MASTER OF ARTS IN EDUCATION SAINT MARY'S UNIVERSITY

AIMS AND GOALS OF HIGH SCHOOL COMPUTER SCIENCE CURRICULA WITH IMPLICATIONS FOR NOVA SCOTIA

BY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE MASTER OF ARTS (EDUCATION)

APRIL 8, 1987

Approved: G. E. Forrester
Faculty Advisor

Approved: Frances B. Phillips
Dean of Education

ISBN 0-315-40290-3

AIMS AND GOALS OF HIGH SCHOOL COMPUTER SCIENCE CURRICULA WITH IMPLICATIONS FOR NOVA SCOTIA

Lonny G. MacLeod

April 1987

ABSTRACT

AIMS AND GOALS OF HIGH SCHOOL COMPUTER PROGRAMMING CURRICULA WITH IMPLICATIONS FOR NOVA SCOTIA

Lonny G. MacLeod

April 1987

This study examines computer programming courses offered in ten American states and nine Canadian provinces and compares them to the Computer Related Studies course offered in Nova Scotia high schools. A detailed look at contemporary literature reveals current trends in educational computing which are examined in light of present course offerings in Canada and the United States. Specific similarities and differences among the various programming courses are discussed, and specific recommendations are made for the future of secondary computer programming in Nova Scotia. Three tabular appendices (I, II and III) compare the states' and provinces' courses and a short glossary of computer terms is given in Appendix IV.

ACKNOWLEDGEMENTS

No work of this type could have been completed without the help of many other people. I would like to thank the folowing for their assistance: all those education experts from across Canada and the United States who took the time to provide information on the courses taught in their province or state, Fred Crouse and Donna Giles of the Nova Scotia Department of Education, Dr. Fred Dockrill and Dr. Donald Weeren for their assistance in cleaning up a multitude of grammatical sins, Professor Kevin MacLeod for his constant encouragement and assistance, and, above all, Professor Glane Gorveatt for always being there when I needed help and direction.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

1. Literature Review

1.1 Introduction

The purposes of this thesis are: to examine recent literature on secondary computer programming courses, to explore curricula used in teaching these courses and to evaluate Nova Scotia's new CRS (Computer Related Studies) 441 course.

In September 1986, Nova Scotia schools began teaching the new Computer Related Studies (CRS) course. This course had been in existence for several years, but there were no set curriculum guidelines for teachers to follow. Teachers of the course were using widely varied hardware, software and textbooks and a wide variety of courses resulted. Some courses were literacy oriented, while others were almost exclusively dedicated to programming. The new curriculum guidelines attempt to standardize teaching of CRS 441 and provide teachers with direction and enrichment materials. This thesis should help Nova Scotia educators to determine whether the new curriculum guidelines are giving the CRS 441 course the direction we need for the 1990's.

1.2 Recent Literature on Computer Programming Courses

An examination of the literature on secondary programming courses led to the discovery that, although a large quantity of research has been done with regards to computers in education, a relatively small amount of

Page 2 omitted in page numbering

current research deals with programming courses. Since 1980, the Education Index and the Current Index to Journals in Education (CIJE) list fewer than fifty articles written concerning programming courses. Of these articles, many are out of date and others either are concerned with topics belonging to literacy rather than programming courses or with hardware and financial considerations. This relative scarcity of information may be attributed to the "revolution" in computer education in the United States (where most studies of computer education take place) now emphasizing the implementation of computer aided instruction (CAI) in all subject areas at all grade levels.

Courses which teach programming skills have been in existence for several years, but have changed as hardware and software have evolved. Articles written since 1980 were chosen for this study because of a desire to examine the most current literature available and to avoid the use of outmoded information. There now appears to be an emerging consensus on content, direction, format and output (see Chapter 2, Section 25). Many computer education experts have turned their attention away from programming and toward more recent areas of interest in computer education (networking, use of computers in non-technical fields, accessing databases, etc.). The literature on these topics was not as useful for this study as were articles on programming topics.

1.3 Programming Languages

Any discussion of computer education should begin with the views of Dr. Alfred Bork of the University of California, who has authored several texts and numerous articles concerning computer education and is in the forefront of educational computing. He leads the push for structured programming in schools, a teaching method which emphasizes that program coding should be undertaken in much the same manner as problem solving in mathematics: through a series of carefully thought out steps which may be used to solve other, similar problems. Bork (1984) stated that the teaching of programming in schools is a disaster area and builds up habits which are very difficult to overcome in later life. He believes these problems are caused by use of the BASIC language and by teachers who do not understand programming. In an earlier article (Bork, 1983), he stated his feelings that BASIC should be avoided at all costs. BASIC is a poor choice for a programming language because it allows the creation of "spaghetti code" (hopelessly tangled program structure) and lacks other attributes of structured languages. These shortcomings include adequate control structures, procedures and subroutines.

Pascal seems to have become the language of choice in most programming courses. As stated above, many experts in educational computing fear that most children programming in subsets of BASIC will be unable to write understandable

code. BASIC has its adherents, but even they are beginning to accept the fact that students should learn a structured programming language such as Pascal so that they can avoid the confused coding methods which are learned by beginning BASIC programmers and which must be later "unlearned." In his discussion of the Tennessee High School Computer Science Project; Baird (1984) states that, by using Pascal and a very careful and thorough approach to both hardware and software, a course evolved which showed that:

1. students of average ability can learn structured, top-down programming.
2. students can learn to break down problems into smaller tasks and test procedures before combining them into a complete solution.
3. working in teams can relieve pressures, promote peer teaching and buffer fragile egos.
4. debugging programs as a group helps students who may view errors as personal failures to realize that mistakes are nuisances which happen to all programmers.

This pilot course proved over an extended period of time (three years) that Pascal could indeed be taught successfully in high schools and that peer teaching is very desirable when learning to program.

Anderson (1982) compared the relative merits of BASIC and Pascal and found that, while BASIC fell short in terms of modularity, readability (especially where lengthy

programs are required) and lack of mnemonic names for variables (variable names are usually limited to two letters), Pascal excelled in all three areas and should be given serious consideration when choosing a programming language.

Brown (1984) praised Pascal as being a language specifically designed to teach programming. He felt that one of Pascal's strengths was its ability to break complex problems down into a series of smaller tasks (procedures), which made it the best language for use at the secondary level. Five reasons were listed why Pascal should be taught: it encourages good programming practices, it gives students access to other programming languages such as C and Ada, it is the language used in introductory university courses, it may be used (in the U.S.) to give students advanced placement credit in computer science and it enables programmers to write programs of significance which do not lose clarity as they become lengthy.

Er (1984) stated that the choice of a programming language was critical to the success of a programming course. BASIC and FORTRAN are to be avoided because of their use of the GOTO statement which complicates a program's logic and can easily destroy its structure. Er suggested the use of Pascal as a programming language, but without the GOTO statement. His experiences indicated that students trained using BASIC, FORTRAN or COBOL have difficulty producing well-structured programs and that

students often produce programs in which they have little confidence. Er believes that computer programming involves two distinct activities: designing algorithms and coding the resulting algorithms, and that course time should be evenly divided between these two activities.

Bitter (1983) set out a computer curriculum which introduced Pascal at the Grade Ten level and taught advanced Pascal and database programming in Grade Twelve. He stated that high school students can not only deal with Pascal, but that Grade Twelve students can engage in a variety of self-teaching activities using Pascal and databases. It must be noted that this ambitious plan is aimed at students who have received some sort of computer instruction throughout their formal education.

Masterson (1984) identified his requirements for a programming language as: simplicity, power, compatibility and cognitive richness. Even Pascal, the programming language choice of the majority of secondary curricula, was found wanting because programs are created in an editor (programs must be compiled and run before they can be debugged) and procedures must manipulate the content of memory registers one at a time. BASIC was praised for its interactive nature (programs can be run and therefore debugged without compiling). Unfortunately, BASIC lacks adequate control structures, subroutines and procedures. APL and LISP were rejected because, although they are very powerful, they are also difficult to learn. Masterson

recommended AMPL and Logo because both languages share features which Masterson feels makes them ideal for use as programming languages:

1. powerful, interactive code
2. powerful primitives for creating and altering whole data structures
3. functional notation that often emphasizes the hierarchical structure of a computation
4. dynamic memory allocation
5. stored workspaces containing variables and function definitions
6. user access to system variables

Some educators object to the use of AMPL, a rather mathematical programming language, in high schools because many programming courses are beginning to de-emphasize mathematics in an attempt to establish computers as a separate discipline and to Logo because it is thought to be aimed mainly at younger children. However, Masterson stated that, of the languages currently available, they most nearly provide flexible, interactive, powerful programming languages for the student.

1.4 Problem Solving

Bork (1981) stated his belief that students do not need courses which deal only with programming and that introductory courses should accomplish more than teaching students to code programs. The algorithmic approach to

problem solving should be stressed along with the application of structured programming techniques. He identified the diversity of preparation among computer students (some know absolutely nothing, while others may be more knowledgeable than their instructors). The grammatical approach, which involves an exhaustive examination of the syntax of a computer language, was rejected as a teaching technique and instructors were encouraged by Bork to allow students to learn by examining, running and modifying existing programs of ascending complexity before they begin writing programs of their own. In this way students will learn what does and does not work when constructing their own programs, a method very similar to the "whole language" approach used in teaching foreign languages. Students who learn this method of programming begin to write programs based on realistic problems as soon as possible and program in more than one computer language so they can gain insight into the nature of programming languages by comparing the features of each language studied.

Swigger (1984), in her review of computer education literature, cited several studies which showed that the use of a model of some type when teaching programming resulted in improved comprehension and facility in writing programs. She noted an interesting study by Shneiderman (1977) which found that the use of flowcharts had no effect on student performance in learning to write programs.

Studies by both Lemos (1978) and Cheney (1977) demonstrated the value of the team approach to programming and problem debugging. Swigger concluded that novice programmers in general have difficulty formulating problem solving algorithms because of their inability to think in general terms and that novices have the greatest problems with I/O (Input/Output), assignment, looping and recursion statements.

Coburn (1985) cited a study by the University of Massachusetts which showed that algebraic word problems were easily solved by student when stated in a programming-task format. He also stated that unlike many of the innovations of the sixties, computers seem to have strong fundamental support among teachers, administrators and the community. Many educators are concerned that computers in the classroom may become neglected and forgotten, as educational television was in the seventies. This does not seem to be happening, although Coburn warns that while computers will continue to affect our everyday life, they could disappear from the classroom if efforts are not made to maintain the momentum which has begun in educational computing.

1.5 Teacher Training

Teacher training is identified as another major problem area, not because universities fail to teach programming well, but because universities are not preparing teachers

to teach computing in the schools. Two explanations for this are offered: many instructors currently involved in teaching programming have little preparation because their professional training predated the introduction of computers in the schools and, especially in the United States, teachers with computer training are leaving education for higher paying jobs in the computer industry.

Weizenbaum (Brady, 1985) holds opinions similar to Bork's in that both believe that teacher training is inadequate with respect to computers and both are not in favor of teaching programming only courses to secondary students. However, Weizenbaum feels that the larger problems faced by education such as problems with mathematics and science teaching are much more urgent and that time spent teaching programming courses would be better spent trying to improve math and science skills.

Schwartz (Brady & Levine, 1985) believes that no one is well served by having teachers who are unskilled programmers teaching children to be unskilled programmers. He also believes that while computer literacy (especially applications such as spreadsheets and databases) is important, programming should be taught only to those who intend to pursue programming careers, since most jobs involving computers require little or no programming knowledge.

Teacher education is also addressed by DeVault and Harvey (1985), two computer educators who run Computer

Fest, a summer school workshop at the University of Wisconsin, which mixes students and teachers (of various levels of knowledge) together with beneficial results for both teachers and students. They stated that the computer science curricula should not be rigidly defined, but should be able to evolve and benefit from new insights which develop.

1.6 Curriculum Issues For Programming Courses

Ragsdale (1982) stated that programming courses should be divided into program design (selection and creation of an algorithm) and program coding (transcribing of the selected algorithm into the appropriate computer language). He believes in the implementation of a peer teaching system in which knowledgeable students help their less able classmates, that secondary programming courses will eventually become similar in content to university courses due to the fact that the courses will increasingly be aimed at those who plan to study computer science at a higher level and that teachers must master innovative instructional techniques which downplay traditional teaching methods in favor of a more hands-on, individualized approach. Ragsdale also stated that student teachers should be given greater exposure to computers during their professional training in order to prepare them for the computer-literate students they will be teaching.

Plog (1984) noted that leadership is a major problem in educational computing, not because of a lack of leadership, but because of a lack of followers. This results in duplication of effort and a total lack of consensus on computer education at any level, although some groups such as MECC (Minnesota Educational Computing Consortium) have produced materials of such high quality that they have become a de facto standard. Plog believes that curriculum is determined by a combination of school teachers, professors in the field of study and textbook authors. He also believes that computer education lacks the natural organization of professionals and some tradition of what makes up the field of study. He agrees with Wagner that time must be allowed for computer education to develop before any consensus can be reached in curriculum planning and implementation.

1.7 Advanced Placement (AP) Computer Science

At the present time many American computer education specialists are implementing an advanced placement course in computer science, which gives students university credits for a course covering material well beyond the scope of present high school offerings. The rationale behind this is that the test for the course which has been designed by university computer instructors will give secondary teachers guidelines and goals to use in their courses. The course has settled on Pascal as the

programming language to be taught, for the reasons discussed above (Task Force on School Curriculum, Task Force on Teacher Certification, 1985).

