user has chosen to overwrite existing output text the existing output will be overwritten, otherwise the existing output text is retained. This process continues with the next field in the data transfer definition until one output line is built. If either of the flags reflecting that a If Null Ignore Line or If Not All Filled Ignore Line condition occurred is "True" the line number and error type are printed in the error log and the output buffer contents are printed to the rejected records file. If not, the line is written to the output file and the number of lines successfully transferred counter is incremented. This process continues line by line until the end of the input file is reached. End of the input causes a success message box to be printed with the names and locations of the files used. The number of lines successfully transferred is displayed and printed in the error log. Finally, all files are closed.

The third menu bar selection, View, has no submenu selections. Selecting this option loads the Viewer form. This form load event prompts the user, through the common dialog control, for the name of a file that they wish to view. The user is then asked if they want to print the file as well. When the user chooses Done, the only command available, control is returned to the MainMenu form.

The Visual Basic list box control implementation has limited the contents of a list box to 32K bytes. The View option is typically used for checking the input column locations, the error log, the rejected records file, or the output. These files can be checked by viewing less than five hundred lines of the file. Therefore, the View list is limited to five hundred lines. The contents of the file are displayed in a scrolling window.

The form MDIForm1 provides a backdrop and a parent form for the data entry forms, Form1
through form20, and for the list form. The MDIForm1 load event sets the Height and Width attributes of the form to ninety percent of the screen and the top and left of the first form at about one-third inch from the top and left margins of the background form. The form unload event prompts the user to save his work by calling the subroutine SaveDef and setting the Saved flag to "True."

Data entry forms (Form1 - Form20) are identical with the exception of the Previous command on the first and the Next command on the last, which both beep when clicked. These forms provide the means by which the user enters information in the TranRec array. The form number corresponds to the index of the currently edited array element. Entry boxes appear on the form for integers corresponding to the input beginning and ending columns and the output beginning and ending columns. Two check boxes allow the user to specify If Null Ignore Line or If Not All Filled Ignore Line as requirements for this field if the data line is to be accepted into the output. Four commands are available to the user: Next, Previous, List, and Done.

Selection of any one of the commands causes the contents of the form to be stored to the correspondingly indexed element in the TranRec array. Next takes the user to that index plus one form number for data entry; Previous to the index minus one form number; List turns control over to the form ListForm; and Done closes all forms. A form load event for one of the entry forms results in a call to the FormLoad procedure in Main with the parameters form number, EditMode, and whether or not the form should be shown at this time. The form unload event causes execution of the FormUnload procedure in Main.

ListForm is a scrolling list box with two command buttons, Done and Editor. A transfer definition is viewed in the format of one line per transfer field. The form load event causes the height and width of the form to be set to the same size as that of the data entry windows. A string is built containing a line of formatted data from the TranRec array. If the If Null Ignore Line flag is "True" a "Y" will be added to the string, otherwise an "N" is added. The value of the If Not All Filled Ignore Line flag causes the string to be appended similarly. The string contains the field values in the same order as they are listed in the record definition. The Tab character is inserted between fields to assure a columnar format regardless of font. The AddItem list box method is used to add a line to the display. The entire definition is processed in this way. Before the show command is invoked the list box index is set to the current window, or the first item in the list if there is no valid value for the current window. Selecting Done closes all data entry forms and the list form. Choosing Editor closes all open entry forms and reopens the entry forms up to the number of the index of the line number that the cursor was on when Editor was selected.

ParmForm is a window with two check boxes, Stop Transfer On Error and Overwrite Fields If Multiply Defined, and two command buttons, Ok and Cancel. The check box values are saved to the StopOnTransferError and OverWriteFields variables, respectively, should the user choose Ok.

The Viewer form is a file viewer with only a minimize and maximize button in the frame and

12
one command, *Done*. The form load event calls the common dialog control to get the name of the file to view. If an existing file name is specified the user is presented a message box displaying the file name again. Choosing *Ok* causes another message box to appear asking if the file's contents should be printed as well as displayed. The clear method clears the list box, the font is changed to one of fixed-width characters, and a non-empty line of the file, trimmed of trailing blanks, is added to the list. This process is repeated until five hundred lines have been added to the scroll or the end of the file is reached. The file's contents are optionally printed. If at least one item has been added to the list then the Viewer's show method is invoked, otherwise an error message box is displayed telling the user that the file is empty.

