PROVIDING ADEQUATE INSTRUCTION TO DATA PROCESSING STUDENTS SPREAD OVER A WIDE GEOGRAPHICAL AREA

Capt. Sheridan J. West II
Lindsey Education Center, A-14
City Colleges of Chicago

I. The Situation: Students are spread out over approximately 20 European U.S. military installations. There are generally not more than two programming classes going at any one location during a single term. Enrollments run from 17-20 students per class. The objective of the course of study is the Data Processing Core and Advanced Certificates awarded after 12 and 24 semester hours of course work, respectively. Students are active duty military personnel and dependents. Courses are offered in the evening, two nights per week, three hours per night, for eight weeks. In each programming course, students are expected to learn the basics of a programming language, program, run, and debug three fairly complex programs before the end of the term. This is complicated by the fact that programs are not assigned until about the third week of the course--leaving only five weeks to run each program. Each program is run an average of three times to complete the problems portion of the course. In most cases, programs have to be sent to a central point for keypunch and compilation. Instruction is provided by U.S. military and civilian personnel qualified in data processing and the language being taught. They are employed on a part-time basis. There are no funds available to pay for data processing services other than tuition, which is considered minimal.

II. Purpose: To give students the fundamentals of programming in several of the most common programming languages in use today. They are not expected to become full fledged programmers. In the introductory courses, an important objective is to provide a general appreciation of data processing and how it is affecting the lives of everyone. In addition, it is hoped that students will apply this understanding so that data processing is seen as a tool to make jobs more meaningful--as opposed to the threat people feel as their job is "automated."

III. Methods: Because data processing resources are limited, the primary teaching method is classroom instruction. In the introductory courses (Introduction to Data Processing and Programming, DP 101 and DP 102) no programming is assigned that is actually run on the computer. All of the tools believed necessary to learn programming languages (logic, flowcharting, number systems, and computer organization) are taught prior to any specific language. Assembler languages and FORTRAN are introduced in DP 102, but only as teaching aids. The programming language courses then concentrate on the language and how it is applied. Students begin writing programs about the third week and are given only nine compilations to complete three programs. Turn-around time ranges from one week for initial runs and three to five days for subsequent runs. It is emphasized that the programs are assigned as learning aids and are to play a secondary role to classroom instruction.

IV. Problems: Two major problems are student program turn-around time and software support.

1. Turn-around time: In the five weeks in which three problems (three runs per problem) are to be completed, it is practically impossible to keypunch, update, and return programs to students fast enough. Partial solutions are:

a. Computer operations have, to a degree, been decentralized. Processing is done in Wiesbaden, Germany; London, England; and Madrid, Spain.

b. A fairly tight suspense system has been enforced requiring students to have programs in by a given date in order for them to be returned in the shortest possible time.

c. Since keypunching is done for students at a central location, many keypunch operators are screened so that only the most accurate keypunching service is provided. This is, of course, subject to keypunch operator availability. Programs are also verified.

d. Tours are arranged so that classes may visit the computer center and run programs on site. This has limited value in that beginning students can, at most, turn around a program twice during each tour. Distance to the central point is also a limiting factor.

e. Instructors are only to assign problems of such difficulty that they cannot be feasibly hand graded. On the other hand, programs are not to be so long and difficult that it would be impossible for students to reasonably complete the assignment. This requires very close supervision of programs assigned, almost to the point of developing standard programs. Standard program assignments are, however, considered an unnecessary restriction on the instructor's use of the computer as a teaching aid.

f. In some cases, students have formed teams to cut down on the number of programs submitted. This is discouraged because the less aggressive team member does not achieve the sense of success required in early mastery of programming skills. Studying and working on other hand-graded projects, as a team, however, is encouraged.

g. Instructors must use classroom instruction as opposed to the computer in teaching the programming language. This is a hard thing for many instructors to adapt to. Many have learned languages when having unlimited access to a machine, and would like to do the same for their students. Because of the turn-around problem, they find this is practically impossible. It is very important that each new instructor be made aware of limitations so he may adopt techniques for teaching the language mainly in the classroom. Students also have to be convinced that a programming language can be understood without extensive practical experience. This is a controversial subject amongst the teaching staff and one which can have a serious affect on a student's satisfaction with the instruction he is receiving. Students who become frustrated with computers in school are very likely to carry this frustration with them in dealing with computers on the job. Slow turn around can adversely affect general attitudes toward data processing; therefore, we make every effort to educate students in why turn around is slow and how they can cooperate in speeding the process.

h. A proposed solution to the turn-around problem is the installation of terminals at each location for remote entry of student programs. This is complicated by the fact that the demand for courses shifts from one location to another precluding permanently installed equipment. Because the student population is so dispersed, terminal would have to be very inexpensive ($100-$200/month) and portable. A terminal which provides limited storage capacity (magnetic tape cassette or a small disk) produces a hard copy of the compiler listing, and is capable of processing simple updates off line to student programs is envisioned. Students would enter their programs through a console or keyboard off line, the programs accumulated on off line storage at the terminal, and periodically the terminal connected to a central facility and all programs sent to the computer for processing. The compiler output would then be transmitted back to the terminal and stored for later off-line listing on the console. This would eliminate keypunching and shift the responsibility for data preparation to the student. The processing would be done in batch as opposed to conversational mode because of the nature of the languages taught and expensive line charges across Europe.

i. An implemented solution is the establishment of a source program library update capability. Student programs are loaded on a source statement library. After the initial run, subsequent runs are initiated by sending update card decks between the classroom and computer. It cuts down on tremendous administrative workloads previously experienced in manually updating decks after program changes are keypunched. Tracing student programs and controlling the number of runs allocated to each class is simplified.

2. Software Support: Most compilers are not designed for student use. Large systems have the now common in core FORTRAN compiler and Fast Assemblers. These student oriented compilers cut data processing costs tremendously. But, they mostly assume a large university data processing center is available. The processing done in the European program is done on several computers, all in the IBM 360 model 25 to 40 range, 24 to 64K. Using standard DOS systems software support, economies of student oriented processing are not achieved. This it seems is a more and more serious problem as data processing becomes a part of junior college curricula. The junior college is more likely to use smaller scale computers or buy excess time from non-educational organizations than to have access to large university computers. Junior colleges are also not in a position to support large efforts to develop student oriented compilers for smaller computer systems. Particularly in the European program, the staff is completely absorbed in teaching and does not have the time or resources to develop and support software for student use.

V. Summary: For junior colleges to effectively implement data processing courses given limited resources and in some cases a very dispersed student body, the problems of program turn-around time and software support for student oriented processing require solution. The solutions are certainly not beyond the state of the art, but may require a collective effort combining the resources of several schools. Solutions that have been developed should be made generally available to all junior colleges. This would allow greater attainment of academic objectives, less tendency to compromise these objectives because data processing facilities are limited, and provide richer experience for citizens learning to cope with an environment supported more and more by computers.