Some educators such as Wagner (Watt (1983)) of Computer Using Educators (CUE) oppose the test because they feel it prematurely imposes standards on computer courses. Watt (1983) stated that, to the extent the AP course eliminates the idea that BASIC is the best language for teaching programming, the course is a positive development, as long as another premature standard is not imposed in place of BASIC and that we cannot afford to be locked into a current standard when the situation is changing so rapidly. Braswell (1984) stated some of the tenets of the new course: it is more than a programming course, it is not intended as a vocational training course and it is not intended to replace current secondary computing courses.

1.8 Conclusion

At first glance, there appears to be little argument among computer educators in this area, but two opposing groups can be identified: those who view programming as a vital part of a student's computer education and those who feel that CAI (Computer Assisted Instruction) and computer literacy are more important goals than learning a programming language. At the present time, the forces of programming have the upper hand, as evidenced by the U.S. federal government recommendation that every American high

school student should receive a minimum of one half course in computer programming before they are permitted to graduate.

2. Secondary Programming Courses in the United States, Europe and Asia

2.1 Introduction

This chapter examines specific curriculum materials from the U.S. and very briefly discusses programming courses elsewhere. In the United States most educational decisions take place at the school board level, with state and federal input usually occurring in the form of financial incentives and mandated curriculum requirements. It should be noted that information from only twenty two American States is included in this chapter because some states did not reply to requests for information and some states had little or no information available on programming courses.

2.2 U.S. National Initiatives

American state governments lack the overt control of education which Canadian provinces enjoy. However, they exert a great deal of indirect control through funding, legislation, etc. There are several national voices of computer education including Minnesota Educational Computing Consortium (MCCE) which develops top quality courseware and distributes it nationally, International Council for Computers in Education (ICCE) which produces excellent teaching materials for elementary and secondary computer educators and Association for Computing Machinery

ACM), a national organization of computer educators at all levels (although mainly concerned with post-secondary computer education).

In 1983 the ACM struck a task force which was to make curriculum recommendations for secondary computer science courses. They stated that computer science should be offered to as many students as possible and should not be limited to students who show promise in the subject area. The task force laid out a four course curriculum in computer science:

1. Introduction to Computer Science I (full-year)
2. Introduction to Computer Science II (full-year)
3. Introduction to a High-Level Computer Language (half-year)
4. Applications and Implications of Computers (half-year)

The first two courses are "computer science" courses in the traditional sense and the final two courses fit into what is currently termed "computer literacy." Pascal is recommended as the programming language to be used in courses one and two, although structured BASIC is listed as an option for Introduction to Computer Science I because beginners usually find an interpreted language (BASIC) easier to work with than a compiled language (Pascal). Students who successfully complete both courses should receive credit for a first year introductory computer science course at the university level, which places the

level of the two courses beyond any broadly available programming courses currently available in Canada. Introduction to Computer Science I includes the basic idea of structured programming and top-down design. Students should learn to design (structured) algorithms to solve (programming) problems and the coverage of algorithms and design techniques should be general enough to enable the student to use this course as a foundation for structured problem solving in any language. Introduction to Computer Science II is designed to develop the concepts introduced in the previous course. Programming will occupy approximately two-thirds of the instructional time with students being expected to complete several individual and group projects through the year. Both courses are designed to be within the abilities of all high school students, but it is expected that only students with a real interest in programming will take both courses. The ACM also makes recommendations for teacher certification in secondary computer science which include three courses in computer science and two courses in computer education because of the ACM's belief that self-trained computer specialists are inadequate for the proposed curriculum (Task Force on School Curriculum, Task force on Teacher Certification, 1985).

Education Turnkey Systems, an American educational contracting firm, conducted a study in 1985 on educational computer use which found that secondary schools use their

computers more for computer literacy and programming than for any other activity and that drill and practice uses are very limited at the secondary level. It was also determined that 47 states now offer some form of teacher training in educational computing and that 25 states have some form of guidelines for secondary computer courses. Programming in BASIC was taught in 53% of American schools offering computer literacy courses, while other programming languages were found in only 8% of schools (Education Turnkey Systems, 1985).

U.S. State Initiatives

2.3 Alaska

Alaska has had the most favorable computer to student ratio in the United States for the past two years (see Appendix I). This is due to the state's commitment to implementing computers across the curriculum. Very few schools teach programming as a separate subject, with an across the curriculum approach to computers being preferred (Lind, 1986).

2.4 Arkansas

The state of Arkansas is in the process of requiring all school districts to offer a course in computer studies which will be given to students in Grade 9-12. The four major units of study in this course are: Fundamentals,

Applications, Programming and Logic and Ethical and Social Implications (Watson, 1986).

Problem solving is to be stressed throughout the course with programming to be taught using the following procedure:

1. Define the problem
2. Plan and design strategies to solve the problem
3. Write structured code
4. Test and evaluate results
5. Refine solutions
6. Write documentation

The Programming and Logic Unit includes the following topics: flow charts, describing program output, executing programs, editing, modifying existing programs, problem solving using programming and programming techniques (I/O, calculations, variable handling, internal documentation, conditional statements, graphics, looping, screen formatting, random number generation, string handling techniques, subroutines, arrays, printer formatting, animation graphics, sequential file handling, logical operators, simulation graphics, iterations, Monte Carlo techniques, random file handling, information interchange codes and skills in a second programming language). The main language used in this course is BASIC with Pascal used in some districts.

2.5 California

The California State Department of Education recommends a computer science course which is intended for those students with a keen interest in computers. It is stressed that the course should be adjusted to match the interests and abilities of the students being taught. The course does not concentrate on programming since California students receive computer instruction throughout their schooling, but instead stresses ideas which are thought to hold great promise for the future such as computer architecture, procedural languages, data representation and storage, circuit design and job market information (Smith, 1987).

2.6 Connecticut

The state of Connecticut does not have a mandated curriculum for computer science courses, with local school districts having the final say about the type of courses which are offered. Suggested high school programming courses include BASIC, Logo, COBOL and Pascal (mainly in the Advanced Placement Computer Science Course). The suggested goals for high school programming courses are: to be able to create successful programs through an advanced BASIC course (subscripted variables, substrings, output formatting, sequential and random access files, graphics), to use peripherals for programming purposes (store data files using a DOS (Disk Operating System), to use word processing systems, access communications networks, visit

computer installations) and to have the opportunity to take an additional programming course (advanced problem solving techniques, assembly and/or machine language programming, other high-level languages) (Naimi, 1987).

The Bristol Board of Education (1983) in its Computer Education plan lists a course entitled "Advanced Computer Programming", a half credit math department offering which has as its objectives that the student will demonstrate: knowledge of advanced BASIC programming skills, knowledge of program planning, development, implementation and assessment of problem solving applications and skill in completing projects. Independent study is recommended with a staff member dealing with students on an individual basis and the majority of the course work being undertaken as projects.

2.7 Florida

Florida is one of the states which is vigorously promoting computer education. The state Department of Education is developing a statewide test for computer literacy and has set guidelines for eleven different programming courses offered at the secondary level: BASIC I and II, Computer Programing I, II and III, Pascal, COBOL, PL/C, FORTRAN, AP Computer Science and Computer Studies-International Baccalaureate (Florida Department of Education, 1983).

BASIC I is designed to teach the fundamentals of programming in BASIC and includes topics such as subroutines, looping, branching, string functions and graphing. Its aims include teaching students to: analyze problems and develop algorithms and flowcharts, implement programs based on the arrived at algorithm, describe computer systems, discuss social implications of computers and show an understanding of structured programming. BASIC II extends the Basic I course with a concentration on the following topics: subscripted variables, matrices, string manipulations, sorting techniques, disk operations, data files and advanced graphics. BASIC II has aims very similar to those of BASIC I and extends those topics covered in the previous course.

Computer Programming I is designed to introduce students to a high level language, with or without the inclusion of computer literacy topics. Its intended outcomes include being able to: create algorithms, use syntax, vocabulary and data structures of the programming language being studied and develop programs in the selected language. Computer Programming II uses a high level language different from the one taught in Computer Programming I and extends the previous course to include file handling techniques, detailed syntax and comparisons of high level languages. This course is designed to enable students to: develop algorithms independently, translate these algorithms into computer programs, use file handling

techniques and become familiar with other programming languages. Computer Programming III is designed to study multiple programming languages in depth and includes study of algorithms, syntax and problem solving. Its intended outcomes are to develop facility with the syntax of suggested programming languages and to enable students to choose the appropriate language for a particular problem.

Pascal, PL/C, FORTRAN and COBOL include those elements of the respective languages necessary to write simple programs and list as their intended outcomes to develop and implement algorithms to solve problems.

The purpose of Advanced Placement Computer Science is to examine applications of computing in the contexts of programming methodology, algorithms and data structures. Its intended outcomes include being able to: design and implement computer based solutions; explain, develop and select algorithms; code programs in a structured manner; explain the components of a computer system and examine the social implications and ethical considerations of computer use (see Section 1.7 for further discussion of this course). Computer Studies- International Baccalaureate follows the content of the International Baccalaureate Computer Studies program and is intended to teach: a logical problem solving approach, good programming practices and familiarity with computers and their societal implications. The course's aims are to enable students to demonstrate in-depth skill in a programing language, study

logical processes in problem solving and learn about computer architecture.

2.8 Georgia

Georgia has no mandated curriculum in computer science and computer instruction varies widely across the state's 186 school systems. The state's "Essential Skills For Computer Technology" lists the following topics: A. Knowing about and using computers and other electronic technology, B. Understanding the social impact of computers and C. Programming computers (Georgia Department of Education, 1986).

2.9 Indiana

Indiana has mounted a serious effort to educate its teachers (and consequently its students) in computers. In its first two years of existence, the Indiana Consortium for Computer and High-Technology Education reached over 31% of Indiana teachers through inservices, workshops and summer courses. The state mandates that all students must have the opportunity to become computer literate, a requirement which can be satisfied in one of the following ways: integrating computers into the existing elementary and junior high curriculum, including a unit of study of at least nine weeks within an existing junior or senior high school course, teaching a separate high school course of at least one semester and developing an educational

improvement plan. A high school programming course may teach Logo, BASIC, Pilot or Pascal. The choice of language and the framework of the course is independently set by each school district in the state (Usher, 1986).

2.10 Iowa

Iowa's recommended computer science course is taught at the Grade 11 or 12 level with most schools using BASIC (with an growing reliance on Pascal). The major emphasis in this course is on problem solving and programming methodology using a high-level language. The skills which are to be developed include: mastery of BASIC language commands, being able to read and explain a computer program, locating program bugs, working with files, seeing the need for documentation and appreciating the program design process. The course goals for computer science are: to introduce problem-solving techniques, to introduce algorithm development, to teach good programming style, to teach a high-level programming language, to provide familiarity with computer systems, to provide a foundation for further studies and to provide career information. The student goals for the course are: to understand good program design; to be able to read and explain programs; to demonstrate mastery of a high-level language; to locate bugs in programs; to load, save and merge files and do simple calls to and from files and to write a program of significant length (more than 500 lines) (Iowa Department of Public Instruction, 1986).

2.11 Kentucky

Many schools in Kentucky offer the following computer courses at the secondary level (the title of the courses may vary from district to district): Introduction to Computer Applications, Computer Programming, Structured Programming and Advanced Placement Computer Science. Teachers of these courses must meet certification requirements laid down by the state which require an increasing amount of computer training as the level of the course becomes higher (Kentucky Department of Education, 1984, 1986).

The Introduction to Computer Applications course is intended to introduce computers to students who have no previous computing experience. Some introduction to BASIC programming is given, but the main thrust of the course is toward helping students understand the role of computers in society and introducing them to a variety of software packages.

The Computer Programming course is designed for students with career aspirations in the computer field and for students wishing additional enrichment in mathematics. The course emphasizes programming using a high-level language (BASIC or Pascal). Students enrolled in this course will be expected to: develop expertise in using commands and functions, identify symbols for arithmetic calculations and relational expressions, learn common error message and correct responses for each, understand how the

computer deals with numerical concepts, apply concept of subscripted variable, build and use sequential and random access files, learn printer control and graphics commands and develop universal techniques of communication and standard communication formats (ASCII).

Structured Programming extends the programming concepts learned in the Computer Science course. This course utilizes one or more programming languages (Pascal, FORTRAN) and has the following student expectations and experiences: using computer commands and functions; identifying symbols for arithmetic calculations and relational expressions; understanding and responding to error messages; understanding numerical concepts; using subscripted variables; data communication and applying knowledge attained to write programs independently.

The aims of the Advanced Placement Computer Science follow those laid down by the College Board (see Section 1.7) and the course is designed to prepare students for the Advanced Placement Computer Science Examination.

2.12 Louisiana

Louisiana's computer science course is very well defined in the state's Computer Science Curriculum Guide. It lists the following as general guidelines: variables' names must reflect their function in the program, variable names should not be abbreviated excessively, programs must be structured properly, with subprograms used whenever

possible, source programs should be formatted for readability, GOTO statements should be avoided whenever possible, simple and direct program construction is to be stressed, programs should be written in pseudocode before being coded, bad code is to be rewritten, not patched, program correctness is more important than program execution speed, all programs should check for bad input, output must be formatted for comprehension and the top-down approach is to be used when writing programs. The goals of this course are:

1. Gain understanding of history of computers
2. Learn basic design and functions of computers
3. Understand algorithm construction and flowcharting
4. Know and apply principles of structured programming
5. Understand construction and uses of symbolic language in structured programming
6. Understand technological aspects of computer design and operation

The five units which make up the course are entitled: I Computer Literacy, II Algorithm Construction and Flowcharting, III Steps in Writing a Program, IV Programming Procedures and V Optional Advanced Topics (Louisiana Department of Education, 1983).