The module Main contains the program declarations discussed in the first part of the procedure section. Main also contains all procedures and functions not directly activated by events. The CheckZero function returns a string from an integer if it is passed. If the integer is zero the empty string is returned; otherwise, the built-in Visual Basic function StrS returns the string value of the integer. The GetTransFile routine gets a transfer file name from the user by means of the common dialog and message box routines. The routine is passed one parameter, OkRetryCancel, which can be set to one of the ID global constants (IDOK, IDRETRY, or IDCANCEL) used for message box interaction. If a valid filename is given, the procedure will load the file's contents into the Tran_Rec array. Another routine, SaveDef, invokes the common dialog control to allow the user to specify a file name to use to save the current contents of the Tran_Rec array.

The remaining procedures handle the data entry forms and the Tran_Rec array they change. The procedure ClearArray initializes the fields in the Tran_Rec array, from one to the value of MaxRecords, to zeros. The subroutine CloseOpenWins looks at the values in the OpenWins array and calls the CloseWin procedure for those values that are "True." The index of the open form is passed to CloseWin. CloseWin uses a case statement to check the value of the integer it is passed and unloads the data entry form number (1 to MaxRecords) that corresponds to that integer. Next, CloseWin sets that indexed value in the OpenWins array to "False" and the CurrentWindow variable to the integer corresponding to the value of the index minus one. The InitOpenWins procedure initializes the OpenWins array variables to "False." The last procedure to be discussed that handles window opening and closing is OpenWin. OpenWin is passed a single parameter, the index of the window to be opened. The Load command is executed with the parameter of the form number corresponding to the index of the parameter passed to OpenWin.

A FormLoad procedure is called from the form load event procedure in the data entry forms. FormLoad is passed an integer representing the index of the form number to load, another integer representing whether the user is in edit mode, and a third integer describing whether to invoke the show method on that form. The top and left properties are set for the next form's placement and the OpenWin's value indexed by the first parameter passed to FormLoad is set to "True." A case statement sets the correct form properties and makes sure that the Tran_Rec array element indexed by the first parameter passed to FormLoad is used to initialize the forms variables. Finally, if the third parameter passed to FormLoad is true, the show method is invoked on the form. An UnloadForm procedure for the entry forms is also found in Main. This procedure's only purpose is to decrement TopField and LeftField to ensure that the entry form's cascade effect works.
The final two procedures to be discussed are ReadNArr and StoreArr. The StoreArr routine transfers one data record from the currently active form to the corresponding Tran_Record array index. ReadNArr, designed to do the reverse, is not currently used.
RESULTS AND CONCLUSIONS

The project resulted in a fully functional data transfer program that enables the user to create transfer definitions and execute these definitions. The definitions work on ASCII text files. The user interface is intuitive, and users will find the program to be an aid to the organization, as well as to the definition, of their data transfer needs. Pull-down menus allow for easy selection of menu options. A user's guide, reference, and on-line help provide aids in the creation and transformation process.

This student found that as the project was begun, in 1992, using Borland's Turbo-Vision in an MS-DOS environment, it seemed that the next release of Turbo-Vision would answer the difficulties encountered in the tools used. This next release never materialized as Borland embraced Object-Vision and other Microsoft-Windows based products, dropping or curtailing all work on DOS products. Refusing to embrace MS-Windows and consequently, Object-Vision, the student was unable to find any DOS product meeting his goals of a windowing environment and a smooth upgrade path for future versions. Finally, at the project committee chair's suggestion, he turned to Visual Basic for DOS. It then became apparent that the problems that he was still facing stemmed not from a product but from an environment. Visual Basic for MS-Windows provided the window frame controls and memory management needed to develop DataTrans, without the probability of premature obsolescence.

Improvements are necessary in screen resolution detection and window size placement based on that resolution. A textual format for the saved transfer definitions would allow for their modification using any editor or word processor. A macro capability would allow for reference to form numbers without the awkward case statement mechanism. The data entry forms can probably be treated as a control array. Inability to use the control array to display more than one form at a time in previous versions required that this less preferred method be used. The current version of Visual Basic for Windows (3.0) may have the enhancements required for its implementation. Inclusion of binary field data as input should be explored.

