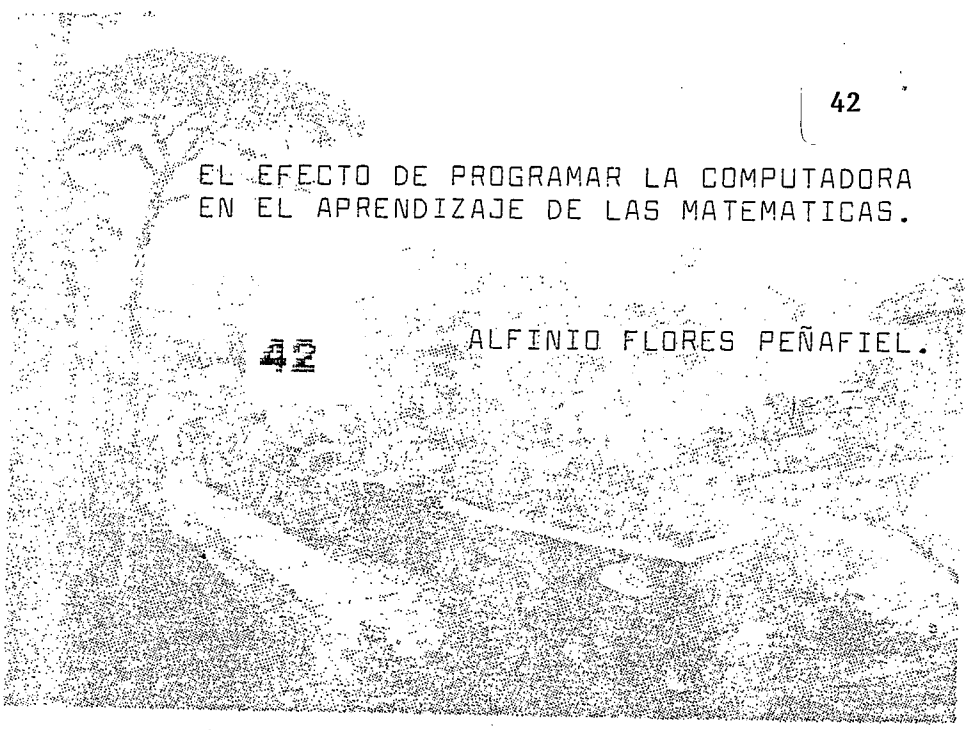
COMUNICACIONES DEL CIMAT



42

EL EFECTO DE PROGRAMAR LA COMPUTADORA EN EL APRENDIZAJE DE LAS MATEMATICAS.

42

ALFINIO FLORES PEÑAFIEL.

PRIMERA REUNION CENTROAMERICANA Y DEL CARIBE SOBRE FORMACION DE PROFESORES E INVESTIGACION EN MATEMATICA EDUCATIVA.

MERIDA, YUCATAN, MEXICO.
DEL 9 al 11 de Abril de 1987.

CENTRO DE INVESTIGACION EN MATEMATICAS

Apartado Postal 402
Guanajuato, Gto.
México
Tels. (473) 2-25-50
2-02-58

EL EFECTO DE PROGRAMAR LA COMPUTADORA EN EL APRENDIZAJE DE LAS MATEMATICAS

Alfinio Flores Peñafiel

Centro de Investigación en Matemáticas, Guanajuato, MEXICO

RESUMEN

Entre las formas de uso de la computadora para el aprendizaje de las matemáticas en las que el alumno tiene un papel activo, la programación es una de la más difundidas. Las razones son el valor intrínseco de programar la computadora más la transferencia esperada de habilidades aprendidas al programar a otros campos del conocimiento. Aunque los resultados obtenidos en esta dirección son promisorios, es también claro que se necesita plantear el problema con más precisión y realizar investigaciones cuidadosas a fin de apoyar las afirmaciones.