The Computer Literacy unit includes the following topics: Methods of Calculation, Manual Aids, Mechanical Aids, Punched Cards, Automatic Mechanical Aids, Computer Age, Types of Computers, Hardware and Software, Functional Units of a Computer and Uses of Computers.

Unit II, Algorithm Construction and Flowcharting, covers two topics: Problem Analysis and Flowchart Symbols. Problem Analysis refers to defining the problem to be solved, reducing the solution to a series of discrete steps and arranging the solution steps in a logical sequence.

The Steps in Writing a Program unit lists seven topics: Statement of Problem, Algorithm and Flowchart, Write Program, Trace Program, Run Program, Debug Program and Documentation.

Unit IV, Programming Procedures, covers fourteen areas: Available Languages, System Commands (execute, list, save, load, erase, edit), Remarks or Comments, Output (Literals), Variables (numeric, alphanumeric, alphabetic and logical), Input/Output (methods of I/O and formatting), Assignment Statement and Elementary Library Functions, Rational and Logical Operators (relational operators, logical operators), Control Structures (IF..THEN..ELSE, WHILE..DO, REPEAT..UNTIL, ON.. GOTO, CASE, FOR..NEXT, DO..CONTINUE), Subscripted Variables (dimension arrays, I/O arrays, sorting routines, storing concatenation, operations with arrays), Subprograms (purpose and construction, related commands), Graphics, Data Files (creating, reading, sequential and random access files, updating) and Advanced Programming Projects.

The five topics covered in Unit V (Optional Advanced Topics) are: Computer Design and Operation, Arithmetic-Logic (instructions registers, data registers,

arithmetic-logic units), Machine Language, Assemblers and Compilers (source and object programs, compilers) and Assembly Language.

2.13 Maine

The state of Maine passed a law in 1984 requiring that computer instruction be available for all students and that standards for computer literacy and proficiency be required of all students before they are permitted to graduate. The state Department of Education is currently emphasizing applications for all students to meet the proficiency requirements with programming serving as an enrichment topic for more advanced students. BASIC is the most popular language followed by PASCAL (to meet the requirements of the AP course) and, to a lesser extent, LOGO, COBOL and FORTRAN. Maine does not currently have a certification program for teachers with most teachers gaining expertise through experience or university upgrading courses. Dennis Kunces, the Maine Department of Education computer consultant, has stated that CAI (Computer Aided Instruction) and applications are a higher priority than the teaching of programming in state high schools (Kunces, 1986).

2.14 Maryland

Charles Phipp (1984) of the Montgomery County Public Schools stated the assumptions made by his district about

computer education: all students should be able to take elective courses in computer education, most future computer users will not be programmers, high school courses should be centered on problem solving, courses should deal with social and ethical issues and students serious about computer science should spend much of their time taking demanding courses in other subject areas (including English and foreign languages). Five new computer courses are being offered in the Montgomery County school system, culminating in a Pascal course and an advanced placement computer science course. This district has embraced Pascal as the programming language to be taught to students in high school and seems to be in the forefront of educational computing with its range of secondary offerings in computer science.

2.15 Mississippi

Mississippi has determined that programming courses must be available for all students and that computer education in general is a crucial need. The overall goals for computer education are to instruct students in: history of computers, computer applications in our society, use and care of hardware and software, structure of computer systems, computer programming using a structured approach, use of problem-solving techniques and practical programming to show the applications of computers in the real world. Five half-year high school programing courses are offered:

BASIC, Advanced BASIC, Pascal, COBOL and FORTRAN (Mississippi Department of Education, 1986).

The units which make up the BASIC course are: Orientation With Resident Computer System, Introduction to BASIC and Structured Programming, Input/Output/Processing, Decision-Making and Branching and Single and Multi-Dimensional Arrays. Advanced BASIC extends the introductory BASIC course and includes programming techniques such as numeric functions, string functions, user-defined functions, advanced search routines and advanced sort techniques. Sequential and random files are introduced and discussed at length and error trapping is covered briefly.

The Pascal course covers the following topics: Orientation With Resident Computer System (start-up, maintenance, execution and compilation of programs), Introduction to Pascal Programming (problem-solving techniques, top-down program design, variables, structure of simple programs), Manipulation of Simple Data Types (arithmetic and Boolean operators, predefined functions and procedures), Input/Output/Processing (READ, READLN, WRITE, WRITELN, logical devices and simple data files), Selection Control Structures (IF..THEN..ELSE, CASE, REPEAT..UNTIL, WHILE..DO, FOR..DO, ITERATE), Procedures and Functions (block structure, scope of identifiers, parameters, recursion) and Structured Data Types (arrays, records).

The COBOL course is made up of five units: Orientation With Resident Computer System (as in Pascal course), Input/Output Operations (program divisions, clauses, open, read, perform, close, Stop Run, and Move statements, field theory, Output, Input), Arithmetic Operations and Report Editing (definition of numeric data, basic arithmetic operations, Compute and Zero statements, insertion characters, program constants, heading lines, spacing), Comparing (IF/IF-THEN-ELSE structure, relation test, numeric and alphanumeric comparisons, And/Or logical operations, condition names, accumulators) and Control Breaks (Control Break processing, multilevel control breaks, indicators, compare area, total accumulators).

There are eight units in the FORTRAN course: Orientation With Resident Computer System (as in Pascal course), Introduction to FORTRAN Language (integers and reals, variables, column specifications, arithmetic operators), Input and Output (FORMAT, WRITE, READ, I-FORMAT, F-FORMAT, X-FIELD, DATA statements), Branching Techniques (decision-making, GOTO, CONTINUE), Functions (definition, use), Looping Techniques (DO statement, nesting, premature ending), Arrays (DIMENSION, type declaration, use of DATA statement, implied DO loops, Input/Output of whole arrays) and Character Manipulation (A-FORMAT, input/output).

2.16 Missouri

Missouri currently requires high school computer science teachers to possess a science certificate plus five (science-oriented) college hours in computers. As of the fall of 1985, the vast majority of schools offering courses in computer science used BASIC as the main programming language (Ogle, 1986).

2.17 New Hampshire

All New Hampshire schools are required to offer students computer literacy instruction. This requirement can be satisfied by courses at any level or by a program of integrating computers into the existing curriculum. Most high schools offer a programming course which relies on BASIC, but the use of Pascal is increasing rapidly. Courses vary widely across the state and may be classed as business, mathematics or computer science courses. There are currently no concrete certification requirements for computer science teachers (New Hampshire State Department of Education 1985, 1986).

2.18 North Carolina

North Carolina high schools offer a course in Computer Mathematics at the Grade Eleven and Twelve level which is not language specific, but BASIC is used by most teachers of the course. The Advanced Placement Computer Science course which is taught at a limited number of high schools

uses Pascal exclusively (see Chapter 1 Section 7). Students taking Computer Mathematics receive a mathematics credit on completion of the course, which deals with advanced mathematical concepts through computer programming (North Carolina Department of Public Instruction, 1986).

The eight units of Computer Mathematics are titled as follows: Mathematical Formulas (fundamental programming statements), Coordinate Geometry (Graphics), Equations, Inequalities and Polynomials (subroutines and procedures), Mathematics of Finance (counters, accumulators, nested loops, rounding, arrays), Number Theory (INT function), Probability and Statistics (subscripted variables, random number generation), Measurement of Geometric Figures (trigonometric functions) and Matrices and Determinants (two-dimensional arrays).

2.19 Oregon

The state of Oregon has been intensively involved in computer science curriculum development for many years. There are currently no certification requirements for computer teachers in the state and no "recommended" computer science curriculum. Tillamook High School offers two full year courses in computer programming, one in BASIC and an advanced course in Pascal, assembler and machine language (Scheneberger, 1987).

Gold Beach Union High School offers a full year computer programming course at the Grade Eleven or Twelve

level which briefly reviews BASIC and moves on to Logo and Pascal. The student goals for this course are: be able to read, modify and write programs, know the components of a computer, be able to operate peripherals, and be aware of careers in computing. The course is divided into four units: Introduction, Use of Computers, Use of Peripherals and Programming (Renner, 1987).

2.20 Tennessee

Tennessee's computer technology education courses aim to increase a student's knowledge in many different facets of computer education, including the development of computer technology, careers in computer-related fields, general knowledge of computer systems, programming skills and problem solving techniques. The state department of education has authorized the following secondary courses: BASIC I and II (classed as business education courses), Pascal I and II, LOGO and FORTRAN (Tennessee Board of Education, 1986).

The objective of BASIC I is to understand and use the BASIC language to solve problems. BASIC II aims to teach students to understand the knowledge and skills needed for advanced structured BASIC programming .

The overall objective of Pascal I is to understand the use of Pascal in structured problem solving. The topics to be studied include: Basic Concepts (problem analysis, structure and format and the local operating system),

Standard Scalar Data Types and Operators (proper selection of data type with appropriate operator), Input and Output (the means to interact with the computer system), Pascal Programming Statements (the use of assignment, interactive and conditional statements), Subprograms (modular program development), One-Dimensional Arrays (the arrangement of data for use in programs) and Applications (practice in programming). Pascal I is a prerequisite for Pascal II.

Pascal II continues instruction in Pascal with additional emphasis on structured problem-solving techniques. Its topics are: General Concepts (enhances Basic Concepts from Pascal I), Advanced Uses of Procedures and Functions (use of parameters, nesting and recursion), Nested Statements (methods for solving complex problems), Multidimensional Arrays (manipulation of matrices), Structured Data Types (use of record, set and file) and Advanced Applications (programming practice in sorting and searching techniques).

LOGO's objective is the use of LOGO in problem solving and lists the following topics: Introduction to Computers and Programming (background for the course), Operations and Characteristics of the Local Computer System (efficient use of computer systems), Introduction to Procedures (implementation and manipulation of procedures), Simple Data Types and Operations (beginning problem solving), Input/Output Operations (interacting with the computer through programming), Control Statements (use of computers

for decision-making repetitive tasks) and Integrated Applications (ability to apply previously learned concepts).

The objective of FORTRAN is for students to develop programs using an organized approach to solve mathematical and scientific problems. The topics to be taught are: Introduction to Computers and Programming (background), Operation and Characteristics of the Local Computer System (efficient use of the computer system), Simple Data Types and Operations (beginning problem solving), Input/Output Statements (interaction with computers through programming), Problem Analysis and Program Design (systematic development of problem solving abilities), Control Statements (use of computers for decision making), One-Dimensional Arrays (arrangement of related data for computer manipulation), Subprograms (applications of independent modules in effective programming) and Integrated Applications (use of previously learned concepts in a scientifically oriented environment).

2.21 Texas

Computer Science I and II are intended to prepare college bound computer science students, with Computer Science I serving as an introductory course for those students who are prospective computer science majors and a foundation for other students planning to enter fields which use computer technology. Both courses are equivalent

to first year university courses and emphasize programming methodology, algorithm selection and analysis and selection of data structures. Texas law requires that all school districts offer students at least one year of computer science and that all students in an advanced program must take one year of computer science (Texas Education Agency, 1986).

The objectives for both Computer Science I and II are: to master the use of a high-level language while creating solutions which are well structured and modular, to maintain readability, ease of debugging, program maintenance, utility and validity when writing programs, to understand the function of computer components and to address ethical, social and career issues in computing.

There are eight topics covered in Computer Science I. Topic One, Beginning Concepts Associated with Computer Science deals with planning problem solutions, use of current design methodology, coding programs with style and clarity, demonstrating skill in testing for program correctness, developing effective debugging strategies and developing adequate internal and external documentation. Topic Two, Beginning Concepts and Skills Associated With Programming Languages covers the use of block-oriented languages, correct use of identifiers, developing code which uses data, expressions, assignment statements and control structures correctly, effective use of predefined input and output, procedures and functions and proper annotating of code.

Topic Three, Beginning Concepts and Skills Associated With Data Types and Structures, covers developing program code using numeric, character, and Boolean data and use of strings and arrays. Topic four, Beginning Concepts and Skills Associated With Algorithms, examines solving programming problems using algorithms and designing and coding search algorithms to be used in solving information storage and retrieval problems. Topic Five, Beginning Concepts Associated With the Application of Programming, covers coding programs for text processing, simulation and modeling, data analysis, data management, system software, graphics and games.

Topic Six, Beginning Concepts Associated With Computer Systems, deals with identification of hardware components and the use of system software. The final topic, Beginning Concepts Associated With the Social Implications of Computers, covers the responsible use of computer systems and social issues concerning computers.

Computer Science II is made up of eight topics, the first of which, Concepts Associated With Programming Methodology, covers developing larger, well organized programs, developing further skill in testing programs, and performing analysis of design methodologies. Topic Two, Concepts and Skills Associated With Programming Languages, deals with developing flexibility with file input and output, comparing programming languages and developing skills with recursive procedures and algorithms.

Topic Three, Concepts and Skills Associated With Data Types and Structures, is concerned with coding using linear data structures, binary tree structures and the representation of data structures sequentially. Topic Four, Concepts and Skills Associated With Algorithms, covers using searches, sorts and manipulation of data structures. Topic Five, Concepts Associated With Numerical Algorithms, deals with programming solutions using approximation, statistical algorithms and the importance of numerical accuracy when designing algorithms.

Topic Six, Concepts Associated With the Applications of Computers, discusses coding solutions for text processing, simulation and modeling, data analysis, data management, system software, graphics and games. Topic Seven, Concepts Associated With Computer Systems, covers the functions of compilers and interpreters and the trade-offs made by computers among factors such as cost, storage, speed, etc.