DataTrans will not convert application-proprietary data formats. In other words, converting a Word Perfect document to a Microsoft Write format should be left up to other software solution providers, and DataTrans should concentrate on the mainframe and report conversion market where manipulation of textual files is a primary consideration. The ability to read files with variant records, like COBOL data files, would be a useful enhancement. This could be done by providing much more extensive input data checking options. In addition, the utility could be enhanced to allow for the definition of a specific group of expected values to be found in one of the input data fields and the output format of the data to be used for each of these values. This would be useful, for example, when trying to extract COBOL data with varying records by record type.
This enhanced basic transfer definition would make the basic building block for the new transfer definition. The new transfer definition would incorporate multiple input and output descriptions. Users would be able to define transfers that would draw from any number of inputs and write to any number of outputs. The user interface can be enhanced by providing a hot key to the viewer and adding a ruler to the viewing edges for column and row placement identification. Drag and drop capabilities, then, would give the user the graphical ability to define a transfer. An OLE interface to Microsoft Word and Excel would provide access to highly developed software tools. Excel would provide a table-like browse as well as storage and editing mechanisms for transfer parameters. Word can provide a customized window with a ruler where clicking the mouse on the correct column would cause the column number to be saved to a transfer description. The Tran_Rec_Type will need to evolve into a class to contain the more complex transfer descriptions that will be built and provide methods and attributes for the OLE interface.

**DataTrans** ease of use could be enhanced by adding the ability to interface with data dictionary information from a number of sources. The transfer definition would already be available in many cases. This would greatly magnify the storage and retrieval capabilities of **DataTrans**. **DataTrans** can be further extended to keep specific information in the form of descriptions and notes about a definition.
BIBLIOGRAPHY


# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before You Begin</td>
<td>1</td>
</tr>
<tr>
<td>Introduction</td>
<td>2</td>
</tr>
<tr>
<td>Hardware and Software Requirements</td>
<td>3</td>
</tr>
<tr>
<td>Installation Guide</td>
<td>4</td>
</tr>
<tr>
<td>Essentials</td>
<td>5</td>
</tr>
<tr>
<td>Tutorial</td>
<td>7</td>
</tr>
</tbody>
</table>
BEFORE YOU BEGIN

On-Line documentation is available in DataTrans by means of a context-sensitive reference guide. Pressing the F1 key will bring up the help topic most closely related to your current activity in the program.

In the MS-Windows environment an item such as an entry box, a command, or a submenu selection may be selected by clicking it once with the mouse or by using the Tab key, Shift-Tab combination, or arrow keys to move the cursor to the selection. To open a menu, simply click on it and a list of submenu items is shown. Move the mouse pointer to the item you want to select on the submenu and then click on that item. You may also move directly to a command on a menu by typing the underlined letter in the command or subcommand. The Alt key will help you focus on the menu bar and command selections if you do not have a mouse. The entry boxes prompt you for files to save data to, retrieve data from, or delete. All have a look and feel similar to those that MS-Windows uses for all file functions, including those of file manager. The DataTrans application is made of forms that have frame structures with functions similar to those discussed in the MS-Windows User's Guide.

The typographical conventions used in the document are as follows:

<table>
<thead>
<tr>
<th>Formatting convention</th>
<th>Type of information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bold</strong> type</td>
<td>menu selections, program name</td>
</tr>
<tr>
<td><em>Italic</em> type</td>
<td>command button captions</td>
</tr>
<tr>
<td><strong>Monospaced font</strong></td>
<td>list, check box, button labels,</td>
</tr>
<tr>
<td></td>
<td>textual data</td>
</tr>
</tbody>
</table>

The Introduction gives a brief overview of the philosophy and utility behind DataTrans.

The Hardware and Software Requirements section gives the user a feel for the necessary prerequisites to using DataTrans.

The Installation section gives the user specific directions for loading DataTrans into the directory structure on the hard drive, as well as instructions for starting DataTrans.

The Essentials section provides the user with an overview of the transfer creation and execution process.