CONTENIDO

FORMAS DE USO DE LA COMPUTADORA EN LA ENSEÑANZA DE LAS MATEMATICAS

El acceso amplio a la computadora está modificando lo que percibimos como importante en la enseñanza de las matemáticas. Habilidades a las que se les dedica mucha atención en el presente, serán sustituidas por otras (Conference Board of the Mathematical Sciences, 1982; 1984; Flores Peñafiel, 1986). Nuevos conceptos serán importantes en la preparación de profesores de

matemáticas (CUPM Panel on Teacher Training, 1983; Commission on the Education of Teachers of Mathematics of the NCTM, 1981; Ralston, 1984). El efecto que pueden tener las computadoras en la enseñanza de las matemáticas ha sido discutido por varios autores (ver por ejemplo Corbitt, 1985; Fey, 1984). Aunque se reconoce entre los matemáticos y los maestros que la computadora tiene un gran potencial en la enseñanza de las matemáticas, hay menos acuerdo acerca de cómo usarla en el salón de clases para mejorar realmente el aprendizaje de las matemáticas. La computadora puede fungir como instructor, servir como herramienta o ser instruida (Taylor, 1980). Entre las formas de uso están las que el alumno tiene un papel activo, donde le dice a la máquina qué debe hacer, la utiliza como herramienta o le enseña a hacer algo (Flores Peñafiel, 1985b). El desarrollo de programas y paquetes de computación sofisticados tales como las hojas de cálculo, paquetes "inteligentes" como muMath y TK!Solver, dan a esta forma de uso un gran potencial de creatividad (Arganbright, 1985; Fey, 1984; Wilf, 1982; Williams, 1982). Programar la computadora es la más difundida de esta forma activa de uso. Esto se debe en parte a razones prácticas: no se necesita equipo muy sofisticado o costoso, se puede llevar a cabo en condiciones reales. Sin embargo también hay razones teóricas.

RELACION ENTRE PROGRAMAR Y DESEMPEÑO MATEMATICO

Varios matemáticos y científicos de la computación afirman que programar puede tiene un efecto benéfico en varios aspectos del aprendizaje de las matemáticas. Kemeny (1966) menciona que al hacer que el alumno escriba un programa para hacer un algoritmo y tener que explicar el algoritmo a la computadora, los alumnos llegan a un nivel de comprensión del problema mucho mayor de la que hayan tenido antes. Knuth (1974) afirma que una persona no entiende realmente algo hasta que puede enseñarselo a una computadora.

Forsythe (1968) afirma que aprender junto matemáticas y computación tiene ventajas pedagógicas ya que los conceptos básicos de una refuerzan el aprendizaje de la otra, por ejemplo función en matemáticas y procedimiento en un lenguaje de programción. McKenna (1972) sugiere usar la programación matemática como un medio de estimular la experimentación de los alumnos en matemáticas y que los alumnos pueden aprender de los errores. Friske (1985) señala las analogías entre los procesos de pensamiento usados al programar y los requeridos para escribir demostraciones. Sugiere que la metodología y principios que han sido tan efectivos para enseñar programación se aplique a la enseñaza de habilidades de demostración. La lista de matemáticos que afirman que programar la computadora puede tener un efecto benéfico puede extenderse (Birkhoff, 1972; Dijkstra, 1974; Smith, 1984; Stroyan, 1978). Ross y Howe (1981) resumen de esta manera las afirmaciones:

- la programación proporciona una justificación del rigor formal de las matemáticas y lo ilustra;
- la programación estimula a los alumnos a estudiar las matemáticas a través de actividades exploratorias;
- la programación permite una comprensión clave de ciertos conceptos matemáticos;
- la programación proporciona un contexto para resolver problemas y un lenguaje con el cual los estudiantes pueden describir sus propias soluciones a problemas.

Varias personas y organizaciones dedicadas a la educación matemática también han expresado que los alumnos pueden mejorar su comprensión de conceptos matemáticos al programar una computadora (NCTM Commitee on Computer Oriented Mathematics, 1965). Desde entonces, otros autores han apoyado que los alumnos de todas las edades y habilidades usen activamente la computadora en el aprendizaje de las matemáticas (Bell, 1978; LaFrenz y Kieren, 1969; Hatfield, 1982; Luehrman, 1980; Shumway, 1983). Algunos

autores han dado ejemplos del tipo de ideas matemáticas que pueden ser exploradas con programas relativamente sencillos (Flores Peñafiel, 1984a; 1984b; Shumway, 1987; Smith, 1984). Se ha detectado el paralelismo entre el pensamiento necesario para formar, probar, corregir y refinar un programa de computación, y varios aspectos del pensamiento matemático. De acuerdo con Hatfield (1982), algunos de los aspectos que pueden ser mejorados por la programación son los procesos de analizar, simplificar, particularizar, generalizar, justificar, conjeturar y estructurar. Wells (1981) encontró los siguientres procesos comunes en la resolución de problemas y en la escritura de programas de computación: uso de heurística, submetas, ensayo y error, y patrones regulares de análisis y síntesis. Clement, Lochhead y Soloway sugieren que corregir programas puede contribuir al uso de la heurística de mirar hacia atrás, y que la descomposición de los problemas de pogramación en subproblemas pueden estimular el uso de la heurística de planeación. Ellos sugieren que las actividades de programación pueden mejorar la persistencia y la habilidad de formular problemas.

