SESSION II

WRITING INSTRUCTIONAL AND RESEARCH SOFTWARE

Thomas B. Perera, *Chair*

Writing and publishing instructional and research software

THOMAS B. PERERA

Montclair State College, Upper Montclair, New Jersey

Many psychologists who currently use computers in research and instruction will eventually consider developing software of their own. Recommendations for structuring, writing, documenting, and possibly publishing software, as well as dealing with publishers, modifying programs, and converting syntax, are presented in hopes of stimulating the writing of new software.

Psychologists who use computers for instruction or research often find that they need to modify the software or that they need to perform tasks for which no software is currently available. Many decide to try their hand at writing or contracting for the writing of customized software to suit their individual requirements. The decision to develop software programs is more complex than may be apparent. A number of different considerations are addressed in this paper.

WRITING SOFTWARE VERSUS HAVING IT WRITTEN FOR YOU

The simplest way to obtain custom software is to have it written for you by an experienced programmer. It saves time and energy and may produce a finished product more rapidly than if you have to learn to write code yourself. The drawbacks are that it is virtually impossible to communicate all of your specialized requirements to a programmer in such a way as to allow the production of satisfactory software on the first pass. This means that numerous frustrating revisions are usually necessary before the program becomes usable. These strain budgets and relationships with programmers and may lead to a total breakdown of communications at a time when critical final program revisions are needed. The trade-off between taking the time to learn programming and make the ongoing revisions yourself and establishing and maintaining a working relationship with a programmer is the first major decision that has to be made.

WHO WILL USE YOUR PROGRAMS?

Careful consideration of the requirements of the users of your programs helps dictate most of their structure. If the program is to be used only by your research laboratory, it can be a bare-bones dedicated process controller with little or no explanatory screen display and minimal documentation. Surprisingly, however, many programs that start out this way become popular among other psychologists who are doing similar work. A few extra hours spent in organizing and documenting the structure of a program pays major dividends when it comes to allowing other users to understand and even modify a program for their own needs. It also helps retain the usefulness of the program in your own laboratory. I have stared in utter bewilderment at programs that I had written and fully understood but failed to document several years before. I find that it takes almost as much time to reestablish a working understanding of an undocumented program after a few years as it did to write it.

In the days when memory was expensive, it was impossible to include lengthy remarks that explained every line of code within the program itself. With the price of memory falling rapidly, it is now feasible to comment every line of a program and to block off groups of lines into visually identifiable sections that make reading a program listing very easy.

WRITING PROGRAMS FOR OTHERS TO USE

You may set out to create a simple program for your own use and find that it is in demand by others in your field, or you may decide initially to write programs that are designed to be distributed to others. As soon as you

The author's mailing address is: Montclair State College, Upper Montclair, NJ 07043.

reach the decision to write programs for other users, you take on the responsibility of ensuring that the programs are comprehensible, well documented, and as free of bugs as you can possibly make them.

Preparing programs and documentation for distribution may take at least as much time and effort as writing the code. Ideally, the documentation should include comments within the program itself to explain the operation of each line. In addition, a booklet should be written that includes a brief historical introduction to the topics addressed by the program, a detailed description of the workings of the program, suggested ways to modify the program to meet specific requirements of users, and a complete printed listing of the program code with a list identifying every variable used.

DISTRIBUTING YOUR SOFTWARE

The decision regarding how to distribute your software is also a complicated one. If you decide to place it in the public domain by publishing it in a journal or distributing it free of charge, your career may benefit, but you may have to deal with dozens or even hundreds of users who will contact you for more information, for modifications of the programs, or for all kinds of hand-holding assistance.

If your employer is one of the growing number of institutions that value computer programs as scholarly publications, the inconvenience of having to deal with so many end-users and the lack of financial gain may be outweighed by the possibility of obtaining further promotion, tenure, or advancement in your job. However, a growing number of tenure and promotion committees are requiring evidence of peer review of software submitted as part of an applicant's credentials. In some cases, commercial publication with its inherent review processes may actually enhance an applicant's vita.

If you decide to distribute your software for sale through a publisher, you will have to assess its impact on your career advancement, but you will be shielded from end-users by the publisher and you probably will realize a significant financial gain.

To calculate the probable financial gain, consider that there are only about 2,000 4-year colleges and an additional 700 junior colleges and community colleges in the United States. If your publisher were to sell one of your programs for \$30 and you were able to talk your publisher into a royalty as high as 20%, you would realize a profit of \$6 for each program you sold. If your publisher were aggressive enough to be able to sell a single copy to one third of the colleges, your earnings would be \$5,400. These figures should allow you to calculate expected earnings based on an estimate of the selling price and the percentage of the colleges you expect to purchase your software.

