SIGCSE Bulletin, Vol. 12, No. 1, February, 1980. Eleventh ACM SIGCSE Technical Symposium on Computer Science Education, Kansas City, MO, February 14-15, 1980

AN INTRODUCTORY COMPUTER SCIENCE COURSE FOR NON-MAJORS

ROGER L. WAINWRIGHT
COMPUTER SCIENCE DEPARTMENT
THE UNIVERSITY OF TULSA
TULSA, OKLAHOMA 74104

INTRODUCTION

A recent survey among the faculty at The University of Tulsa concerning faculty computer experience, usage, needs, literacy and attitudes was conducted. The results are reported in [4]. Each faculty responding to the survey was classified as a user (those who have had any exposure to computers or made use of any computing facilities not including hand calculators) or nonuser (those with no exposure to computers). The results showed both groups agree most college graduates are computer illiterate and do not know how computers are used in their disciplines. Also both groups recommended their students have more computer literacy than they are currently receiving. The survey results also showed students of the nonuser's faculty rarely received any exposure to computing or computers during their college careers. Furthermore, the nonuser's faculty felt there was no current computer course geared especially for their students. We suspect similiar results can be found in other colleges and universities. Graziano [2] reports the "introduction to computer literacy" course to be the most recent course to appear in the computer science discipline across the country. His survey shows only 28% of the institutions are offering such a course with an additional 16% of all colleges planning to offer such a course in the near future. However, 32% of all colleges predict this course to be a school requirement within the next three to five years.

This paper describes an approach to an introductory computer science course designed especially for students who are not specifically required to take a computer course and thus ordinarily receive no appreciation for computers or computing. This is the third semester this course has been offered. Student enrollment has been 31, 46 and 41 respectively. We anticipate higher enrollment figures next semester as more advisors are becoming aware of the course. In a typical semester students majoring in such disciplines as English, advertising, nursing, psychology, sports administration, sociology, broadcasting and communication, music, elementary education, art and anthropology have enrolled. This course is ideal for those majoring in mathematics education as one day they may be teaching such a course to high school students. To encourage this group of students to enroll in the course, we restricted students from engineering and physical sciences and business disciplines from attending. They are required to take a different computer course. We have observed that most students not required to take a computer course desire to learn something about computers, and because of the above restriction are less hesitant to enroll.

COURSE OBJECTIVES

Mazlack [3] suggests there are three approaches to an introductory computer science course: technical competence, nontechnical awareness, and a mixture of technical competence and nontechnical awareness. This course falls within the mixed awareness category. We already offer two technical competence introductory courses, one for business students and another for engineering and physical science students. Both involve applying FORTRAN 77 to structured programming. The nontechnical (no programming) approach for this course was not considered. The author supports the school of thought that it is impossible to truly develop computer awareness without some level of technical competence. Without programming students fail to develop an awareness of a computer's potential, power and capabilities.

There are two main objectives of the course. One objective is to make students aware of a computer's impact on society by discussing various computer related topics. These topics include the ethical and unethical uses of computers, laws governing computing, advantages and disadvantages of using computers, specific examples of the use of computers in local business and the impact of computers in other areas of society such as government, education and health. This is an important part of the course but not the primary objective as this accounts for approximately 35% of the course. The primary objective is to develop sufficient skills so students can use a computer to solve simple problems. The student should gain some understanding of how a computer works and what a computer is capable of doing effectively and efficiently and what it cannot do. Students should learn that one need not be an engineer or computer scientist in order to solve problems on a computer. A wide variety of programs are assigned in the course not only to develop programming skills but to provide examples of the use of computers in society.

To determine what material is included in this course, we state the following objectives.

1. The student should gain knowledge of the function and structure of an algorithm and the use of flowcharts for depicting simple algorithms.

2. The student should gain knowledge of how a computer works at the low level by writing a simple program in a low level language.

3. The student should gain knowledge of a high level language and be able to construct simple programs in this language.

4. The student should acquire man-machine interface skills in a computing environment including the use of the keypunch and terminals.

5. The student should develop self-confidence and a sense of accomplishment when working with this man-machine interface.

6. The student should gain an understanding and appreciation of information processing concepts needed in todays business world.

7. The student should gain an understanding of the impact of computers in other areas of society such as government, health, education, humanities and gain the understanding to apply this to their discipline.

8. As an introductory course, no prior knowledge of computer science is assumed and no college mathematics is required. First year high school algebra should be sufficient, but not necessary. Students should be able to survive this course with moderate effort and without special help sessions or treatment by the instructor.