Papert (1980) después de ver a niños usar LOGO afirmó que los niños que están inmersos en un ambiente tan rico como ése, pueden aprender ideas matemáticas importantes, incluyendo recursión y geometría casi por sí mismos.

Sin embargo, de acuerdo con varios autores (Pea, 1983; De Corte, 1984) se necesita hacer más investigación para sustentar estas afirmaciones. Es necesario establecer si hay una correlación positiva entre programar y aprovechamiento matemático, y si la hay, determinar si hay una relación de causa y efecto.

INVESTIGACIONES REALIZADAS

Algunos estudios establecen una relación entre escribir programas y un

mejor desempeño en matemáticas. Johnson y Harding (1979) en la universidad de Cambridge, encontraron que los alumnos que toman cursos de programación en computación, obtienen mejores resultados en una prueba que requiere habilidad matemática de mayor nivel. Otros estudios encontraron correlación entre programar y diversos aspectos de las matemáticas. Duggar (1983) encontró que la habilidad algebraica estaba positivamente correlacionada con la habilidad de programar. Zbyszynski (1983) encontró correlación entre heurística y programación.

También se han hecho investigaciones tendientes a determinar si existe una relación de causa efecto. Clement, Lochhead y Soloway (1980) encontraron que un número significativamente mayor de alumnos universitarios fueron capaces de resolver problemas algebraicos al escribir programas de computación que al escribir simplemente una ecuación algebraica. Hatfield y Kieren (1972) encontraron que alumnos de secundaria tenían una mayor habilidad para resolver problemas desconocidos después de construir, corregir y refinar programas de computación. Blume y Schoen (1985) encontraron que los alumnos que programaban usaban las estrategias de ensayo sistemático, ecuaciones y visión retrospectiva con más frecuencia que los alumnos que no programaban, aunque no hubo diferencias significativas en cuanto a desempeño. Knerr (1982) encontró que el enfoque apoyado por la computadora fue más efectivo que el enfoque tradicional, especialmente para conceptos diversos y complejos.

Flores Peñafiel (1984c,1985b,1987) realizó una serie de estudios con el propósito de determinar si programar la computadora mejora la comprensión de los alumnos de conceptos de cálculo. Antes del experimento formal se condujeron dos estudios piloto. Un experimento informal de enseñanza con alumnos de licenciatura de matemáticas sirvió para seleccionar los programas para el estudio. Otra experiencia con alumnos de cálculo de un bachillerato tecnológico sirvió para desarrollar un conjunto de materiales de referencia y de actividades para familiarizar a los alumnos con el lenguaje BASIC. En el

experimento conducido con dos grupos de cálculo de una Facultad de Química de una universidad del país, para determinar la contribución específica que la escritura de un programas tiene en el proceso de comprensión, se compararon dos formas de uso de la computadora en el estudio del concepto matemático de derivada. En una de las formas los estudiantes escribieron, corrieron y modificaron sus propios programas cortos para estudiar los conceptos de velocidad, pendientes de secantes y tangentes, y derivadas como un límite. En la otra forma, los alumnos recibieron programas ya hechos que ilustraban los mismos conceptos. Los alumnos de los dos tratamientos teclearon los programas, analizaron los resultados y modificaron parámetros y datos. Los alumnos no tenían experiencia anterior con computadoras. Se dio a los alumnos un tratamiento previo de cuatro sesiones de dos horas a la semana con la computadora, para habilitarlos a escribir programas cortos. Presentaron un examen para determinar sus habilidades para programar. Los alumnos fueron asignados aleatoriamente a uno de los tratamientos. Los tratamientos duraron tres sesiones con dos horas a la semana. Una prueba de cálculo fue el examen subsecuente. Un análisis de varianza de 2×2 (tratamiento por grupo) no mostró diferencias estadísticamente significativas por la interacción o el tratamiento. Otro análisis de varianza (tratamiento por habilidad para programar) tampoco mostró diferencias estadísticamente significativas. Además de las mediciones escritas, se realizaron observaciones durante el laboratorio y se entrevistaron alguno de los alumnos.Las actividades del tratamiento previo fueron suficientes para desarrollar familiaridad y confianza con el equipo y con los elementos de BASIC. Los alumnos pensaban que sabían lo suficiente para escribir programas cortos. Sin embargo, con programas dados, no siempre les era claro el propósito de cada parte del programa.