The Tutorial takes the user through a basic data transfer creation and execution example, providing helpful hints along the way.
INTRODUCTION

**DataTrans** is a utility program that facilitates the creation, storage, and execution of data translation definitions. Operating in an MS-Windows environment, **DataTrans** uses the latest programming tools available to provide an easy to use graphical user interface. The current version of the program allows for the creation of one output file from one input file based on a one-to-one correspondence between input and output fields.
HARDWARE AND SOFTWARE REQUIREMENTS

DataTrans will require the MS-Windows operating environment, version 3.1 or later. An MS-Windows compatible display adapter is required and an MS-Windows compatible mouse is highly recommended. The display adapter and monitor must be capable of 640 x 480 resolution or greater in SVGA mode. 800 x 600 resolution or greater must be used for the DataTrans data entry forms to cascade properly. DataTrans also requires that the common dialog DLL be in the Windows system directory. DataTrans is meant to be operated on an 80386 or higher series Intel processor or the equivalent. A minimum of four megabytes of RAM is suggested. DataTrans can not be used from a floppy drive. A minimum of two megabytes of free hard disk space is suggested before attempting installation.
INSTALLATION

Place the diskette marked "disk 1 - type a:install" in your 3-1/2" diskette drive and type a:install from the Dos prompt. The program will install into the directory "datatran" on the hard drive you specify. An icon can then be created following the directions in the Microsoft-Windows operating manual, or the Program Manager File Run option may be used to run the program by typing \datatran\datatran from that prompt. DataTrans may be run from the Dos prompt, as well, by changing to the datatran directory and typing "datatran."
ESSENTIALS

This section of the user's guide contains a brief overview of the functions of DataTrans. For a more detailed description of the menu choices and their behavior, refer to the Reference Guide.

DataTrans copies selected records from an input to an output file, reformattting each record according to field locations you specify. Reformattting is described in terms of fixed-length fields. A field is a datum, like social security number, or address. DataTrans expects a text file as input and produces a text file as output. You create a definition describing how the input looks and how you want the output to look. Column numbering begins with one. When a transfer definition is saved any field definitions containing a column location of zero will be eliminated. DataTrans has not saved your transfer definition until you assign it a file name. A file name is prompted for when you choose Done in the Create, Edit, or List submenu selections forms.

The Define section of the menu groups together functions that have to do with creating an initial data transformation description, editing an existing description, storing a new or corrected description, deleting a description, or loading a description for change. A shortcut to description editing is to use List. Using the List selection allows you to access a scrolling list box, where each line of the transformation description is displayed in one line of the box. Simply go to the line you wish to edit and click on or otherwise activate Editor, and the data entry form corresponding to the field that needs correcting appears. The New and Edit selections allow you to input into one form, go to the next or previous form, or jump a number of forms by using List. Choosing List from New or from Edit brings you to the same form as choosing List from the menu.

The Transfer section of the menu contains a grouping of the functions needed for DataTrans to carry out an actual transformation of input data into output data using a transformation description. The transfer definition file may be specified in the Recall Definition submenu selection. Selecting Parameters allows you to specify if data that has already been transferred is to be overwritten and whether or not you wish the program to suspend processing should this overwrite situation occur. The input, output, error log, rejected records, and data transfer files may be specified in the Data Files section. Most importantly, Go will perform the actual transfer of data. If an input, output, or data transfer file has not been specified and you choose Go, you will be prompted for the file names you want to use.

The View section of the menu contains only one item; there are no submenu selections. You are prompted for the name of the file you want to view. Supply the name of an existing file, and you are asked if you want a printout. You are presented with a scrolling list box where the contents of the file (limited to 500 lines) is displayed. Caution: Always use a text file to print or display, or the results of the printer and/or file viewer are unpredictable!
Exit is selected when your work is completed or you simply wish to leave DataTrans. You will be prompted for a file name where your work may be saved if you have forgotten to save it.

Context-sensitive Help is available by pressing the F1 key. The help available on-line is identical to the text contained in the Reference Guide. Deleting a field definition is accomplished by placing a zero in any column location and saving the file. If the definition is used in a transfer then any column location of zero causes the field to be ignored.
TUTORIAL

This section contains an informally presented guide leading you through your first **DataTrans** application. When you start **DataTrans**, you are presented with a menu bar offering three possible choices. Selecting the first item on this menu bar, **Define**, you are presented with a submenu from which you may choose **Create**, **Edit**, **Store**, **Delete**, **List**, or **Retrieve**. Choosing **New** takes you to a form for entering the first field of data transformation information. Transformation information may be entered in any order, although for the sake of approaching the task with some degree of order, we will begin from left to right with the input file.