The Frisco (Texas) Independent School District advocates a comprehensive computer education program beginning in kindergarten and culminating in elective secondary courses in BASIC and Pascal. Plans are underway to move BASIC to a lower grade level and implement another programming language in high school in order to remain current with curriculum developments elsewhere (Berry and Kramer, 1984).

Texas educators such as Dr. Barbara Sadowski (1984) at the University of Houston are recommending that structured

languages such as FORTH, LISP and COBOL (Common Business Oriented Language) be introduced in high schools in which students receive extensive BASIC and Pascal instruction in the earlier grades.

2.22 Vermont

The Vermont State Department of Education has recommended a sweeping program of computer studies which begins in kindergarten and teaches rudimentary BASIC programing as early as second grade. High school, students are to receive instruction in BASIC, Pascal, FORTRAN and assembly language, which, if achieved, would place Vermont schools at the forefront of teaching programming (Lengel, 1983).

2.23 Wisconsin

Wisconsin has released a set of suggested guidelines for computer science courses offered in the state's high schools. These guidelines distinguish between computer science and computer literacy (literacy involves less programming and the coverage of many, more general topics). The overall goals of computer science are to emphasize problem solving strategies, the study of algorithms and structured programming. The specific goals which the courses are aimed at are: to learn and use computer languages for problem solving, to master basic programming techniques, to learn applications of problem

solving languages, to become aware of societal goals and issues concerning computers and to learn about career opportunities in the field of computers. The Department of Public Instruction states that studying a high-level language for its own sake is not the major goal of the curriculum, but rather to use a programming language for problem solving in a variety of areas. The courses offered in high schools use either BASIC or Pascal as their programming language, although FORTRAN, COBOL and machine language may also be used to a lesser degree (Wisconsin Department of Public Instruction, 1986).

At Chetek High School, four courses are offered in computer science, the final two (Computer Science III and IV) dealing extensively with programming. These two courses are aimed at students who plan to enter the computer field and, while BASIC programming is taught, Pascal and FORTRAN are also introduced (as Bork advocated) (Adams, 1983).

2.24 Secondary Programming Courses Outside North America

Unfortunately, very little information is readily available on programming courses outside the U.S. and Canada. Most of the literature which is available deals with the general introduction of computers in the schools and financial considerations. In West Germany (where educational objectives are determined by the schools themselves) computer courses are optional and a part of the

mathematics curriculum. BASIC is widely used, although Pascal is beginning to become popular. In general, teacher training is termed unsatisfactory and efforts are being made to improve teacher training courses in computers (Bosler, 1985). In Denmark, where the national ministry of education determines educational goals, COMAL has entirely replaced BASIC in the schools, because COMAL was developed in Denmark and combines many desirable features of both BASIC and Pascal (Keil, 1982).

In Sweden, it was decided to integrate computers into the existing curriculum rather than introducing a new subject in the schools. This was partly for financial reasons, but also because the need for programming instruction was questioned widely by Swedish educators. The programming which is taught is mostly limited to mathematics courses (Kollerbaur, 1983).

Malaysia is attempting to introduce a computer program in the schools which includes, but is not centered on, programming in FORTRAN, ALGOL or BASIC. Peng (1982) states that it is believed that students' grasp of mathematical concepts will be enhanced through programming instruction. Malaysia has embarked on a comprehensive one hundred hour computer appreciation course for secondary teachers. However, computer studies has not yet been introduced in Malaysia due to a lack of funds and curriculum development problems.

The major push in Britain was to get computers into the schools (through programs such as the Microcomputer Education Programme (MEP)) and to train teachers to use computers in their subject areas. Where programming courses are offered, emphasis is placed on development of problem solving techniques and structured programming with Pascal being the preferred programming language (Woodhouse, 1983).

2.2.5 Conclusion

Many of the American States surveyed offer dedicated programming courses which rely on Pascal or BASIC and some offer courses in more than one programming language at the secondary level. The majority of States place major emphasis on structured programming and problem-solving approaches. A few States offer a computer science program in which students advance from BASIC to other high- and low-level languages and gain the widest possible exposure to computer programming. Unfortunately, not enough information was available for this study to draw overall conclusions about courses offered in Europe or Asia.

3. Canadian Initiatives

3.1 Introduction

This chapter examines the programming courses offered by most Canadian provinces. New Brunswick and the Northwest Territories did not reply to requests for information. It should be noted that in Canada control over education is mainly a provincial matter as long as minority or second language education is not involved (Ollivier, 1962).

3.2 Alberta

Alberta's secondary computer courses are titled Computer Processing 10, 20 and 30 and are classed as business education courses. Although these courses are classified as business courses, all Alberta students receive extensive computer literacy instruction in junior high school. There are eighteen common components to all three courses: a required core topic and seventeen elective topics (Alberta Education, 1985).

Core Topic
1. Computer Information Systems

Software Topics
2. Overview of Software
3. Applications: Data Entry
4. Applications: Word Processing
5. Applications: Simulations
6. Applications: Data Base
7. Applications: Electronic Spread Sheet

Introduction to Programming Language Topics	8. Overview of Introductory Programming Language 9. Fundamentals of Input/Output 10. Introduction to Advanced Computer Programming Techniques 11. Advanced Computer Programming Techniques 12. Extended Programming Project.
Second Programming Language Topics	13. Introduction to Second Programming Language 14. Applications in Second Programming Language 15. Extended Project in Second Programming Language
Specialized Programming Functions Topics	16. Graphics 17. Systems analysis and Program Development 18. Machine/Assembly Language.

The objective of Module 1: Computer Information Processing is to make students aware of the terminology and concepts of information processing systems. It includes the following topics: Evolution of Data Processing, Terminology, Characteristics of Information Systems, Information Processing Environment, Data Communications and Issues and Trends.

The objective of Module 2: Overview of Software is to demonstrate and use commercial software packages. Its topics are: Terminology, Legal Constraints to Use of Software, Proper Handling Techniques, Types of Software, Software Applications and Software-Use Procedure.

The objective of Module 3: Applications: Data Entry is to enable students to develop skill in data entry, data manipulation and report generation. Its topics are: Data Entry Functions, Coding Data, Entry and Generating Reports.

The objective of Module 4: Applications: Word Processing is to enable students to learn procedures of word processing. Its topics are: Introduction to Information Processing, Software Commands and Input/Output.

The objective of Module 5: Applications: Computer Simulations is to teach students to use the computer and appropriate software in problem-solving. Its topics are: Define Simulation, Focus on Topics Within Subject Area and Evaluate the Simulation.

The objective of Module 6: Applications: Data Base is to develop understanding of and ability to use data bases. Its topics include: Management of Information, Parts of Data Bases, Clarification of Data Bases, Use of Data Base Management Software, Data Integration, Data Transfer, Telecommunications and Commercial Data Bases.

The objective of Module 7: Applications: Electronic Spread Sheet is to develop understanding of and skill in

using an electronic spread sheet. The topics covered in this module are: Management of Information, Use of Spread Sheets and Spread Sheet Applications.

The objective of Module 8: Overview of Introductory Programming Language is to reinforce problem solving techniques through development of structured programming skills. Its topics include: Terminology (use of computer related terms), Hardware (parts of computer system and their use), Software (use of program and system commands), Programming Techniques (designing, editing and running programs, controlling output, looping and branching commands, manipulation of variables) and Programming Applications (coding, debugging, running and documenting programs).

The objective of Module 9: Fundamentals of Input/Output is to develop understanding of data communication networks and additional programming skills using arrays and functions. Its topics are: Input/Output Processes (differences between batch and interactive processing), Processing of Data (how the CPU manipulates and stores data), Arrays (use of subscripted variables, table handling and searching techniques), Library and Operating System Functions, Programming Applications (flowcharting and writing programs to solve problems) and Data Communications (networks).

The objective of Module 10: Introduction to Advanced Computer Programming Techniques is to further develop

programming techniques and refine skills in graphics programming. Its topics are: Numbering Systems (decimal, binary and hexadecimal), Professional Programming Techniques (use of structured programming techniques, error trapping routines and development of documentation writing skills), Subroutines (hardware-dependent and user defined subroutines and their application in problem-solving), Multiple Dimension Arrays (organization of information), Introduction to Graphics (hi-res and low-res graphics, enhancing programs graphically and design of graphical output).

The objective of Module 11: Advanced Computer Programming Techniques is to develop advanced programming skills through file manipulation techniques. Its topics are Types of Files (identification of various data structures) and File Handling Procedures (manipulation of information using various types of files).

The objective of Module 12: Extended Programming Project is to allow students to increase their programming skills through a project. The four topics included in this module are: Develop a Plan, Prepare a Program (using "user-friendly" techniques), Hardware and Software Requirements and Evaluate the Program (preparation of an evaluation summary and recommendations for improving the final program).

The objective of Module 13: Introduction to Second Programming Language is to provide students with an

opportunity to study a second high level language (COBOL, Fortran, C, Pascal and COMAL are recommended choices). Its topics include: Comparison of Languages, Hardware/Firmware Specifications, Language Syntax, Applications of the Language (writing programs), Modifications of Programs (editing and modifying both student-written and existing programs) and Output (basic output formatting skills).

The objective of Module 14: Application in Second Programming Language is to encourage students to gain understanding of programming syntax by applying the language in problem-solving. Its topics are: Application of Language, Modify and Debug Programs, Language syntax (contrasting syntax with that of the previously learned language), Program Documentation and Format Output.

The objective of Module 15: Extended Project in Second Programming Language is to allow students to demonstrate skill in programming in a second language. Its aims are: Develop a Plan, Document the Process, Program Applications (reflecting "user-friendliness"), Hardware and Software Requirements and Evaluating the Program (using an evaluation summary with recommendations for improvements).

The objective of Module 16: Graphics is to provide opportunities to enhance programming skills by integrating sound and graphics routines into the programs. The topics covered in this unit are: Programming Applications (graphics and use of graphics and text windows), Graphics Modes (screen resolution, string graphics and use of video

display sheets), Animation, Color Commands and Sound Commands.

The objective of Module 17: Systems Analysis and Program Development is to understand system analysis and solve problems using a systematic approach. Topics covered in this module are: Systems Problem-Solving, System Investigation, System Analysis, System Design (using flowcharts), Programming Techniques, Program Documentation, Systems Implementation and System Evaluation and Maintenance.

The objective of Module 18: Machine/Assembly Language is to utilize machine and assembly language. Topics covered in this module are: Comparison of Language, Hardware Configuration, Memory Map, (modifying existing programs), Storage Location (construct low-level programs in the correct format using operation, input/output and arithmetical/logical instructions), Assembly Language Syntax (writing and running application subroutines), Program Modification and Program Applications (identifying suitable applications for low-level languages and demonstrating ability to use an assembler).

3.3 British Columbia

The province of British Columbia offers a one hundred hour course titled Computer Science 12 which continues and expands on Computer Science 11. The main emphasis of Computer Science 12 is to acquaint students with the

structured-design method of solving problems. This involves using a top-down approach which necessitates the use of a structured programming language (Pascal). The goals of this course are: to enhance students' thought processes and problem solving skills, to develop understanding of structured design, to apply structured design, programming and communication skills in a team programming environment, to develop a working knowledge of programming techniques used in file management and to extend students' understanding of computer system software. The five strands of the course and suggested time frames are: 1. Structured Design (12 hours), 2. Operating Systems (10 hours), 3. Computer Science Techniques (12 hours), Programming (46 hours) and Student Team Project (20 hours) (British Columbia Ministry of Education, 1985).

The learning outcomes for Structured Design are: Problem Definition (producing a written definition of a problem and determining desired output, defining data structures, drawing structure charts, use of pseudocode, completion of input-process-output chart) and System Development and Implementation (determining order for development of program modules, developing and testing modules, documentation).

The learning outcomes for Computer Science Techniques are: Sorting (bubble, shell and other sorts), Searching (linear and binary search routines), Indexing (key-indexed

and non-indexed files) and Debugging (structured "walk-throughs", trace program operation, debug individual modules, verify output).

The learning outcomes for Programming are: Program Structure (identify and use major components of programs), Program Style (formatting of statements and commenting to increase program clarity), Data Structures (identification and use of data structures, operators and functions), Control Structures (sequence, selection, repetition), Modular Structure (structure and IPO charts, nesting of procedures and functions) and File Handling (typed and untyped files, file access statements, file I/O statements).

The learning outcomes for Student Team Project are: Team Project Environment and Related Concepts ("egoless programming", structure of programming teams and responsibilities of members).

3.4 Manitoba

Manitoba offers a course entitled Computer Science 305 which has the following goals: to recognize the basic features and functions of computers, to develop problem-solving techniques, to use computers as problem-solving tools, to utilize structured programming techniques, to understand the operation of the computer as it uses pseudo-machine and pseudo-assembly language programs, to understand the relationship between

pseudo-machine, pseudo-assembly and high-level languages, to prepare and execute structured programs using a high-level language, to develop skill in detecting and correcting program errors and to use accepted programming techniques. The ten units which make up the course are: I Review, II Subprograms (20 percent), III High-Level Languages, IV Team Programming Project, V Character Manipulation, VI Boolean Arithmetic, VII Simulations, VIII System Software and language Translation, IX File Manipulation and X Numerical Applications. It is recommended that at least two high-level languages chosen from BASIC, Pascal, FORTRAN and COBOL be used in Computer Science 305 (Manitoba Department of Education, 1983).

The objectives of Unit I, Review, are: A. Debugging (detect and correct syntax and logic errors) and B. Characteristics of a Good Program (identify and use good programming techniques).

The objective of Unit II, Subprograms, is to make the student able to design and apply subprograms.

The objectives of Unit III, High Level Languages, are: to identify and use statements for documentation, assignment, output, input and termination; to use conditional statements, loops, formatted input and output, CASE statements, simple data structures; and to create, manipulate and retrieve files.