Habilidad para escribir programas. Aunque la mayor parte de los alumnos hacían el esfuerzo por entender el problema antes de escribir cualquier cosa, había poca planeación de su parte. Los programas eran bastante cortos así que

no era de primordial importancia el cómo estructurarlos.

Errores. Pocos alumnos se daban cuenta, al ver el resultado en la pantalla, que el programa tenía un error. Cuando se les indicaba que había un error, pocos alumnos mostraban un método bien definido o una estrategia para localizar el error en el programa.

Generalizar. Pocos alumnos escribían de manera espontánea programas más generales o poderosos que los que se les pedían.

Correr el programa. En general los alumnos no corrían el programa con más ejemplos que los que se les solicitaba en la actividad. No buscaban ejemplos particularmente interesantes o iluminadores.

Búsqueda de patrones. Los alumnos trataban de encontrar un patrón en los resultados pero sólo después de que en varias actividades se les había pedido explícitamente que lo hicieran.

Se puede concluir que la mayor parte de los alumnos sólo habían alcanzado el nivel dos en habilidad de programar (Pea y Kurland, 1983). Los cuatro niveles descritos por Pea y Kurland son:

1. Usar programa
2. Generar código
3. Generar programa
4. Desarrollar software

De acuerdo con esta clasificación, un estudiante que ha alcanzado el nivel dos conoce la sintaxis y la semántica de los comandos más importantes de un lenguaje. El estudiante puede leer un programa escrito por otra persona y explicar lo que realiza cada línea. El estudiante puede encontrar errores de sintaxis, y puede escribir programas sencillos del tipo que ha visto previamente. Al programar se planea poco, no se documentan los programas. El estudiante no usa subrutinas o procedimientos creados como partes de otros programas. Este nivel es suficiente para escribir programas cortos.

PROGRAMADORES EXPERTOS

Parte de las dificultades para encontrar una relación más estrecha entre programar la computadora y el desempeño matemático se debe a que no es claro lo que significa programar en cada caso. Lo que los alumnos hacen en las primeras etapas de programación, sugiere que en la práctica muchos no utilizan el pensamiento sistemático, analítico y reflexivo que es característico de los programadores expertos. Para que la programación promueva el desarrollo de ciertas habilidades de pensamiento, los alumnos deben alcanzar una comprensión relativamente sofisticada de la programación. Hay que distinguir entre programación por líneas y la programación modular. El estilo de construir programas de los principiantes, ya sea en LOGO o en BASIC, contrasta con la programación modular, un proceso planeado de solución estructurada de problemas. En la programación modular se aíslan los elementos componentes de una tarea, se desarrollan procedimientos para su ejecución, se juntan las partes en un programa y se corrigen.

Para usar un lenguaje como LOGO para desarrollar una comprensión de conceptos matemáticos como variables y funciones, y para aprender técnicas de solución de problemas tales como planear y solución modular de problemas, se deben usar los aspectos de LOGO que involucren estos conceptos. El enfoque no guiado, de exploración libre no conduce a los alumnos a una más profunda comprensión de la estructura y operación de un lenguaje de programación y por tanto no los conduce a usar o desarrollar habilidades de pensamiento tales como descomposición de problemas, planeación o eliminación sistemática de errores.

Kurland, Mawby y Cahir (1985) al entrevistar a programadores expertos, encontraron que las habilidades de pensamiento relacionadas con la habilidad de programar tales como ser lógicos y disciplinados se encontraban en los sujetos

desde antes de que estos aprendieran a programar.

Aprender a programar bien, como volverse experto en cualquier otro campo, lleva una considerable cantidad de tiempo y dedicación. Parece claro que a menos que se descubran métodos radicalmente mejores de enseñar programación, no todos los alumnos pueden ni deben convertirse en expertos en los lenguajes de programación.