The example uses the sample input file "Schedule.txt." The first five lines of the file are listed below:

```
000001ACCT13711 SS1994 6199407081994081219940713 0.00
00.00MTWTHF 800A 935A 480 575C T105 0.00 0.00F1D
  235 0 0 01FFNF

000002ACCT24022,3 SS1994 6199407081994081219940713
24.00MTWTHF 945A 105P 585 785C T222 0.00 0.00F2D
  235 20 10 101FFNF

000003BIOL14093 SS1994 6199407081994081219940713
15.00MTWTHF1130A 105P 690 785C S101 0.00 0.00F2D
  235 30 22 221FFYF

000004BIOL1409A SS1994 6199407081994081219940713
15.00MW 115P 400P 795 960C S108 0.00 0.00F2D
  235 30 22 221FFLF

000005COSC12732 SS1994 6199407081994081219940713
24.00MTWTHF 945A1120A 585 680C T213 0.00 0.00F2D
  235 20 8 81FFNF
```

and we would like the output to look like:

```
001 ACCT 1371 1 MTWTHF 800A 935A TO
```

Counting the character locations begins with one. The data contained in both the beginning and ending column locations is included. Therefore, the first field input beginning column is **4** and the ending column is **6**. The output beginning column is **1** and the ending column is **3**. Type these values in the first form. The Tab key will take you to the next field; shift tab to the previous one. Alternatively, the mouse may be moved and clicked to change the focus to any object on the form, or a command box may be selected by pressing the Alt key in combination with the first letter of the command label. If the field contains any blanks, we do not wish the entire line to be transferred to the output file; so mark the **If Not All Filled Ignore Line check box** and choose **Next**. This is the second data entry form where you will enter transfer information for the second transfer field. The input beginning column is **7** and the
The output beginning column is 6 and the ending column is 9. Type these values in the second form and click on Next (or press enter or use the Alt-n keystroke sequence). Continuing in this manner, enter 11 for the input beginning column and 14 for the ending column, 11 for the output beginning column and 14 for the ending column. Click on Next and form 4 appears. Enter 15 for the input beginning and ending column and 16 for the output beginning and ending columns. Proceed to the next entry form. In form 5 enter 69 as the input beginning column and 74 as the ending column. 23 will be the output beginning column and 28 the ending column. Proceed to the next entry form. Notice that the 800A sequence of characters begins with column 76. If you have a data layout available for the input, this will simplify locating the fields; otherwise some trial and error will probably be necessary. Since 1200A is a possible time you should enter 75 for the input beginning column. Pretend you did not notice this issue and enter 76. Enter an input ending column of 78 rather than the correct 79. This mistake will cause the exclusion of the trailing A or P. You will correct these mistakes later.

The output beginning column is 30 and the ending column 33. Make the same mistakes for the time field locations in the next form (7) by choosing input beginning and ending columns of 80 and 83. The output beginning and ending columns will be 36 and 39. Proceeding to form 8, the input beginning column is 98 and the ending column 102. The output beginning column is 41 and the ending column 45. Click on Done and you are presented with a dialog box asking for a file name under which the definition may be saved. Type in "sample.def" and click Ok. You are presented once again with the main menu. Choose Transfer. The choices are Recall Definition, Parameters, Data Files, and Go. Choose Go. You are asked for the input file name, the output file name, and the transfer definition file name (which already has "sample.def" as the selection) -- just choose Ok. Type "Schedule.txt" for the input file name and "Schedule.out" for the output file name. DataTrans displays a message box containing the names of the files you have chosen and asking if you want to continue. Click Ok and the transfer process begins. Upon completion you are presented with a message box telling you again where to find the data files and that the transfer operation was successful. Clicking Ok here returns you once again to the main menu. Choose View and you are prompted for the name of the file to view. Type in "Schedule.out" and a window appears with a scrolling list box containing the output data. The first five lines should appear as follows:

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>ACCT</td>
<td>1371</td>
<td>1</td>
<td>MTWTHF</td>
<td>800</td>
</tr>
<tr>
<td>002</td>
<td>ACCT</td>
<td>2402</td>
<td>2</td>
<td>MTWTHF</td>
<td>945</td>
</tr>
<tr>
<td>003</td>
<td>BIOL</td>
<td>1409</td>
<td>3</td>
<td>MTWTHF</td>
<td>130</td>
</tr>
<tr>
<td>004</td>
<td>BIOL</td>
<td>1409</td>
<td>A</td>
<td>MW</td>
<td>115</td>
</tr>
<tr>
<td>005</td>
<td>COSC</td>
<td>1273</td>
<td>2</td>
<td>MTWTHF</td>
<td>945</td>
</tr>
</tbody>
</table>

Notice that the comma and three (,) are left off the course section number for ACCT 2402 and that the A or P are missing off the times. Go to Define, then select Edit. Now, prompted for a filename, enter "sample.def" and the previously entered information will be available when you return to the data entry or list form. The first entry form, seen earlier in Create, appears. The entry form heading now contains the word "Edit" instead of the word "Create." You need to change the fourth form so press Next until form four appears. Enter 15 in the input beginning column, 20 in the ending column, 16 in the output beginning column and 21 in the output ending column. This adds an extra 4 characters to the size of the section number and should take care of any data we missed on the first run in this field. Choose Done and you are prompted for a file
name. Choose Ok since the name of the file we want to use, "sample.def", is already in the prompt. You are returned once again to the main menu. Ending columns on the sixth and seventh fields, however, still need changing. This time choose List from the Define submenu. The now familiar background window appears, but this time, instead of an entry form there is a columnar formatted list of the transfer fields, one field per line. Scrolling down to the sixth line and choosing Edit will take you into the sixth form where the ending input and output columns are to be increased by one. Next takes you to the seventh field editing form where the input and output columns should also be incremented by one. Choose Done and then Ok to save the file. Returned to the main menu you choose Transfer, then Go. The file names previously used were retained, and you are asked if you want to continue using them. Choose Yes and transfer begins. The success message appears once again. Acknowledging the message leads you back to the main menu. Viewing the data as before you see:

```
001  ACCT  1371  1     MTWTHF  800A  935A  T105
002  ACCT  2402  2, 3  MTWTHF  945A  105P  T222
003  BIOL  1409  3     MTWTHF  130A  105P  S101
004  BIOL  1409  A     MW      115P  400P  S108
005  COSC  1273  2     MTWTHF  945A  120A  T213
```

This time the ,3 appears on ACCT 2402 and the times reflect whether they are morning or afternoon classes. There is one remaining problem: the class meeting times are not correct if the class meets or lets out between 10:00 and 12:00. This was mentioned before, and you should be able to make the correction as an exercise. (See the hints in the first transfer creation) When you are finished the output should be:

```
001  ACCT  1371  1     MTWTHF  800A  935A  T101
002  ACCT  2402  2, 3  MTWTHF  945A  105P  T222
003  BIOL  1409  3     MTWTHF  1130A 105P  S101
004  BIOL  1409  A     MW      115P  400P  S108
005  COSC  1273  2     MTWTHF  945A  1120A T213
```

Deleting a field definition is accomplished by placing a zero in any column location and saving the file. If the definition is used in a transfer then any column location of zero causes the field to be ignored.
REFERENCE GUIDE

This section is organized in a fashion similar to that of the menus in DataTrans, rather than alphabetically. The contextual help function brings up this reference guide. The information appears here in the following order:

DataTrans

Define
  Create
  Edit
Store
Delete
List
Retrieve

Transfer
  Recall Definition
Parameters
Data Files
Go

View

Exit

Contextual Help
**Data Trans** - A utility program that facilitates the definition, creation, and viewing of file transformations from one textual columnar format to another.

**Define** - A menu selection allowing a user to access a submenu containing selections for the creation, editing, and management of transfer definitions.