The objectives of Unit IV, Team Programming Project, are: to participate in a team project involving selection

and definition of a task, division of responsibilities, segmenting of program designing and coding, structured walk-throughs, documentation and testing and preparation of a final report.

The objectives of Unit V,Character Manipulation, are: use ASCII and ASCII commands, solve problems using character control commands, handle input characters correctly, handle character output and manipulate data which is internally represented.

The objectives of Unit VI, Bbolean Arithmetic, are: A. Number Representation (convert back and forth from decimal to binary, decimal to Binary Coded Decimal, binary to octal and binary to hexadecimal, solve basic binary arithmetic problems), B. Internal Representation in Integers (convert from integers to internal representation and vice versa, perform simple operations using complement arithmetic), C. Sets (define and use terms used in set theory), D. Logical Operations (define with truth tables negatives, conjunction, disjunction, implication, exclusive disjunction) and E. Basic Digital Logic (draw circuit diagrams, describe similarities between circuits and truth tables, describe NAND and NOR gates with truth tables, draw circuit diagram for a logic expression).

The objectives of Unit VII, Simulations, are: define with examples the term simulation, identify two main types of simulation (deterministic and probabilistic), define random and pseudorandom numbers, identify number-generation

techniques and their strengths and weaknesses, list various tests for randomness, write simple simulations and identify real-time simulations.

The objectives of Unit VIII, System Software and Language Translation, are: A. System Software (define and describe system software, operating system, language translators, high-level languages, editor and loader) and B. Language Translation (describe job of language translator, name and identify merits of two types of language translation, describe interpretation and compilation, define terms related to compilers).

The objectives of Unit IX, File Manipulation, are: create and retrieve sequential and random access files, describe various searching and sorting techniques and merge two files.

The objectives of Unit X, Numerical Applications, are: to define the terms numerical calculation, truncation error and round-off error and to develop computer solutions for various numerical applications.

3.5 Newfoundland

The province of Newfoundland offers a single elective course titled Computer Studies 2206, which is essentially a literacy course with some programming included. The purpose of this course is: to provide students with a functional literacy of computer technology, to enable them to recognize the effect of computers on our everyday lives,

to utilize the potential of computers and to contribute to its continued development. The objectives of the course are to: learn the principles of the operation, capabilities, history and applications of computers, to learn elementary programming skills using BASIC, to be aware of careers in the computer field and to appreciate the computer's role and influence in society (Newfoundland and Labrador Department of Education, 1982a and 1982b).

The stated broad objectives of the course are to enable students to develop and exercise creative thinking, reasoning and problem solving skills. The second objective is to be met by teaching programming, which is to be taught according to student abilities and desired outcomes. The course content and time frame is as follows: Module I (Programming in BASIC) 17 weeks, Module II (Computer Systems; Operation) 4 weeks, Module III (Computer Systems: Equipment) 4 weeks, Module IV (History of Computers) 1 week, Module V (Applications of Computer Technology) 4 weeks and Module VI (Careers in Computer Technology) 1 week. Although programming takes fifty percent of the class time, it is to count for forty percent of a student's final evaluation. The department states that an essential component of Module I is a consideration of the program development process, which includes: analysis (defining and analyzing problems through block diagrams and flowcharts), program coding (translating algorithms into BASIC programs), program evaluation (debugging) and program documentation.

3.6 Nova Scotia

Nova Scotia's Computer Related Studies 441 course is a Grade 12 academic course which was introduced in September 1986. Before this date teachers of Computer Related Studies courses had great latitude in their choice of topics, texts and courseware. The new curriculum guide is an attempt to standardize the topics covered and provide additional resource materials for teachers . The other Nova Scotia Computer Related Studies course, CRS331, is a Grade Eleven computer literacy course and therefore is not discussed here (Nova Scotia Department of Education, 1986).

The objectives of Computer Related Studies 441 are: to develop familiarity with microcomputers through hands-on experiences, to develop skill in structured programming using one computer language, to study the components of computer systems, to use word processor, spreadsheet and database software, to identify career opportunities in the computer field and to gain an understanding of the legal aspects of computer use, including copyright, privacy and software piracy. The six units of the course and the time allotted to each are:

TOPIC	%CLASS TIME	%HANDS ON	%CRS TIME
1. Word Processing	4%	6%	10%
2. Structured Programming	20%	20%	40%
3. Spreadsheets	5%	8%	13%
4. Computer Components	14%	6%	20%
5. File Management	5%	8%	13%
6. Careers	2%	2%	4%

The objective of Unit One, General Introduction Through the Word Processor, is to experience an overview of a word processing system. The skills developed in this unit are: loading programs, retrieve and save files, format diskettes, back up data files, use special print functions, care for and handle diskettes properly, feel at ease using software packages, appreciate how computers can change the way certain tasks are done, use good keyboarding skills and use spell checker programs.

Unit Two, Structured Programming, has as its objective to provide students with an opportunity to solve real-world problems by planning, coding and testing programs of their own. Problem solving is to be stressed and should overshadow the specifics of coding programs using a particular language. The four steps to be used in problem-solving are: understand the problem, devise a plan, try the plan and check the solution. The following are classed as essential to structured programming: Working Environment (use and care of equipment, entering programs, editing programs, compiling programs, running programs, managing program files on a disk, merging programs, checking available disk and main memory space), Program Development (top-down design, structured approach to program design, planning, debugging/testing, determining and solving errors), Documentation (internal, external), Input/Output (interactive programming, screen and printer displays, use of menus, special I/O devices such as paddles

and mice, disk I/O), Variables (proper use of variable names, numeric variables, alphanumeric variables, Boolean variables, subscripted variables), Flow of Control (Looping) Structures (use of iterative, conditional, selection control structures), Procedures, Functions and Subroutines (subdividing problems, modular approach to programming, avoidance of GOTO statement, utility libraries), Data Manipulation (sorting, searching), Building Utilities (build, store and refine helpful procedures and file handling routines), File Handling (create, input and output a sequential data file, random access files, managing disk storage space) and Special Functions (random number generators, graphic displays, trigonometric and other numeric functions, sound generation). A team programming project which involves the planning, design and implementation of a major program is also to be part of this unit.

The goal of Unit Three, Electronic Spreadsheet, is to develop awareness of the nature of electronic spreadsheets. The specific objectives of this unit are: Prepare Data Disk, Load and Run Spreadsheet, Move Through Document Using Cursor Keys and by Naming Cells; Entering of Values, Text and Formulae; Editing and Altering Data in Single Cells; Saving, Retrieving and Printing Worksheets; Reproducing Cells; Definition and use of Templates; and solving Algebraic Equations Using Spreadsheets.

The objective of Unit Four, Computer Components and Communications, is to study the computer system's components in detail. The skills to be developed in this unit are: select appropriate computer for given task, select appropriate input devices, select appropriate output devices, select appropriate storage devices, backup data files and transfer data between microcomputers using a modem.

Unit Five, File Management Systems, has as its objective to experience an overview of a file management system. The skills developed in this unit are: load program, retrieve and save files, format diskettes, use special print reporting functions, care and handling of diskettes, be at ease using software packages, develop structure for given data, sort data alphabetically or numerically, search data using several criteria and appreciate possibilities which computers offer in this field.

The final unit, Careers, aims to identify employment opportunities related to computers. The four goals of this unit are: develop overview of traditional computer occupations, develop overview of applications of computers in other occupations, illustrate use of computer when making career plans and develop appropriate work skills.

3.7 Ontario

Ontario offers eleven different computer science

courses at the secondary level: three Grade Ten courses in Introductory computer Studies, four Grade Eleven courses (Data Processing Techniques, Computer Science and Technology, Computer Technology and Data Processing Concepts) and five Grade Twelve courses (Systems Analysis and Design, Computer Science, Computer Technology (Advanced), Computer Technology (General) and Data Processing Applications) (Ontario Ministry of Education, 1983).

The Computer Science course is designed to attract those grade 12 students who show promise in programming and problem-solving and is therefore the course examined by this study. The nine objectives of this course are: Advanced Programming (10 percent), Planning logically (10 percent), Developing Diverse Applications (50 percent), Impact of Computer Science (5 percent), Converting Program Code (5 percent), Computer Architecture (5 percent),Low-Level Languages (10 percent), Careers and Further Studies (2 percent) and Personal Communication Skills (3 percent).

In advanced programming, students learn advanced techniques which extend their knowledge of programming by utilizing the available constructs of the language being studied. They should be able to: manipulate character strings using built-in functions, use available subprogram features, apply user defined function capabilities of a programming language, use single and multiple-dimensioned

arrays and choose and use the most effective variable for any given problem.

Under the heading of Planning Logically, students develop and implement algorithms and should be able to: write data sorting programs, demonstrate knowledge of sorting techniques, understand sequential and binary searches, write list merging programs, understand how random number generators are created, create and access sequential and random-access files, linked lists and tree structures.

Developing Diverse Applications aims to have students expand their knowledge by using advanced programming techniques to solve a variety of problems, including the following: textual manipulation, three-dimensional graphics and animation, solving problems such as areas under curves by using interactive methods, simulations, statistical processes and any additional concepts which may arise from developing technology.

The Impact of Computer Science objective aims to teach students the impact of programming applications on society, so that they may: identify social implications, examine ways in which computers might be used to assist special groups of people and design and implement software and/or hardware solutions to the problems they have identified.

Converting Program Code refers to the understanding of the characteristics of operating system modules. Students should be able to: describe system software, describe the

operation of a multi-pass compiler and correctly use terms related to this area.

Computer Architecture refers to an understanding of a computer's internal organization and an ability to synthesize: the various addressing techniques, memory organization, the flow of data through a computer bus and the interrelationships among computer components made possible by the computer bus system.

The Low-Level Language objective is satisfied by having students use a subset of the instructions of a low-level language to solve simple problems including: loading and storing accumulators, incrementing registers, using logic operations and using various addressing techniques.

Careers and Further Studies aims to inform students of possible careers in the computer field and training which is available. Personal Communication Skills tries to provide students with opportunities to develop communications abilities by classroom interaction and formal research and reports.

Students in this course are expected to produce at least one program of significance during the year, and languages used include BASIC, Pascal, COBOL, FORTRAN and APL, as well as some low-level languages.

3.8 Prince Edward Island

Prince Edward Island offers a single computer literacy course (Computer Literacy 721) which lists as its main

strands: flowcharting and programming, about computers and computers in society. The flowcharting and programming strand has the following objectives: flowcharting (write an algorithm and draw it in flowchart form), BASIC commands (writing BASIC programs), corrections (finding and correcting program errors), utilization (use microcomputers to enter, run and edit programs), peripherals (use of computer peripherals such as disk drives and printers), applications (word processors and spreadsheets) and data bank (accessing bulletin boards and data banks) (Prince Edward Island Department of Education, 1984).

Teacher training has been implemented through a series of inservices held across the province. It is recommended (but not required) that students enrolling in Computer Literacy 721 have a strong background in mathematics with an optimum student to computer ratio of 2:1.

3.9 Quebec

Quebec offers two optional half-year computer science courses (entitled Introduction to Computer Science) for Secondary IV and V students. The first course serves as a prerequisite for the second although students are free to take only the first course. The six general objectives for these courses are: Construct Program Algorithms for Solving Problems (30 percent), Develop Problem Solving Strategies (30 percent), Program Computers (30 percent), Know the Fundamental Structure and Components of a Computer System

(4 percent), Evaluate the Role and Place of Computers in Society (3 percent and Develop Critical Attitudes Toward Computers and Information Processing (3 percent) (Ministiere de l'Education, Province du Quebec, 1984).

The first objective, Construct Program Algorithms for Solving Problems, has the following intermediate objectives: write a proposal for a project, reformulate the statement of the project, organize the project, describe the problem-solving process, produce an algorithm, code the algorithm, run the program, revise and refine the program and produce a report on the project.

The second objective, Develop Problem Solving Strategies, has the following intermediate objectives: select information for a problem, find a simpler problem which is part of the larger problem, find a problem isomorphic to the problem, generalize the problem, subdivide the problem, analyze the problem, find analogies to the problem, use the deductive process, define, formalize, describe the problem and its solution, destructure, test the solution's validity, classify the problem and its solution, explain the problem and its solution, proceed by trial and error, use the iterative process, use the recursive process, select a subset of the preceding strategies and use the modelling process.

The third objective, Program Computers, has the following intermediate objectives: use constraints; use variables; use arithmetic, Boolean and alphanumeric

operators; construct arithmetic and alphanumeric expressions; use assignment instructions; use display instructions; use data entry instructions; use conditional and unconditional branching instructions; use predefined functions of the selected language; use iterative and recursive structures; use subprograms; use files; use system commands; produce graphics; produce music; and use instructions to enable computer to control external devices.

The fourth objective, Know the Fundamental Structure and Components of a Computer System, has the following intermediate objectives: describe how the computer works, classify computers, identify peripherals, use control commands for computer and peripherals, identify types of software, identify types of codes used by computers, master binary numbers and their derivatives and know the history of computers.

The fifth objective, Evaluate the Role and Place of Computers in Society, has the following intermediate objectives: describe computer applications, explain the computer as tool, evaluate the computer's impact on individual rights and freedoms, identify effects of computer systems on the job market, evaluate possible computer careers.

The sixth objective, Develop Critical Attitudes Toward Computers in Society, has the following intermediate objectives: evaluate the true nature of the computer and

information processing, react appropriately to computers, evaluate activities related to information processing and develop self-confidence and self-esteem by planning and writing computer programs.