Sin embargo, esto no significa que no se deban enseñar las ideas fundamentales de la programación a todos los alumnos. Se pueden enseñar los conceptos fundamentales de programación y de computación en el salón de clases de forma que pueda servir como fundamento para la interacción futura de los alumnos con la computadora. Hay que encontrar un papel para la programación de modo que el alumno sea expuesto a los conceptos computacionales incorporados en los lenguajes de programación y que son utilizados por la nueva generación de programas de aplicación.

Para que los alumnos entiendan más acerca de la programación se necesitan más materiales de apoyo que ayuden a los alumnos a obtener una idea más amplia de lo que es la porgramación, cómo funcionan los lenguajes y cómo los problemas se pueden resolver de manera más elegante (Pea, 1984).

PENSAMIENTO ALGORITMICO

Además de una mayor claridad en cuanto al nivel de programación necesaria para esperar transferencia de habilidades desarrolladas con la programación a otras áreas del pensamiento, es necesario identificar habilidades para las cuales esta transferencia se pueda dar de manera natural. Como se ha visto es poco probable que la programación propicie la transferencia de habilidades muy generales de pensamiento.

De acuerdo con Hatfield (1985), el pensamiento algorítmico proporciona un

marco clave para pensar acerca de las razones para integrar la programación en las matemáticas escolares. La matemática algorítmica en el sentido contemporáneo es lo que se refiere a crear los algoritmos, y a la disposición mental de pensar en términos de algoritmos par resolver problemas y desarrollar teoría (Maurer, 1984). Un algoritmo se define como un plan completo e inambiguo para la solución de un problema matemático.

Un algoritmo tiene cinco características (Knuth, 1973):

1) el plan de solución debe terminar después de un número finito de pasos

2) cada paso de la solución debe estar definido de manera precisa

3) el plan de solución debe considerar la posibilidad de datos iniciales

4) el plan de solución debe dar al menos un resultado

5) la implementación de la solución debe de ser factible dadas las restricciones de tiempo, dinero y capacidad de la tecnología de computación disponible.

Desde un punto de vista educativo los algoritmos se proponen como herramientas excelentes que pueden mejorar la enseñanza, el aprendizaje, la comprensión y el hacer de las matemáticas (Johnson, 1985). Un buen algoritmo no sólo obtiene un resultado, sino que mejora la comprensión de un concepto. Desarrollar un algoritmo requiere una fuerte comprensión del proceso. Para diseñar o escoger los algoritmos para ser usados con los alumnos se requiere decidir si el algoritmo se va a utilizar par ilustrar un proceso valioso o si es un algoritmo elegante diseñado unicamente para obtener la solución (Fraser, 1985).

La calidad del conocimiento matemático del alumno es la cuestión fundamental. Un factor crucial para determinar la calidad del conocimiento es la calidad de las experiencias del alumno al construir su conocimiento. El aprendizaje de un algoritmo debe ser una actividad de resolver un problema. Lo que se busca con los algorítmicos es la fase crucial de crear su propio

procedimiento. El contexto de formular, probar, refinar y extender programas de computación en respuesta a tareas algorítmicas viables puede ser un enfoque poderoso para promover los mejores aspectos de un tema algorítmico.

La enseñanza de las matemáticas, en la cual estén integradas actividades de programación de los alumnos que sirvan para construir conocimiento de algoritmos debe tener los siguientes procesos (Hatfield, 1985):

1) identificar una tarea algorítmica apropiada con fuertes conexiones con los aspectos conceptuales de los contenidos matemáticos que se estudian;

2) proponer la tarea algorítmica en forma de problema donde el alumno entienda y acepte el reto como valioso y soluble.

3) ayudar a los alumnos a aceptar sus propias producciones iniciales de un programa de computación como un objeto activo en el cual pueden probar y revisar sus ideas;

4) ayudar a los alumnos a desarrollar estrategias para identificar los errores y limitaciones de sus programas y aprender de ellos;

5) estimular a los alumnos a buscar soluciones más generales de su algoritmo;

6) guiar a los alumnos a hacer abstracciones reflexivas sobre algoritmos relacionados y acerca de los procesos intelectuales usados en resolver problemas algorítmicos.