There are a number of ways to increase software sales, which may make your software more attractive to potential publishers. Obviously, if you could sell multiple copies to the colleges or arrange for each student to purchase

the program disk at a reduced price, you and your publisher could make a significant amount of money. Perhaps arranging for programs to be distributed on floppy disks that would record the experimental data over the original program and then be handed in as completed assignments or would become unusable after the experiment was completed would increase the profit margin for authors and publishers and therefore increase the amount of available software. An increasing number of college textbooks include software as part of their overall packages, and a cooperative arrangement with a textbook publisher can open up an extremely large market.

There are also a significant number of nontraditional markets for academically oriented software. Twenty-five thousand high schools, 6,500 hospitals and rehabilitation facilities, and all of the public and private research institutions may be interested in psychological or statistical software. To access these markets, however, a publisher has to have a very wide advertising base and budget, and few publishers reach out to all of these areas.

PUBLISHING YOUR SOFTWARE

Dealing with a publisher can be a very complicated undertaking. You have to convince the publisher that you have something worth publishing. This often requires demonstrating your software and obtaining letters from colleagues who testify to its value and salability, as well as developing a realistic estimate of potential sales. The potential profitability of your software will probably be the main, if not the only, concern of a commercial publisher whose existence depends on profit.

After a publisher shows interest, you must bargain with the publisher for a contract in which you receive the highest possible royalties for all conceivable forms of domestic and foreign distribution. You must also retain ownership of the copyright on your software in case the publisher should go out of business. You must also obtain guarantees regarding publication and distribution with contingency clauses that return all rights to you if the publisher should cease publication of your materials for any reason. Finally, you must read every word of the contract to ensure that you are not giving up some right to your present or even future software.

After you obtain a contract, you invariably will be given a deadline for the completion of the finished program. This deadline also applies to the completion of the written documentation that is to accompany your program. The publisher will put a great deal of pressure on you to complete your program on time so that advertising deadlines can be met. After completion, your program will be sent out for review and your publisher will require you to modify your program in accordance with the comments of the reviewers. You may have to argue extremely forcefully to override the comments of an unreasonable reviewer.

Once your programs are published, your publisher will generally shield you from the trivial comments of most end-users and you might think that all you have to do is

deposit your royalty checks. Unfortunately, there is another problem to consider. The computer field is changing so rapidly that your programs, if written for one specific brand of computer, will become obsolete as rapidly as will the computer for which you wrote them. It may be only a matter of 2 years before you are faced with the difficult decision of whether to convert your programs to run on the latest computer systems or to abandon them as commercially salable packages. The task of converting a program to run on another computer system can be extremely tricky and tedious if the program makes extensive use of input/output calls, peeks and pokes in memory locations, substantially different program timing, or any kind of graphics. But, as is the case with introductory psychology textbooks, your programs must be frequently revised and converted to new syntax if they are to remain usable.

If you want more information regarding publishing your software, you can obtain an author's kit from Life Science Associates, 1 Fenimore Road, Bayport, New York 11705 (516-472-2111).

SHOULD YOU COPY PROTECT YOUR PROGRAMS?

Another decision involves whether to protect your program from being copied by using the various copy-protection schemes or whether to distribute it without copy protection. Since the sophistication of copy-protection-breaking software has almost caught up with the sophistication of copy-protection schemes, the value of copy protection is minimal. In addition, I have found that writing my software (Perera, 1979) in BASIC, providing listings with all programs, and allowing them to be copied freely, has made it possible for an amazing variety of graduate students and other researchers to adapt my software to their own specific projects.

COPYRIGHTING YOUR PROGRAMS

Whether you decide to place the program in the public domain or publish the program individually or through a publisher, I strongly recommend that you make the second line in the program a copyright notice. The statement "Copyright 1987," followed by your name, establishes you as the creator of the program. It is also a simple matter to formally apply for a copyright by requesting Form TX from the Copyright Office of the Library of Congress, Washington, DC 20559. After completing the form, send \$10 and 1 or 2 copies of the program to the copyright office to formally register your copyright. I recommend registering your copyright because the computer field is changing so rapidly that your programs may become extremely important either financially or with regard to your professional career before you realize it. Without a copyright notice, anyone can claim to have writ-

ten your program and can benefit professionally or financially from your work.

SELECTING A PROGRAMMING LANGUAGE

One of the many decisions that must be made is what language to use. If you have already started programming, you have chosen a language with which you are comfortable, and the following comments are superfluous. If you use a simple high-level language, such as BASIC, it is easy to write your programs and easy for others to understand and modify them. The two drawbacks to programming in BASIC are that it is slow and that it is difficult to copy protect if you plan to protect and sell your programs. Both of these objections can be partially overcome by using a BASIC compiler, which converts the BASIC code into machine language and speeds up program execution by as much as a factor of 10 while making the code virtually undecipherable. For applications that require extremely high speed, it may be necessary to program in assembly language. This language is difficult to learn, but you may find that you can hire an experienced programmer to write small assembly language subroutines that can be called by the higher-level language program.