3.10 Saskatchewan

Saskatchewan offers two computer literacy courses (Computer Applications 10 and 20) and three programming courses (Computer Science 10, 20 and 30). Computer Science 30 is intended for students who have either completed Computer Science 10 and 20 (half-year courses offered at the Grade Ten and Eleven levels) or their equivalent and thus possess a solid foundation in BASIC (Saskatchewan Department of Education, 1984).

Pascal is the recommended programming language for Computer Science 30, and teachers are advised that students will require two to three hours per week of computer time outside class to complete the requirements of the course. A major programming project is to be assigned to all students taking the course to give them the opportunity to synthesize the components of their programming experiences.

The stated objectives of the course are that the student should learn general problem-solving techniques and be able to: clearly define problems; design and code algorithms; hand trace algorithms and code; test and debug programs; anticipate, isolate and correct errors; and discuss the power and limitations of a high-level structured language with regard to data and control

structures. The major topics to be covered in the course are: I. Transition From BASIC to Pascal, II. Extending the Fundamentals and III. Optional.

Topic number one, Transition From BASIC to Pascal, includes the following subtopics: Compiled Versus Interpreted Languages, Structure, Language Details and Data Types. The transition from an interpreted language (BASIC) to a compiled language (Pascal) is to be accompanied by an examination of compilers and also by some discussion of structured versus non-structured programming languages. Pascal's structure and syntax are to be examined with some discussion of procedures (one of Pascal's strengths) taking place at this point in the course. Data types to be dealt with include scalar, string, array and user defined. The total class time to be allotted for topic number 1 is twenty hours.

Unit II. Extending the Fundamentals, further develops the topics discussed in Unit I under the following headings: A. Procedures and Parameter Passing, B. Functions, C. Data Structures, D. Searching and Sorting Techniques, E. Recursion Versus Iteration, F. Text Files, G. A Closer look at the Hardware and H. Algorithms. The treatment of procedures widens to include parameter passing, local and global variables and calls by value and reference. Functions are introduced at this point in the course with data structures a major focus of this unit. Data structures covered include: simple variables, arrays,

set, user defined types, simple records, character files, packed arrays, stacks and queues. Under searching and sorting techniques linear and binary searches are covered, along with an introduction to sorts (selection and bubble) and several intermediate sorting techniques such as merge, insertion, quick and shell sorts.

A Closer Look at the Hardware includes information on computer architecture, accessing memory, machine language programming and assembly language programming. The Algorithms section covers topics including design methodology for problem solving, development of algorithms (selection of data and control structures and the designing of programming modules), applications of high level languages to a variety of representative and classical problems, the use of counters and flags, manipulation of records and arrays, operations with strings, sorts, searches, merges and turtle graphics.

The optional unit deals with such topics as robotics and artificial intelligence which are at the forefront of computing and will give students an opportunity to pursue areas of special interest.

3.11 Conclusion

Approximately half of the Provinces surveyed offer dedicated programming courses which rely on Pascal or BASIC with only Manitoba and Ontario offering courses in more than one programming language at the secondary level. Five

of the Provinces place major emphasis on structured programming and problem-solving approaches. Five Provinces place importance on a major programming project in order to gain the widest possible experience in computer programming. Most Canadian provinces appear to be very much up to date in secondary computer programming courses.

4. CONCLUSIONS

4.1 Introduction

This chapter draws comparisons between the Nova Scotia CRS 441 course and secondary programming courses offered elsewhere. An attempt is made to point out strengths and weaknesses of our Nova Scotia course in hopes of improving the course. It should be noted that only ten States are listed in the accompanying charts (Appendix II). Many of the States which supplied information did not provide sufficient detail about their programs to be included in the charts.

4.2 Overall Goals

An examination of Appendix I reveals that Nova Scotia fares rather well when compared to other provinces and various American States. The reader should keep in mind that information on United States courses is based on suggested courses and that the actual courses may vary widely from those discussed here. Appendix II compares the major aims of the programming courses and lists seven areas of comparison: Problem Solving, Top-Down Design, Programming Languages, Team Programming, Structured Programming, Programming Stressed and Major (Programming) Project.

All courses examined listed **Problem Solving** as a major goal or objective. This is due to the emphasis placed on problem-solving in mathematics and the fact that computer programming courses are seen in many areas as extensions of mathematics course offerings. The usual method of teaching programming is through progressively more difficult problems which utilize concepts taught in class. This requires the student to apply problem-solving techniques and reduce the problem to a series of subproblems which are more easily solvable by the student.

The second major goal, **Top-Down Design**, refers to the careful setting of goals when programming and the breaking down of those goals into subgoals which are easily achievable. Nova Scotia is one of only four provinces in Canada which places stress on the principles of top-down design. This speaks well of the intention of the Department of Education in this province to teach good programming technique in CRS 441. While elementary problems can be solved without the application of top-down design, more involved programming projects are nearly impossible to tackle without the use of this method of problem-solving.

Nova Scotia is lacking in the area of **Programming Languages** taught, because while BASIC or Pascal satisfy the requirements of CRS 441, almost all Nova Scotia schools use only BASIC as the programming environment of choice. This may be due to the lack of available Pascal software,

the preferences of teachers who, for the most part, completed their programming courses in university before Pascal appeared on the educational scene or to simple inertia on the part of teachers who developed their CRS courses from scratch and are hesitant to move into unfamiliar territory. I am not advocating that schools in this province begin teaching more than one programming language in CRS because it appears that in the few areas where this is done students have had extensive experience with BASIC before they ever reach high school. However, Nova Scotia students entering university computer science courses are finding Pascal and FORTRAN courses very difficult because they have no training in a structured programming environment. This is not to say that Pascal is by any means the perfect structured language, but it does possess some structured elements in its microcomputer implementations. Some educators have stated that structured BASIC would serve as well or better than Pascal, but structured BASIC is even more rare in this province than Pascal. I am sure that a survey of Nova Scotia high schools would reveal that the vast majority are using outmoded versions of BASIC such as Pet BASIC, Applesoft BASIC and TRS-80 BASIC and that schools are only beginning to implement more modern implementations of the language such as BASICA for the IBM PC. The Department of Education is understandably hesitant to set a software standard for the CRS course due to the diversity of hardware being used

in Nova Scotia high schools, some of which is unable to run any language but BASIC (due to memory limitations, obsolescence, etc.). However, we are beginning to reach a hardware standard (IBM) and should now begin to make some move toward a software standard which will prepare students for real world computing environments.

Team Programming, major objective number four, is found in the computer curricula of only two provinces (Manitoba and Nova Scotia). The team approach is seen as a way of encouraging students to adopt "egoless programming" (which holds that all members of the team are equally responsible for the success or failure of the project). It is also hoped that students will become willing to experiment because of a decreased fear of individual failure placed on them in a group environment. This approach is widely used in computer courses everywhere (whenever there are more students than microcomputers) with results which vary widely due mainly to the makeup of each group. Nova Scotia is to be commended for this attempt to expand the programing horizons for its CRS students by making team programming a major part of the Grade 12 course.

The fifth overall goal, **Structured Programming**, is universally recognized in the literature as the correct path to follow when teaching programming. Surprisingly, only five Canadian provinces specifically mention structured programming as a goal for their course.

Structured programming refers to a generally organized method of writing a program utilizing subprograms and certain recognized programming practices such as: the avoidance of GOTO statements in BASIC; formatting of programs by indenting; and keeping code clear, readable and logical. The phrase "structured programming" has become almost a cliche in computing literature and is often proposed as an antidote for "spaghetti code" (hopelessly tangled program logic). The Nova Scotia Department of Education has wisely determined that this concept needs to become a major goal of the CRS 441 course in order that students who continue on in the computer field will become accustomed to programming in this way.

The next major objective, **Programming Stressed**, refers to whether or not programming is the main focus of the computer science course. Nova Scotia's course does not stress programming (only 40% of class time is spent on programming), but, since there is no prerequisite for CRS 441, some students would experience a great deal of confusion and mental strain if a course dealing exclusively with programming was offered at the Grade 12 level. Only four Canadian provinces offer courses dealing primarily with programming and all four offer more than one secondary computer science course. Computers have not filtered their way down to elementary and junior high schools in this province as they have in the U.S. due to budgetary restraints, lack of teacher training and the fact that

computers have been a major concern of educators in Nova Scotia only since 1980 (approximately). There has not been enough time to develop CAI throughout grade levels and curricula which would lead to demand for a course devoted to programming and to the development of students who are prepared for a rigorous course in programming.

The final major objective, **Major (Programming) Project,** is embraced by the Nova Scotia CRS course as a method of covering advanced topics which have not been previously examined and of having students demonstrate their programming skills. This is a very good exercise for students, who will have the opportunity to engage in the construction of a detailed, involved program which will give them some idea of what programming in the outside world is actually like.

4.3 Secondary Goals

There are twenty-four secondary goals listed for comparison in Appendix II. These goals are:

1. Applications
2. Graphics
3. Ethical and Social Issues
4. Computer Fundamentals
5. File Handling
6. Computer Careers
7. Program Debugging

8. Pseudocode
9. Flowcharts
10. History of Computers
11. I/O
12. Control Structures
13. Subprograms
14. Data Communication
15. Sort and Search
16. Interpret vs. Compile
17. Modularity
18. Second (High-Level) Language
19. Low-Level Language
20. Systems Analysis
21. Operating System
22. Arrays
23. Computer Architecture
24. Program Documentation

There are many other goals of computer programming courses, but the twenty-four listed here reoccur throughout the computer curricula and provide a basis for comparison.

Applications refers to the use of commercial software in computer courses, an area in which the new Nova Scotia course excels. Sections are included covering the three main types of commercial software: word processors, databases and spreadsheets. Most of the world's computer workers are users, not programmers and use of this type of software will prepare students who will be entering fields

which rely on computers extensively (e.g. secretarial). While the CRS 441 course is not intended to be vocational, it is difficult to name a profession which is not affected by computers and therefore applications are important (approximately 35% of class time is devoted to them). Within this topic, there is some programming done inside the application itself (especially in the database section).

Graphics is another area of interest in computer science courses and one which receives little or no attention in CRS 441. The Nova Scotia course concerns itself with the most commonly used applications (word processing, spreadsheet, database) and does not allow any time for the less common applications (i.e. graphics). While graphics are certainly an area of interest, I feel that the CRS 441 course wisely emphasizes the most important applications.

Ethical and Social Issues are stressed in the Nova Scotia course (and in those in most other areas) due to the computer anxiety which affects many people and fears that students will not receive correct information about software piracy and theft of data. Ethics have a part in all school subjects and especially in the relatively new field of computers. Students must be informed that theft of computer programs and/or data is exactly that- theft. The Nova Scotia computer curriculum guide states this concept explicitly in its introduction and the Department

is to be congratulated for taking a firm, clear stand on this issue.

Computer Fundamentals refers to a general introduction to computers including information on the operation of the computer system, care and use of hardware and software and execution of programs. This process takes time away from programming instruction, but, as stated earlier, since the CRS 441 course has no prerequisite, students taking the course often have no prior computer experience and must receive some introductory orientation. This component could be eliminated by having Nova Scotia's other CRS course (CRS 331-Computer Literacy) serve as a prerequisite for CRS 441.

Almost all provinces and States surveyed (including Nova Scotia) listed **File Handling** as an objective of their courses. This refers to the efficient storage, retrieval, searching, sorting and indexing of various file types and is a necessary component of any computer programming course.

Another very popular component of computer courses is a section devoted to **Computer Careers**. It is not sufficient to discuss programming in detail without providing at least a cursory discussion of what careers are open in the computer field. Nova Scotia's course devotes 4% of class time to this topic, which is covered in Unit I of CRS 441. Again because programming courses are not vocational courses, they are not designed to channel

students into careers and therefore, instruction about computer careers should be limited to a general orientation.

Program Debugging is a vital area of interest for all computer programming courses and Nova Scotia's CRS course makes debugging a focus of the Structured Programming unit. No computer program is ever without flaws and an intelligent programmer always checks his code under all possible conditions. Teaching programming without teaching program debugging would be analogous to teaching creative writing without teaching sentence construction.

Arrays form the basis for the computer's information processing capabilities and a programming course in any high-level language which omitted this critical area would be seriously flawed. Nova Scotia's CRS course again receives full marks in this area because of the large portion of time devoted to arrays.

Pseudocode is stressed in only three of the provinces and one of the States surveyed. This refers to the writing of a program outline in a mixture of English and a programming language and can serve as a valuable tool for program design. However, the Nova Scotia course cannot be faulted for the omission of this topic. An introductory course which is aimed at students whose abilities vary widely lacks the time required to build skill in using pseudocode.

It was rather surprising to note how few provinces and States include **Flowcharts** in their computer curricula. It seems that no one is indifferent to flowcharts, they are either loved or hated. I have always felt that flowcharts are very useful in explaining program flow especially to students who lack experience in programming. Flowcharts are mentioned in the Nova Scotia course's curriculum guide, but they are not a major focus of the course. They are listed in passing as a possible step in the problem-solving process.

The **History of Computers** is not an objective of CRS 441. This is a literacy topic and one which has little part in a programming course.

I/O describes a multitude of topics including methods of input and output, formatting of output, output routing, etc. No programming course can avoid an extensive discussion of I/O because programs and their results are basically input and output. Almost all provinces and States surveyed stressed this topic in their courses.

Control Structures refers to a language's facilities for specifying a departure from normal program execution (i.e. subroutines or procedures). All programming courses listed in Appendix II (Prince Edward Island's course is a literacy course) stress control structures in the language chosen. This is because the power of a programming language is revealed when control structures are studied. It is here that Pascal overshadows BASIC because of its

procedures which make Pascal programs far simpler to write, trace and debug.