Entre los principios y paradigmas de las ciencias de la computación que son importantes para futuros maestros de matemáticas están el pensamiento algorítmico. Aspectos tales como algoritmo, matemática constructiva, iteración y recursión tendrán efectos profundos en la enseñanza de las matemáticas (Ralston, 1984). Con respecto a la preparación de maestros de matemáticas que puedan integrar la programación y los algoritmos a su enseñanza, se necesitan considerar los siguientes aspectos (Hatfield, 1985):

1) proporcionar atención explícita al desarrollo de su conocimiento del

pensamiento algorítmico,

2) estudiar en varios cursos de matemática de nivel universitario en los cuales el contenido se trate a través de la programación por los alumnos y un tema de algoritmos esté implícitamente o explícitamente involucrado;

3) analizar las metas y contenidos del las matemáticas escolares con suficiente profundidad para ayudarlos a entender la transformación de su propia formación matemática hacia la integración de la programación por los alumnos y el reconocimiento de un tema algorítmico;

4) practicar la enseñanza de clases asesoradas en las cuales ocurran actividades de programación de los alumnos como experiencia integral al construir ideas matemáticas.

El uso de la programación para el desarrollo del pensamiento algorítmico parece ser un campo de investigación promisorio.

Prichard (1985) realizó un estudio encaminado a investigar el desarrollo del conocimiento de iteración por parte de los alumnos al programar la computadora. Se identificaron tres componentes de la iteración: a) valores iniciales específicos, b) un procedimiento que puede ser repetido, c) una regla apropiada mediante la cual se puede detener el proceso. Para entender una iteración, uno debe entender la necesidad y los papeles de las cantidades iniciales, su reconocimiento y selección así como anticipar las consecuencias potenciales de valores particulares. Conocer la segunda componente involucra una comprensión de que un proceso repetitivo y que en cada aplicación del proceso se utiliza un término previo de la secuencia. Para la tercera componente se debe exprear una condición que cuando no sea satisfecha continúe la ejecución del procedimiento y que cuando sea satisfecha conduzca al reconocimiento de la solución del problema. Esto puede involucrar el reconocimiento de un término deseado, el grado de precisión deseada o el número de repeticiones requeridas.

Programar la computadora proporcionó el contexto dentro del cual los alumnos formaron su conocimiento de la iteración. Los alumnos construyeron, ejecutaron, revisaron y refinaron sus programas. El énfasis fue en la conceptualización de los alumnos y en que desarrollaran sus propios algoritmos e ideas que emplearan itereación. Los algoritmos e ideas que involucraban los tres componentes de la iteración fue hechos operativos y probados en los programas de computación.

CONCLUSIONES

De los estudios y observaciones realizadas, se puede afirmar que los alumnos en general no desarrollan de manera espontánea habilidades de pensamiento que se espera promueva la programación tales como:

- pensamiento riguroso, expresión precisa;
- comprensión de conceptos generales tales como procedimiento formal, variable, función y transformación;
- hacer de la corrección de errores una actividad constructiva y planeada;
- habilidad para resolver problemas útiles en cualquier dominio, tales como planear, encontrar un problema relacionado, resolver el problema descomponiendolo en partes; y
- usar pequeños procedimientos como bloques para resolver problemas mayores.

Es necesario guiar a los alumnos a desarrollar estas habilidades. No basta que el contexto de programación las haga posibles para que surjan en los alumnos.

REFERENCIAS

Arganbright, D. (1985). Mathematical applications of electronic spreadsheets. McGraw Hill, 1985.

Bell, F.H. (1978). Can computers really improve school mathematics? Mathematics Teacher, 71, 428-433.

Birkhoff, G. (1972). The impact of computers on undergraduate mathematical education in 1984. American Mathematical Monthly, 79, 648-657.

Blume, G., y Schoen, H. (1985, abril). The effects of learning computer programming on students' performance in mathematical problem solving. Trabajo presentado en la sesión previa de investigación del congreso anual de National Council of Teachers of Mathematics, San Antonio, Tejas.

Clement, J., Lochhead, J., y Soloway, E. (1980). Positive effects of computer programming on the student's understanding of variables and equations. Manuscrito no publicado, University of Massachussetts, Amherst.

Conference Board of the Mathematical Sciences. (1982). The mathematical sciences curriculum K-12: What is still fundamental and what is not. Washington, D.C.: National Science Foundation. 15 p.