RECOMMENDATIONS FOR STRUCTURING PROGRAMS

The following recommendations apply broadly to most programs that are written in a high-level language to perform research or instructional procedures. I recommend that all programs be broken into clearly identifiable blocks of 100 or preferably 1,000 line numbers and that these blocks or sections be consistently organized in a manner similar to the following sections, regardless of the function of the program. An advantage of blocking programs in this way is to allow you to use some of the same blocks as subroutines in many of your programs since they will probably be performing quite similar tasks.

For psychologists who are just beginning to learn how to program, Richard Deni (1986) has published an introductory book on writing microcomputer programs in BASIC, in which he presents most of the important programming techniques and includes many relevant examples for IBM, Apple, and Radio Shack microcomputers. Unfortunately, most of his timing and input/output routines refer to only one specific hardware interface.

Section 1: Program Identification and Copyright Notice

The first block or section contains line numbers 1-99 or 1-999. It should contain the date of the latest revision, the name of the program, and the copyright notice by the author. Since programs are often modified dozens of times, it is difficult to keep track of which revision of the

program you are working on. If the date of the latest revision is the first line, it will be the first thing you see when you list and edit the program and will be in the easiest position to be updated.

Section 2: Dimensioning and Identifying Variables

The second section, consisting of line numbers from 100-199 or 1000-1999, should contain dimensioning and initializing statements for variables and arrays. It should initialize randomization, if required, and read data into arrays. This section is also an ideal place to include remark statements that identify each variable to be used in the program and how it will be used.

Section 3: Screen Presentation of Program Identification

This section displays the name of the program on the video screen. It may also include the name(s) of the author(s) and perhaps a menu of major choices of experiment types or procedures.

Section 4: Menu Presentation of Experimental Parameters

This section should be used to present a screen display of all relevant experimental parameters so that they may be changed prior to the beginning of the experiment. Parameters, such as the total number of trials, intertrial intervals, and stimulus durations, can be modified while this screen display is active. If it is undesirable for the subject to see this display on the screen, a simple statement such as "GO TO 5000" in the first line of this section will cause the program to skip over this section without displaying the menu.

Section 5: Presentation of Instructions to the Subject

This section can be used to present written instructions to the experimental subject and eliminate possible variability caused by the experimenter reading or reciting the instructions to the subject.

Section 6: Running the Experiment

In this section, the stimuli are presented and the data recorded in an array for future analysis. Time-critical events are tracked using either the internal clock of the computer, a higher level language timing loop, or an assembly language program called by the main program. Software counters keep track of the number of trials and transition into the next section at the end of the specified number of trials.

Section 7: Data Analysis Menu

The data analysis menu allows you to select any of the options described in Sections 8-12.

Section 8: Raw Data Display, or Storage

This section allows you to display data or to store your raw data by transferring it from the variable array in which it is stored in active memory onto the video screen, a floppy or hard disk, or any other permanent storage medium for future analysis. Since most commercial statistical analysis programs have a required format and a specific data set terminator, it is helpful to know what analysis software will be used so that you can dump your data onto disk in the proper format.

Section 9: Perform Specific Data Analysis

This section allows the computer to perform specific analytic procedures on your data and obtain the measures of central tendency, variability, and significance that are appropriate to the experiments being performed. It can be divided into sections that perform parts of the data analysis and allow students to do some of it on their own.

Section 10: Display Graphs of the Data

This section allows the computer to format the data and plot it graphically on the video screen. It also allows interactive modification of the graphs by changing the scaling of variables and axes and replotting the graphs.

Section 11: Print Data, Data Analysis, or Graphs

This section allows the computer to format and send the raw data, the results of the statistical analysis, or the graphic data to the printer.

Section 12: Rerun or End Experiment

This section allows the experimenter to rerun the experiment using the original parameters, to add additional trials or subjects to the experiment, or to terminate the experimental session.

DECIDING WHETHER TO WRITE PROGRAMS OR USE PROGRAMS

I hope that the comments in this paper will be useful in helping you make up your mind whether to write your own programs or to use commercially available software. There is a great need for high-quality software in psychology, and I personally hope that many of you will give software writing a try.

REFERENCES

- DENI, R. (1986). *Programming microcomputers for psychology experiments*. Belmont, CA: Wadsworth.
- PERERA, T. B. (1979). *Computer programs for experimental psychology*. Bayport, NY: Life Science Associates.