Subprograms describes a modular approach to writing programs which is basic to structured programming. Most States and provinces surveyed (including Nova Scotia) stress the use of subprograms since subprograms (due to their modularity) can be reused in related problems and students are not forced to continually rewrite the same program.

Data Communication is an important concept which is not included in many programming courses. Nova Scotia's CRS course includes material on communication as part of Unit Five, File Management. Data transmission is becoming increasingly important especially in business environments, where rapid, error-free transmission of information is critical. The Nova Scotia Department of Education is to be commended on their foresight in including this topic in the CRS 441 course because of its importance and timeliness.

Sort and Search refers to the various methods of organizing and retrieving data. Because of the importance placed on databases in the CRS course, sorting and searching techniques play an important role. It is very possible that data handling will replace programming as the most important computer-based topic in the near future due to the information explosion now taking place and the Nova Scotia course prepares students for this eventuality.

Interpret vs. Compile is concerned with the relative merits of interpreted and compiled programming languages. Since the only language used in most Nova Scotia schools is BASIC (an interpreted language) and since no compiled languages are introduced, discussion of this topic is impossible. This is unfortunate because students who continue in programming will be exposed to mainly compiled languages. Since the CRS course is intended for a broad student audience with varied interests, this topic was omitted in order to develop areas which were judged more important to most students (i.e. applications).

Modularity is concerned with the approach that most problems can be reduced to smaller, repeatable sub-problems. While the CRS course utilizes principles of structured programming, modularity is not stressed as an important goal.

A **Second Programming Language** is not utilized in the Nova Scotia course because only 40% of class time is devoted to actual programing instruction and practice and there is not enough time available to move beyond the language used. This is unfortunate because many computer educators believe that by comparing two or more languages, students will gain additional expertise in coding in all languages studied.

Low-Level Languages are not covered in the CRS course because, as stated above, with only 40% of the course devoted to programming, there is simply no time available

to study assembler or machine language. Proponents of low-level languages believe that they help students gain insight into computer architecture and learn about program execution at its most basic level. With a course which spends so much of its available time on applications it is clear that the curriculum planners in this province decided that low-level languages were better left to university courses.

Systems Analysis refers to the overall planning, design, coding, implementation and testing of programs and is also not covered in the Nova Scotia course. Some small discussion of systems analysis may take place during Unit Seven, Programming Project, but this topic has also been left to higher-level courses.

Operating System is concerned with the use and understanding of the resident computer operating system, including I/O routing, control of peripherals, etc. The CRS course deals with programming within the environment of a high-level programming language and fails to discuss operating systems except as they relate to loading and executing programs.

Computer Architecture is concerned with the physical components of the computer system and the ways in which they interact. This is mentioned briefly in the Nova Scotia course but is discussed in a very cursory manner. Again, choices were made and topics such as this were left to university and vocational courses (a choice which I totally agree with).

Program Documentation, both internal (commenting) and external (user guide) are stressed in the Nova Scotia course because of the importance placed on writing programs in a structured manner and producing clear, readable output.

4.4 Specific Recommendations

In summary, Nova Scotia receives high marks in most critical areas except for the choice of programming language and the time devoted to programming. It could be that, as students become computer-literate earlier in their education, demand for a course dedicated to programming will persist. But these programming courses may be offered at more than one level and be hierarchical in nature in order to provide deeper understanding of programming skills.

There are several other problems with Nova Scotia's CRS courses beginning with the confusion over the two courses (CRS 331 and 441). CRS 331 is a literacy course and is intended for students who will not be taking CRS 441, the programming course. Consequently there is considerable overlap in the two courses. The problem which arises in Nova Scotia schools is that students interested in computers take both courses and find many CRS 441 topics a repeat of CRS 331. If a true programming course is to be taught to Grade 12 students, there should be a previous introductory course which would allow students to determine

if they wished to enroll in a programming course. No such course now exists since CRS 331 is not a prerequisite for CRS 441. In many of the provinces and States surveyed CRS 441 would be a computer literacy course, due to the large portion of time spent on applications. If we are to prepare students to take their place in a world which relies more heavily on computers every day I think we must expand and diversify our programming efforts.

The approved texts for CRS 441 are another problem for teachers of the course. The theory text is obsolete after only four years (hardly surprising in the rapidly changing world of computers) and the only programming text available deals with BASIC and fails to use structured programming principles. If schools are ever to successfully introduce Pascal, a teaching text is absolutely necessary. Finally, there are no certification standards for teachers of the CRS course and no training programs specifically designed for computer science teachers. The two CRS courses are usually taught by math teachers, many of whom lack programming skill themselves and fail to teach programming in the correct way.

We have come a long way in our struggle to modernize computer science courses in Nova Scotia but we still have a long distance to travel. We should standardize our software and hardware as soon as possible, choose adequate programming texts, introduce teacher training courses and attempt to stream students entering computer science

courses if we are to continue to improve our students' computing abilities.

Nova Scotia computer educators should be extremely careful to avoid offering an obsolete programming course which fails to meet the needs of students entering a technological workplace because of a weakness in programming instruction. Nearly all provinces and States surveyed have a high-technology programming course in place and we must either follow suit or fall even further behind in the race to prepare students for life in the next century. The total lack of teacher training in computer education in this province is a major problem which must be dealt with swiftly, and the need for a full-time computer consultant is also pressing. On February 26, 1987 the Speech from the Throne (Nova Scotia House of Assembly, 1987) stated that computer literacy is to be stressed in the province's educational system. This shows that the provincial government recognizes that we need to stress computer education. Now is the time to press forward with an in-depth course in programming for high school students because the longer we delay, the greater the gap which must be dealt with.

APPENDIX I

USE OF MICROCOMPUTERS IN SELECTED U.S STATES

State	Rank 1986	Rank 1985	Rank 1984	Units 1986	Units 1985	Units 1984	K-12 Students
Alaska	1	1	12	5,779	3,493	1,083	99,711
Arizona	19	13	21	11,752	9,117	4,077	527,332
Arkansas	42	27	34	7,275	6,172	3,065	436,489
California	43	44	32	70,888	42,430	30,033	4,331,876
Connecticut	10	8	25	11,845	8,815	3,731	465,359
Dist. Columbia	13	48	24	2,131	700	700	87,927
Florida	14	20	14	36,281	23,491	13,803	1,528,504
Georgia	48	49	47	15,321	7,737	4,128	1,151,359
Illinois	31	41	27	33,613	20,740	14,058	1,834,680
Indiana	12	12	18	24,251	17,766	8,860	975,918
Iowa	21	17	5	10,833	8,447	5,377	494,715
Kentucky	40	32	42	11,545	8,367	3,623	672,139
Louisiana	50	50	48	9,343	5,155	2,718	781,346
Maine	36	29	28	3,708	2,887	1,559	208,525
Maryland	35	34	43	11,945	8,643	3,508	670,647
Massachusetts	17	30	13	19,686	12,181	8,204	874,486
Mississippi	49	51	50	6,052	2,724	1,306	464,864
Missouri	45	37	39	12,899	9,619	4,984	809,696
Nebraska	11	9	11	6,598	4,841	2,624	264,806
New Hampshire	47	42	23	2,346	1,743	1,239	156,206
New York	26	36	26	44,367	32,020	20,262	2,653,839
North Carolina	33	43	45	19,959	11,963	5,020	1,099,278
Oregon	7	7	19	12,479	8,539	4,011	442,303
Pennsylvania	34	31	36	30,366	22,761	11,429	1,695,741
Rhode Island	2	18	38	6,428	2,270	894	136,336

Source: Alaska Department of Education

APPENDIX II

PRIMARY GOALS

STATE OR PROVINCE	Problem Solving	Top-Down Design	Programming Languages	Team Programming	Structured Programming	Programming Stressed	Major Project
Arkansas	X		B,P		X		
Connecticut	X		B,P,L,Pi				
Florida	X	X	C,PL,F,P,B	X	X	X	
Iowa	X	X	B,P		X	X	X
Kentucky	X		B		X	X	
Louisiana	X	X	B,P,C,F		X	X	X
Mississippi	X	X	B,P,C,F		X	X	
North Carolina	X		NA			X	
Tennessee	X		B,P	X	X	X	X
Texas	X	X	B,P,C,F		X		
Alberta	X	X	B				X
British Columbia	X	X	P		X	X	X
Manitoba	X		B,P,C,F	X	X	X	X
Newfoundland	X		B				
NOVA SCOTIA	X	X	B	X	X		
Ontario	X		B,P				
Prince Edward I.	X		B				
Quebec	X	X	NA		X	X	X
Saskatchewan	X		P		X	X	X

PROGRAMMING LANGUAGES KEY: B=BASIC, P=PASCAL, C=COBOL, F=FORTRAN, Pi=PILOT, PL=PL/I, L=LOGO, NA=NOT AVAILABLE

APPENDIX III

SECONDARY GOALS

STATE OR PROVINCE	Applications	Graphics	Ethical and Social Issues	Computer Fundamentals	File Handling	Computer Careers	Program Debugging	Arrays
Arkansas	X	X	X	X		X		X
Connecticut	X	X			X	X		X
Florida	X		X		X			X
Iowa					X	X	X	X
Kentucky	X		X	X	X	X	X	X
Louisiana	X	X		X	X		X	X
Mississippi	X		X		X		X	X
North Carolina	X	X			X			X
Tennessee	X	X			X	X	X	X
Texas	X		X				X	X
Alberta	X	X	X	X	X	X	X	X
British Columbia		X			X		X	X
Manitoba					X		X	X
Newfoundland	X	X	X	X		X	X	X
NOVA SCOTIA	X		X	X	X	X	X	X
Ontario	X	X	X		X	X	X	X
Prince Edward I.	X		X	X		X	X	X
Quebec	X		X	X	X	X	X	X
Saskatchewan					X		X	X

APPENDIX III CONTINUED

SECONDARY GOALS

STATE OR PROVINCE	Pseudocode	Flowcharts	History of Computers	I/O	Control Structures	Subprograms	Data Communication	Sort and Search
Arkansas		X	X	X	X		X	
Connecticut		X		X	X		X	
Florida			X	X	X	X	X	
Iowa			X	X	X	X		
Kentucky				X	X	X		
Louisiana	X	X	X	X	X			
Mississippi			X	X	X	X		X
North Carolina					X	X		
Tennessee			X	X	X	X		X
Texas				X	X			X
Alberta				X	X	X	X	
British Columbia	X			X	X	X		X
Manitoba	X	X		X	X	X		X
Newfoundland			X	X	X		X	
NOVA SCOTIA				X	X	X	X	X
Ontario				X	X	X		X
Prince Edward I.		X	X					
Quebec	X		X	X	X	X		
Saskatchewan				X	X	X		X

APPENDIX III CONTINUED

SECONDARY GOALS

STATE OR PROVINCE	Interpret vs. Compile	Modularity	Second Language	Low-Level Language	Systems Analysis	Operating System	Computer Architecture	Program Documentation
Arkansas			X					X
Connecticut			X	X		X		
Florida		X	X		X			
Iowa	X			X	X			
Kentucky								
Louisiana				X			X	X
Mississippi		X				X		
North Carolina								
Tennessee						X		
Texas	X	X	X					X
Alberta		X	X	X	X	X	X	X
British Columbia	X	X		X	X	X		X
Manitoba	X	X	X	X		X		
Newfoundland								X
NOVA SCOTIA								X
Ontario	X			X			X	
Prince Edward I.								
Quebec		X		X	X			X
Saskatchewan	X			X		X	X	

APPENDIX IV

COMMONLY USED COMPUTER TERMS

ALGOL- A language designed by the International Federation for Information Processing to provide a general-purpose language suitable for communicating and executing algorithms, and teaching computer science.

Algorithm- Precise definition of a method of solving a problem.

AMPL- A dialect of APL which avoids the APL character set.

APL- A Programming Language. A high-level language developed by Kenneth Iverson which uses many special symbols for functional operations.

Arrays- A structured data type composed of a fixed number of data components of the same type.

ASCII- A seven bit code used as a standard for data transmission.

Assembly Language- A language in which all operators and virtually all operands are represented by mnemonic names.

BASIC- Beginner's All-Purpose Symbolic Instructional Code. An interactive programming language developed in the 1960's at Dartmouth College by Kemeny and Kurtz which was intended to be simple to learn.

Binary Search- A quick method for searching an ordered, dense list by successively looking at that portion of the data where the desired record is known to be.

Boolean Operators- A symbol used in a Boolean expression

whose operation results in a Boolean value of true or false.

Bubble Sort- An algorithm which operates by scanning a list of records and, whenever adjacent records are found to be out of order, the records are swapped.

Bugs- Errors in either the syntax or logic of a computer program.

C- A general purpose high-level programming language developed by Dennis Ritchie in 1972 which is beginning to replace assembly language in many programming environments.

CAI- Computer Assisted Instruction. The use of computers to present drill and practice exercises and tutorials to students.

COBOL- Common Business-Oriented Language. A high-level language which is English-like in style and is extensively used for business applications.

Coding- The actual writing of a computer program.

Compiler- Translates user-written programs (source code) into a language the computer can execute (object code).

Courseware- Computer software used in education.

Database- A file of data stored and organized for easy access.

Debugging- The process of finding and correcting errors in computer programs.

DOS- Disk Operating System. Software that initiates the interaction of computer components which allows other programs to execute.

Error Trapping- The creation of programs which detect and correct for bad input.

Files- A collection of data with some common aspect(s) organized for some specific purpose.

Flowchart- Graphic means of describing a sequence of computer operations, using standardized symbols for various computer operations.

FORTH- A programming language created by Charles H. More which greatly reduces the work of writing subroutines by defining them as words.