Conference Board of the Mathematical Sciences. (1984). New goals for mathematical sciences education. Washington: Conference Board of the Mathematical Sciences.

Commission on the Education of Teachers of Mathematics of the National Council of Teachers of Mathematics. (1981). Guidelines for the preparation of teachers of mathematics. Reston: National Council of Teachers of Mathematics.

Corbitt, M.K. (1985). The impact or computing technology on school mathematics. Aritmetic Teacher, 32 (8), 14-18, 60.

CUPM Panel on Teacher Training (1983). Recommendations on the mathematical preparation of teachers. Washington: Mathematical Association of America.

Dalbey, J., Tourniaire, F., Linn, M.C. Making programming instruction cognitively demanding: an intervention study. En Kurland, D. Midian (Ed.) Developmental

studies of computer programming skills. Technical Report No. 29 Center for Children and Technology, Bank Street College of Education, Nueva York.

De Corte, Erik. (1984, mayo). Does learning to program improve children's thinking skills? Trabajo presentado en 25th International Congress on Computer-based Education, Columbus, Ohio.

Dijkstra, E. W. (1974). Programming as a discipline of mathematical nature. American Mathematical Monthly, 81, 608-612.

Duggar, Cynthia Miller. (1983). A study of the relationships among computer programming ability, computer programming content, computer programming style, and mathematical achievement in a college level BASIC programming course. Dissertation Abstracts International, 44, 95A.

Ehrlich, K., Abbott, V., Salter, W., Soloway, E. (1984). Issues and problems in studying transfer effects of programming. En Kurland, D. M. (Ed.) Developmental studies of computer programming skills. Technical Report No. 29 Center for Children and Technology, Bank Street College of Education, Nueva York.

Fey, James T. (Ed.). (1984). Computing and mathematics: the impact on secondary school curricula. Reston, VA: National Council of Teachers of Mathematics.

Flores Peñafiel, Alfinio (1984a). A microcomputer and the law of small numbers. Artithmetic Teacher, 31(7), 60-61.

Flores Peñafiel, Alfinio (1984b). Pequeños programas, grandes ideas. En Memoria del Simposio Internacional La Computación y la Educación Infantil, p. 231-233.

Flores Peñafiel, Alfinio (1984c). La microcomputadora en la enseñanza del cálculo. Trabajo presentado en el congreso anual de la Sociedad Matemática Mexicana, Mérida, Yucatán.

Flores Peñafiel, Alfinio (1985a). Uso de la computadora en la enseñanza de las matemáticas. Trabajo presentado en el 4º Coloquio de Matemáticas. Taxco, Gro.

Flores Peñafiel, Alfinio (1985b). Effect of computer programming on the learning of calculus concepts. Tesis doctoral, Ohio State University.

National Council of Teachers of Mathematics Committee on Computer Oriented Mathematics. (1965). Computers for school mathematics. Mathematics Teacher, 58, 393-401.

Noss, Richard (1985). Creating a mathematical environment through LOGO. Trabajo presentado en la sesión previa de investigación del congreso anual de NCTM, San Antonio, Tejas.

Papert, Seymour. (1980). Mindstorms: children, computers and powerful ideas. Nueva York: Basic Books.

Pea, R.D. (1983, abril). LOGO programming and problem solving. Trabajo presentado en el simposio de American Educational Research Association, Montreal.

Pea, R. D. (1984). What will it take to learn thinking skills through programming? En Kurland, D. Midian (1984) Developmental studies of computer programming skills. Technical Report No. 29 Center for Children and Technology, Bank Street College of Education, Nueva York.

Pea, R.D., y Kurland, M. (1983). On the cognitive effects of learning computer programming: a critical look. Manuscrito enviado para su publicación.

Prichard, Mary Kim (1985). Student programming as a context for developing mathematicas concepts. Trabajo presentado en la sesión previa de investigación del congreso anual de NCTM, San Antonio, Tejas.

Prichard, Kim; Hatfield, Larry L. (Eds.) (1985). Student programming and algorithmics in mathematics learning and teaching. Department of Mathematics Education, University of Georgia, Athens, Georgia.

Ralston, Anthony (1984) Computing and computer science in the pre-service education of secondary school teachers of mathematics. En A Collection of Papers on Pre-Service Teacher Education. Action Group 6, ICME 5, Adelaide.