**New** - In this submenu selection a user is presented with one data entry form for the definition of one input and one output field. The form contains four boxes for entering the input beginning and ending columns and the output beginning and ending columns. There are also two check boxes. The check boxes cause the data transfer process to discard the entire input line if this field contains any blanks and the If Not All Filled Ignore Line check box is marked. If the field is all blank and the If Null Ignore Line check box is marked the line will also be discarded as possible output. Once these entries have been completed, one of four commands may be selected. Next takes a user into an almost identical entry form for the definition of the next transfer field. If they are on the form corresponding to the maximum number of possible field definitions (twenty in this version), a beep will sound. Previous allows the user to change or recheck the previously defined transfer field. If they are on the first field definition, a beep sounds. If List is chosen, they are taken from the one field per entry form format into a list box where the format is one field per line. Scrolling through this list, highlighting an incorrect entry and choosing Edit provides a convenient method for checking and correcting entries. Choosing Done closes all open entry forms and prompts the user for a file name under which the transfer definition may be saved. If a transfer definition file has already been chosen during this DataTrans session, this selection will be present as the default file specification. In any case, a user may specify either a new or existing file name for storage of the transfer data on disk.

**Edit** - This submenu selection is similar to the New submenu function with the exception that a user is first prompted for the name of the file containing the transfer records that are to be changed. A user may change the contents of the forms and store the new transfer descriptions to the same or a new file name. If there are few changes to be made they may consider using the List submenu command instead.

**Store** - This choice allows the user to save a transfer definition. They are prompted for a file name and the information that last appeared in the data entry forms or list form is saved to disk under that file name.

**Delete** - This submenu selection prompts the user for the name of a file to delete. If an existing file is chosen, they are asked to verify that they wish to erase the file before the file is deleted.
List - This submenu selection is similar to the New selection. Here, however, the user is first presented with a one line per transfer field scrolling list box where they can scroll to a line that needs to be checked. Should correction be necessary they may choose the Edit command in order to move into the create/edit form where field entries may be changed. Choosing Done from the data entry forms or the list form returns the user to the menu. They are prompted for a file name under which the changes may be saved on disk.

Retrieve - With this selection the user is able to specify the name of the data transfer description file that they wish to review or correct. Any future file prompts for transfer description file names will use this name first, although the user still has the option of choosing a different file.

Transfer - A menu selection allowing users to access a submenu where actual data transfers can be managed and performed.

Recall Definition - This selection allows users to specify the name of the file containing the transfer definition that they want to use for the transfer process. It is similar to the Retrieve function of the Define menu.

Parameters - With this selection a user can specify whether the transfer operation should overwrite existing output and if the transfer operation should stop or continue when data is going to be overwritten. A form containing two check boxes and two commands is presented. Stop Transfer On Error is checked if they wish to stop when output data is going to be overwritten. Overwrite fields if multiply defined is checked if they wish to overwrite existing output data. In other words, if the transfer process has already output a group of characters, say "Over" to columns 3 through 6, then trying to "Write" to columns 3 through 7 in a later field definition will cause this overwrite error. If the overwrite condition should not occur then checking Stop Transfer On Error will prevent it.

Data Files - This submenu selection allows user-specification of file names for the input, output, data transfer, error log, and rejected records file. These files will be used as default values by other DataTrans functions until the user changes them. If file names are not specified here you will be prompted for them in the Go selection. However, the default error log (DataTran.Log) and the default rejected records (DataTran.Err) file names may only be changed here.

Go - This submenu selection performs a data transfer. If file names have already been given for input, output, and data transfer data, the user is asked if they wish to continue using these names. If the answer is "yes" the transfer process will begin. If not, the user will be asked to supply names for the input, output, and transfer data files. The user is then asked if they want to begin
the transfer process. A "yes" will begin the transfer process. Upon completion of the transfer
the user is presented with a message box signifying the success or failure of the transfer process.
A listing of data that was overwritten and the overwriting data as well as the total number of
lines transferred to the output file will be contained in the error log. The rejected records file
will contain lines rejected due to the user specifying, in the transfer definition, that fields not
meeting the If Not All Filled Ignore Line or If Null Ignore Line criteria
should cause that line of data not to be transferred to the output file.

View - This menu selection allows the user to view input data, output data, error logs, or
rejected records on the screen. The user will be asked if they wish to print the contents of the
file as well. A scrollable list box allows the user to view the file's contents. A maximum of
five hundred lines of text will appear on the screen although the file's entire contents will be
printed. When the user has finished viewing the file, Done restores the main menu.

Exit - This menu selection will prompt the user for a file name under which to save their work
to disk if it has not already been saved. All files are closed and the program is ended.