FORTRAN- FORmula TRANslation. A numerical, scientific, high-level language developed in 1956 and still in wide use among the scientific community today.

High-Level Language- Programming languages which are close to English and which must be interpreted or compiled before execution.

Interpreter- Transforms a computer program directly into a sequence of machine actions.

Iteration- The repeated execution of lines of code until some condition is satisfied.

LISP- LISt Processing. A sophisticated, high-level, list-processing language used in artificial intelligence research.

Logo- Programming language invented by Seymour Papert used in computer-related learning activities.

Low-Level Language- Programming languages which are close to the native (binary) language of the computer and which can be executed without intermediate steps.

Machine Language- A programming language which is immediately executable by the computer and whose typical statement consists of a single operator-operand pair.

Module- A logically self-contained and discrete part of a larger program

Pascal- An Algol-related procedure-oriented language developed by Wirth in 1971 which was designed to be easy to learn and debug.

Peripherals- Devices which input data to the computer or receive its output.

Pilot- a simple computer-assisted instruction authoring language.

PL/I- Programming Language One. A procedure-oriented language developed in the 1960's which combines the desirable features of FORTRAN, COBOL and ALGOL. PL/I is equally suited for scientific or data processing applications.

Procedure- A portion of a high-level program which performs a subtask.

Pseudocode- A mixture of English and a programming language used in program design.

Random Access Files- Files stored so that the time required to access a record is independent of the order of the records.

Recursion- A mathematical concept in which one or more functions of a variable are defined by giving initial values and by giving the value for larger integers in terms of smaller ones.

Registers- A specialized storage element of the Central Processing Unit (CPU) which stores a string of bits representing information.

Sequential Files- Files in which data is stored sequentially on a key, but which can also be accessed by means of an index.

Shell Sort- A method of sorting records which is also called the diminishing-increment sort.

Spreadsheet- An "electronic ledger", used for the manipulation of groups of numbers.

Structured Programming- A methodological style of writing programs by combining carefully written subunits and by following certain accepted programming procedures.

Subroutine- A portion of a program which is a logically separate part of the program and which performs a specific task necessary for the program's execution.

Subscripted Variables- Variables used in to store arrays.

Systems Analysis- The analysis, design, installation and evaluation of an information system.

Top-Down Design- Design which begins by setting goal systems, and decomposes that system into subsystems.

Variable- A location in memory referenced by an identifier (variable name) where a data value can be stored.

REFERENCES

Adams, J. C. (1983, January). How one computer science program grew. Classroom Computer News, pp. 74-75.

Alberta Education (1985). Computer processing 10-20-30 curriculum guide. Edmonton: Alberta Education.

Anchorage Borough School District (1985). Computer education: Scope and sequence (Grades K-12). Anchorage: Anchorage Borough School District (ERIC Document ED 264829).

Andersen, B. S. (1982). Pascal vs. BASIC. In Barrette, P. (Ed.) First annual conference proceedings: Microcomputers in K-12 education (pp. 51-60). Rockville, Maryland: Computer Science Press.

Arkansas Department of Education (1986). Computer science. Arkansas Department of Education: Little Rock.

Baird, W. E. (1983). The high school computer science project: An introduction to programming in Pascal with color graphics. Journal of Computers in Mathematics and Science Technology 3 (2), 22-26.

Berry, J. & Kramer, P. (1984). Input: planning and staff development. Output: a successful K-12 computer curriculum. In Computers in the new curriculum: Proceedings of the fourth annual conference of the Texas computer education association (pp. 34-35). Austin, Texas: Texas Computer Education Association (ERIC Document 243455).

Bitter, G. (1983, February) The road to computer literacy part v: Objectives and activities for grades 10-12. Electronic Learning, pp. 54-60.

Bork, A. (1983) Computers and the future: education. In Bork, A. (Ed.) Computers and learning: A compendium of papers (pp. 131-142). Irvine, California: Educational Technology Center (ERIC Document ED 239591).

Bork, A. (1984, November) Computer futures for education. Creative Computing, pp. 178-180.

Bork, A. (1981) Learning with computers. Bedford, Massachusetts: Digital Press.

Borg, K. (1983) Computer teaching in Sweden. Technical Horizons in Education 8 (1), 41.

Bosler, U. (1985) Teaching computer science at secondary school: A survey of research and evaluation in the Federal Republic of Germany. Technical Horizons in Education 12 (5), 109-111.

Brady, H. (1985, November) Hang on to the power to imagine: An interview with Joseph Weizenbaum. Classroom Computer Learning, pp. 24-27.

Brady, H. & Levine, M. (1985, February) Is computer education off track? An interview with Judah Schwartz. Classroom Computer Learning, pp. 20-24.

Bramble, W. & Mason, E. (1985) Computers in schools. Toronto: McGraw-Hill.

Braswell, J. S. (1984) Advanced placement computer science. Mathematics Teacher 77 (5), 372-379.

Bristol Board of Education (1983) Computer education plan K-12." Bristol, Connecticut: Bristol Public Schools (ERIC Document ED 237084).

British Columbia Ministry of Education (1985). Computer science 12 curriculum guide. Victoria: Province of British Columbia.

Brown, D. L. (1984) Teaching pascal: Who, what, when, where, how...and why? In Computers in the new curriculum: Proceedings of the fourth annual conference of the Texas computer education association (pp. 39-43). Austin, Texas: Texas Computer Education Association (ERIC Document ED 243455).

California State Department of Education (1985) Computers in education: Goals and content. Sacramento: California State Department of Education (ERIC Document 255192).

Chaney, D. H. (1977) Teaching computer programming in an environment where collaboration is required. Journal of the Association of Educational Data Systems 11, 1-5.

Coburn, P. et. al. (1985) Practical guide to computers in education. Reading, Massachusetts: Addison-Wesley.

College Entrance Examination Board (1986) Advanced placement course description: Computer science. Princeton, New Jersey: College Entrance Examination Board.

Computer Advisory Committee (1985) Report of computer advisory committee. St. John's, Newfoundland: Government of Newfoundland and Labrador.

Connecticut Board of Education (1985) A guide to computers in education: Instruction. Hartford: Connecticut Board of Education.

DeVault, M. V. & Harvey, J. G. (1985) Teacher education and curriculum development in computer education. Technical Horizons in Education 12 (7), 83-86.

Education Turnkey Systems (1985) Uses of computers in education. Falls Church, Virginia: Education Turnkey Systems.

Er, M. C. (1984) On teaching computer programming. Journal of Science and Mathematics Teaching in Southeast Asia 7 (2), 33-35.

Florida Department of Education (1983) Computer education curriculum frameworks. Tallahassee: Florida Department of Education.

Florida Department of Education (1983) Minimum performance standards for Florida schools. Tallahassee: Florida Department

Georgia Department of Education (1986) Essential skills for computer technology. Atlanta: Georgia Department of Education.

Gold Beach Union High School (1986) Computer course outlines. Gold Beach, Oregon: Gold Beach Union High School.

Guse, G. M. (1986) Establishing a districtwide computer curriculum. NASSP Bulletin 7 (489), 6-9.

Hansen, M. (1986, 16 December). State of New Hampshire Department of Education, Personal Communication.

Indiana Consortium for Computer and High Technology Education (1985) Plan for the second biennium July 1, 1985-June 30, 1987. Indianapolis: Indiana Department of Education.

Iowa Department of Public Instruction (1986) High school computer science: A minimum course. Des Moines: Iowa Department of Public Instruction.

Jensen, T. (1982) New information technologies and education in Denmark. European Journal of Education 17 (4), 383-394.

Keil, K. A. (1982) The general introduction of computers into Bavarian schools. Technical Horizons in Education 9 (1), 61-64.

Kentucky Department of Education (1986) Computer course outlines. Frankfort: Kentucky Department of Education.

Kentucky Department of Education (1984) Program of studies for Kentucky schools grades K-12. Frankfort: Kentucky Department of Education.

Knezek, D. (1986, December 10). Texas Education Agency, Personal Communication.

Kollerbaur, A. (1983) Computers in Swedish schools: Experience, research and problems. Technical Horizons in Education 10 (3), 79-86.

Kunces, D. S. (1986) Planning guide for high school diploma computer proficiency requirement. Augusta: Maine Department of Educational and Cultural Services.

Kunces, D. S. (1986, December 11). State of Maine Department of Educational and Cultural Services, Personal Communication.

Lengel, J. P. (1983) Computer considerations for Vermont schools. Montpelier: Vermont State Department of Education (ERIC Document ED 239593).

Lemos, R. S. (1975) FORTRAN programming: An analysis of pedagogical alternatives. Journal of Educational Data Processing 12, 21-29.

Lind, M. L. (1986, December 19). State of Alaska Department of Education, Personal Communication.

Louisiana Department of Education (1983) Computer science curriculum guide. Baton Rouge: Louisiana Department of Education.

Manitoba Department of Education (1983) Computer science 305 pilot guide 1983. Winnipeg: Province of Manitoba Department of Education.

Masterson, F. A. (1984) Languages for students. Byte 9 (6), 233-238.

Ministiere de l' Education, Province du Quebec (1984) Secondary school curriculum: Introduction to computer science. Quebec City: Province du Quebec.

Mississippi Department of Education (1986) State of Mississippi curriculum structure philosophy, goals,

skills, and concepts for curriculum development and instructional planning in Mississippi. Jackson: Mississippi Department of Education.

Naimi, L. (1987, January 12). State of Connecticut Department of Education, Personal Communication.

New Hampshire State Department of Education (1986) Student competencies for computer education. Concord: New Hampshire State Department of Education.

New Hampshire State Department of Education (1985) Computer survey: Secondary 1984-1985. Concord: New Hampshire State Department of Education.

Newfoundland and Labrador Department of Education (1982) Computer studies 2206 course description. St. John's, Newfoundland: Government of Newfoundland and Labrador.

Newfoundland and Labrador Department of Education (1982) Computer studies 2206 a guide for implementation. St. John's, Newfoundland: Government of Newfoundland and Labrador.

North Carolina Department of Public Instruction (1986) Mathematics course outlines. Raleigh: North Carolina Department of Public Instruction.

Nova Scotia Department of Education (1986) Draft curriculum guide: Computer related studies CRS 331 and CRS 441. Halifax: Nova Scotia Department of Education.

Nova Scotia House of Assembly (1987) Speech from the throne. Nova Scotia House of Assembly Debates and Proceedings, February 26, 1987, 5.

Ogle, T. (1986, December 29). State of Missouri Department of Elementary and Secondary Education, Personal Communication.

Ollivier, M. (1962) British North America acts and selected statutes, 1867-1962. Ottawa: Queen's Printer.

Ontario Ministry of Education (1983) Curriculum guideline for the intermediate and senior divisions 1983 computer studies. Toronto: Ontario Ministry of Education.

Ontario Ministry of Education (1983) Functional requirements for microcomputers for educational use in Ontario schools- stage I. Toronto: Ontario Ministry of Education.

Peng, T. C. (1982) Should computer studies and computer appreciation be introduced in Malaysian schools. Journal of Science and Mathematics Teaching in Southeast Asia 5 (1), 8-10.

Phipp, C. C. (1984) High school computer science: Assumptions, rationale and course descriptions. In Conference abstracts: AEDS 84. Journal of Computers in Mathematics and Science Teaching 4 (2), 57-59.

Plog, M. (1984) Out of diversity, an Evolving curriculum. Rainbow 3 (10), 52-53.

Prince Edward Island Department of Education (1984) Computer literacy revised program of studies. Charlottetown: Province of Prince Edward Island.

Ragsdale, R. G. (1982) Computers in the schools. A guide for planning. Toronto: O.I.S.E. Press.

Ralston, A. (Ed.) (1983) Encyclopedia of computer science and engineering. New York: Van Nostrand Reinhold.

Renner, P. V. (1987, January 6). Gold Beach Union High School, Personal Communication.

Sadowski, B. R. (1984) Beginning an integrated K-12 computer curriculum. In Computers in the new curriculum: Proceedings of the fourth annual conference of the Texas computer education association (pp. 188-193). Austin: Texas Computer Education Association (ERIC Document ED 243455).

Saskatchewan Department of Education (1984) A curriculum guide for division IV computer application 10, 20 and computer science 10, 20, 30. Regina: Saskatchewan Department of Education.

Scheneberger, D. (1987, January 12). Tillamook High School, Personal Communication.

Schneiderman, B. (1976) Exploratory experiments in programming behavior. International Journal of Computer Information Science 5, 123-143.

Smith, J. R. (1987, January 2). California State Department of Education, Personal Communication.

Swigger, K. M. (1984) Teaching beginning students how to program. In Computers in the new curriculum: Proceedings of the fourth annual conference of the Texas computer education association (pp. 227-232).

Austin: Texas Computer Education Association (ERIC Document ED 243455).

Task Force on School Curriculum, Task Force on Teacher Certification (1985) Curricula recommendations for secondary schools and teacher certification. Baltimore: Association for Computing Machinery.

Tennessee Board of Education (1986) Computer technology curriculum frameworks. Nashville: Tennessee Board of Education.

Texas Education Agency (1986) Curriculum framework. Austin: Texas Education Agency.

Watt, D. (1983) The push for standardization. Popular Computing 2 (11), 65-67.

Usher, P. L. (1986, December 15). Indiana Department of Education, Personal Communication.

Watson, C. D. (1986, December 15). Arkansas Department of Education, Personal Communication.

Wisconsin Department of Public Instruction (1986) Computer science course outlines. Milwaukee: Wisconsin Department of Public Instruction.

Woodhouse, D. (1983) Introductory courses in computing: Aims and languages. Computers and Education 7 (2), 79-89.