BASIC and BILLY-O as Programming Languages

The BILLY-O language is a fictitious assembler-like language developed for educational purposes to show the "inner workings" of a computer. It is an old concept. The author used a similiar BILLY-O-like language as a student some years ago. For a description of BILLY-O including instruction format, instruction set and functional diagram see appendix A. A sample program listing can be found in appendix B. Cook [1] indicates one should avoid assembler language as a topic. He indicates most students in a course such as this "have an almost unbounded disinterest in the 'inner workings' of the computer or in studying programming examples in a low level language." We tend to agree if the assembler used is from a modern computer requiring students to learn about such things as half word, single word and double word instructions, index and base registers, indirect addressing, internal representation of data, complement arithmetic, a large instruction set and so forth. None of the traditional assembler subjects need to be discussed in BILLY-O, however. BILLY-O is a simple language and easy to learn. Students are able to grasp the concepts in two or three lectures. The BILLY-O language is presented at the beginning of the course simultaneously with the concept of algorithms and the use of flowcharts as an aid to developing algorithms and programs. It takes only a few hours for the instructor to develop a translator to accept BILLY-O code as input and simulate its execution so students can run programs. Our students have shown a strong interest and enthusiasm in learning the "inner workings" of a computer. When polled at the end of the course, 95% of the students indicated the BILLY-O language (a) was helpful in understanding how computers work, (b) made the concepts of BASIC more meaningful and easier to grasp and (c) recommended BILLY-O remain as a topic in future courses.

BASIC was chosen as the high level language over FORTRAN, PL/1, ALGOL, APL, COBOL, and PASCAL for the following reasons:

1. Ease of learning - a subset of BASIC is easier to learn than a subset of any of the other languages.

2. General purpose language - applicable to a wide variety of problems.

3. Common language - not only is it available on all large computers but many times is the exclusive high level language on microprocessors.

4. Used with time sharing systems.

COURSE DESIGN

The course is a three credit one semester course. As mentioned, approximately 35% of the course is spent on nonlanguage material and 65% on language instruction. One program is assigned using BILLY-O. The remaining five programs are in BASIC. The first exam covers material in the course outline up to and including BILLY-O. Students are allowed to use the material in appendix A during the test. Thereafter, students are not tested over BILLY-O so they can concentrate on learning BASIC. In addition we have or are planning to supplement course material with a tour of the university computing facilities, a speaker from a local business on a computer related topic, a tour of the facilities of a local industry such as an oil company, bank, hospital, newspaper, and at least one film. Mazlack [3] gives an excellent list of films. Students are evaluated on six programming assignments, three one-hour exams, seven to eight unannounced 15 minute quizzes and a two and one-half hour comprehensive final (excluding BILLY-O). Students are not allowed to make up any of the quizzes, but may drop their lowest quiz score. One semester we did not use any quizzes and we noticed a substantial difference in student performance. We found students were not reviewing lecture material between classes which is crucial in any computing course. Instead many students were treating this course like a "reading" course, trying to catch up once a week or so. Test performances were lower. Students had more difficulty programming and in general there was a higher level of frustration among the students and a higher incidence of students dropping the course. We found unannounced quizzes to be an effective teaching tool and recommend they be included in courses such as this.

DETAILED COURSE OUTLINE

The following is an approximate list of the topics in the order in which they are presented.

1. Introduction to course, course policy and requirements

2. Computer history

3. Evolution of the computer - computer generations

4. Characteristics of a computer

5. Computer organization - four functional parts

6. Steps in solving a problem on the computer

7. Algorithms

8. Flowcharting
   (a) Concept of counters
   (b) Concept of loops and loop control
   (c) Advantages of loops
   (d) Various ways to input data
      1. Number of data values is known
      2. First value indicates how many values follow
      3. Use of a trip value

9. Introduction to computer concepts using the BILLY-O language
   (a) The stored program concept
   (b) Transfer of control
   (c) Introduction to labels and machine storage
   (d) A look at the "inner workings" of a computer
   (e) How computers make decisions
   (f) Symbolic names

10. Keypunching

11. BILLY-O program assignment

12. Revolution in computer technology
   (a) Hardware - size, speed, cost capacity, reliability
   (b) Software - various categories
   (c) Computing modes - batch, real time, time sharing

13. Classes of computing devices - digital/analog, general/special purpose, etc.

14. Computer capabilities and limitations

15. Brief introduction to artificial intelligence

16. Computer input and output (characteristics, advantages and disadvantages)
   (a) Punched cards
   (b) Card reader
   (c) Paper tape - even and odd parity

(d) Magnetic tape
(e) Direct access devices
    1. Disk
    2. Drum
(f) Magnetic ink character recognition
(g) Optical character recognition
(h) Printers
(i) Terminals and POS stations

17. University computer facilities tour

18. Test I

19. Brief history of programming languages

20. High level and low level languages

21. Introduction to programming using BASIC
    (a) Elementary features - line number, constants, variables
    (b) Arithmetic operators and expressions
    (c) Priority of operators
    (d) Literal strings
    (e) Assignment statements, LET
    (f) INPUT and PRINT, TAB
    (g) IF - THEN
    (h) Labels and GOTO
    (i) Looping: FOR, NEXT
    (j) Nested loops
    (k) REM, END

22. Test II

23. BASIC and time sharing
    (a) Workspace concept
    (b) Concept of files
    (c) LOGON/LOGOFF
    (d) Various system commands in BASIC: CATALOG, CLEAR, DELETE, LIST, LOAD, PASSWORD, RENUMBER, RUN, SAVE, SYSTEM, WIDTH

24. More concepts in BASIC
    (a) One and two dimensional numeric arrays - DIM
    (b) Subscripted variables
    (c) Sorting and searching concepts
    (d) String variables
    (e) Concatenation
    (f) One-dimensional string arrays
    (g) LEN, VAL, STR
    (h) String comparisons
    (i) Brief introduction to subprograms

25. Test III

26. Interactive computing applications in the business world

27. Computer assisted instruction

28. Computers in Government and Law

29. Computer in Education

30. Computers and Health

31. Computers in Business

32. Computers in Humanities and Arts

33. Comprehensive Final

Summary and Conclusions

This paper has attempted to describe a first course in computers for non-majors. In particular the course is designed especially for those students who are specifically not required to take a computing course during their studies. There is no mathematical prerequisite. Students from a wide variety of backgrounds have enrolled in the course. The majority of students in the course are taking it from interest and not because of an academic requirement. Some, however, are enrolled to satisfy a mathematical sciences requirement. It has been a popular course as enrollment has been increasing with each year. Some students have even gone on to take other computer courses and a few have even changed majors to computer science or added computer science as a second major.

The basic structure of the course is to combine language instruction with various computer related topics to make the student aware of a computer's impact on society and how computers can be applied in their discipline. BASIC was chosen as the high level language primarily because it is easy to learn and widely available. BILLY-O is used as the low level language to show the "inner workings" of the computer. BILLY-O language is easy to learn because all of the traditional low level language topics need not be discussed. Students have indicated BILLY-O clears up the "mystery" of how computers work and overwhelmingly consider it an important part of the course. We have found few references in the literature suggesting the use of a low level language in a course such as this. In view of its success, we recommend a BILLY-O-like language be included. The course material is supplemented with a combination of tours, films and speakers. The course has been a success both in meeting our educational goals as well as popularity. Many disciplines are now strongly recommending their students enroll in this course and others are considering making it a requirement.

References

1. Cook, Robert. "An Approach to Introductory Computer Science Courses for Non-Majors," SIGCSE Bulletin 9, 3 (August 1977), pp. 30-33.

2. Graziano, S.M., and Gruener, W.B. "Survey Results and Conclusions: First Courses in Computers", Interface, The Computer Education Quarterly, 1, 1, (Winter 1979), pp. 42-49.

3. Mazlack, Lawrence J. "Developing Computer Awareness," SIGCSE Bulletin 9, 1 (February 1977), pp. 184-187.

4. Wainwright, Roger L. "A Survey of Faculty Computer Experience, Usage, Needs, Literacy and Attitudes," SIGCSE Bulletin 11, 2 (June 1979), pp. 27-35.

APPENDIX A - BILLY-O COMPUTER LANGUAGE

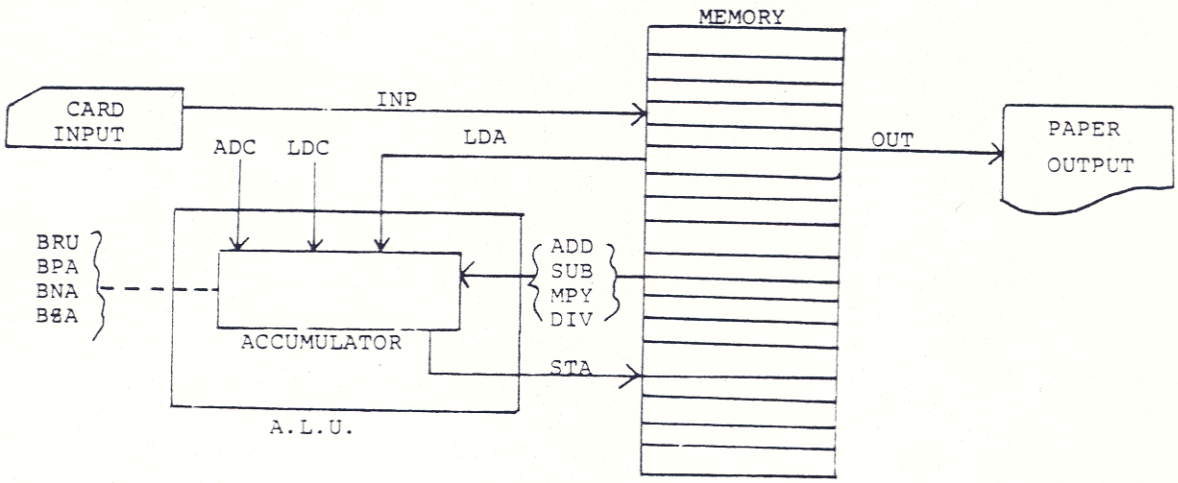
INSTRUCTION FORMAT:

COLUMNS

1-3	LABEL OR VARIABLE NAME (THIS IS OPTIONAL)
4	BLANK
5-7	INSTRUCTION CODE (DESCRIBED BELOW)
8	BLANK
9-16	VARIABLE NAME OR NUMERIC VALUE

INSTRUCTION CODES	MEANING
LDA	LOAD ACCUMULATOR--COPY FROM MEMORY TO THE ACC.(ACCUMULATOR)
STA	STORE ACCUMULATOR--COPY FROM ACC. TO MEMORY
ADD	ADDITION--ADD ACC. AND MEMORY; PLACE RESULTS IN ACC.
SUB	SUBTRACT--ACC. MINUS MEMORY; LEAVE RESULTS IN ACC.
MPY	MULTIPLY--ACC. TIMES MEMORY; LEAVE RESULT IN ACC.
DIV	DIVIDE--ACC. DIVIDED BY MEMORY; LEAVE RESULTS IN ACC.
LDC	LOAD CONSTANT--LOAD A CONSTANT INTO THE ACC.
ADC	ADD CONSTANT--ADD A CONSTANT TO THE ACC.; LEAVE RESULT IN ACC.
INP	INPUT (READ)--READ A VALUE INTO MEMORY
OUT	OUTPUT (WRITE)--WRITE A VALUE FROM MEMORY
BRU	BRANCH UNCONDITIONALLY TO THE INDICATED LABEL
BPA	BRANCH (IF THE ACC. HAS A POSITIVE VALUE) TO THE LABEL
BNA	BRANCH (IF THE ACC. HAS A NEGATIVE VALUE) TO THE LABEL
BZA	BRANCH (IF THE ACC. HAS A VALUE OF ZERO) TO THE LABEL
HLT	HALT EXECUTION (STOP)
(SPECIAL INSTRUCTIONS)	
END	LAST CARD OF THE PROGRAM (ALWAYS PRESENT)
NUM	INITIAL DATA VALUES FOR THE VARIABLES

FUNCTIONAL DIAGRAM FOR BILLY-O



APPENDIX B - SAMPLE BILLY-O PROGRAM

Problem: Read in an unknown number of test scores (range 0 to 100) determine and output the following:

(a) Average of all of the test scores
(b) Maximum test score
(c) Minimum test score
(d) How many test scores > = 60.

Assume a trip value of 9999

```
WWW INP SCR        INPUT A TEST SCORE
    LDA SCR        CHECK FOR A TRIP VALUE OF 9999
    ADC -9999
    BZA BOT
    LDA CNT        ADD 1 TO CNT (CNT COUNTS THE NUMBER OF SCORES)
    ADC 1
    STA CNT
    LDA TOT        ADD SCR TO TOT (TOT IS THE TOTAL OF ALL THE SCORES)
    ADD SCR
    STA TOT
    LDA MAX        CHECK IF SCR IS A NEW MAXIMUM
    SUB SCR
    BPA XXX
    LDA SCR
    STA MAX
XXX LDA MIN        CHECK IF SCR IS A NEW MINIMUM
    SUB SCR
    BNA YYY
    LDA SCR
    STA MIN
YYY LDA SCR        CHECK IF SCR > = 60
    ADC -60
    BNA WWW
    LDA GT6        ADD 1 TO GT6 (GT6 COUNTS THE NUMBER OF SCORES > = 60)
    ADC 1
    STA GT6
    BRU WWW
```

```
BOT LDA TOT        EOF - DETERMINE THE AVERAGE TEST SCORE
    DIV CNT
    STA AVE
    OUT AVE        OUTPUT AVE, MAX, MIN AND GT6
    OUT MAX
    OUT MIN
    OUT GT6
    HLT
AVE NUM 0
CNT NUM 0
GT6 NUM 0
MAX NUM 0
MIN NUM 100        MAXIMUM SCORE IS 100
SCR NUM 0
TOT NUM 0
    END
```