Ross, P. y Howe, J. (1981). Teaching mathematics through programming: ten years on. En R. Lewis y D. Tagg (Eds.), Computers in education. Amsterdam: North Holland.

Shumway, Richard J. (1983). Let kids write programs. Arithmetic Teacher, 30(6), 2, 56.

Flores Peñafiel, Alfinio (1986). Cambios en el currículum de matemáticas en el nivel básico. Revista Informativa del Profesor de Matemáticas, abril 1986, 17-20.

Flores Peñafiel, Alfinio (1987). Efecto de programar la computadora en el aprendizaje de conceptos de cálculo. Cuadernos de Investigación. Programa Nacional de Formación y Actualización de Profesores de Matemáticas (por aparecer).

Forsythe, G. E. (1968). What to do till the computer scientist comes? American Mathematical Monthly, 75, 454-462.

Fraser, Rosemary (1985). Roles of mathematics teachers within a curriculum theme of algorithms. Trabajo presentado en la sesión previa de investigación del congreso anual de NCTM, San Antonio, Tejas.

Friske, M. (1985). Teaching proofs: a lesson from software engineering. American Mathematical Monthly, 92, 142-144.

Hatfield, L. L. (1982). Instructional computing in mathematics teacher education. Journal of Research Development in Education, 15(4), 30-44.

Hatfield, Larry, L. (1985). Theoretical perspectives for algorithmics and student programming in school mathematics. Trabajo presentado en la sesión previa de investigación del congreso anual de NCTM, San Antonio, Tejas.

Hatfield, L. L., y Kieren, T.E. (1972). Computer-assisted problem solving in school mathematics. Journal for Research in Mathematics Education, 3, 99-112.

Johnson, David C., y Harding, Robert D. (1979). University level computing and mathematical problem solving ability. Journal for Research in Mathematics Education, 10, 37-55.

Johnson, Jerry (1985) Algorithmics and the mathematics curriculum Trabajo presentado en la sesión previa de investigación del congreso anual de NCTM, San Antonio, Tejas.

Kemeny, J. (1966). The role of computers and their applications in the teaching of mathematics. En Howard F. Fehr (Ed.), Needed research in mathematics education. Nueva York: Teachers College Press.

Knerr, C. C. (1982). The enhancement of traditional instruction and learning Analytic Geometry via computer support. Dissertations Abstracts International, 42A, 3483.

Knuth, D. E. (1973) The art of computer programming Vol. 1 Fundamental algorithms 2a ed. Reading: Addison Wesley.

Knuth, D. E. (1974). Computer science and its relation to mathematics. American Mathematical Monthly, 81, 323-343.

Kurland, D. Midian (1984) Developmental studies of computer programming skills. Technical Report No. 29 Center for Children and Technology, Bank Street College of Education, Nueva York.

Kurland, D. M., Clement, C., Mawby, R., Pea, R .D. (1984). Mapping the cognitive demands or learning to program. En Kurland, D. M. (Ed) Developmental studies of computer programming skills. Technical Report No. 29 Center for Children and Technology, Bank Street College of Education, Nueva York.

Kurland, D. M., Mawby, R., Cahir, N. (1984). The development of programming expertise in adults and children. En Kurland, D. Midian (Ed.) Developmental studies of computer programming skills. Technical Report No. 29 Center for Children and Technology, Bank Street College of Education, Nueva York.

La Frenz, D. E., Kieren, T. E. (1969). Computers for all students: a new philosophy of computer use. School Science and Mathematics, 69, 39-41.

Luehrman, Arthur. (1980). Pre and post-college computer education. En R. P. Taylor (Ed.). (1980). The computer in the school: tutor, tool, tutee. (p. 141-148). Nueva York: Teachers College Press.

Maurer, Stephen B. (1984) Two meanings of algorithmic mathematics. Mathematics Teacher, 77, 430-435.

McKenna, J. E. (1972). Computers and experimentation in mathematics. American Mathematical Monthly, 79, 294-295.

Mohyla, Jury (Ed.) (1984) The role of technology: theme group 3. South Australian College of Advanced Education, Adelaide.

National Council of Teachers of Mathematics. (1980). An agenda for action: recommendations for school mathematics of the 1980s. Reston, VA: National Council of Teachers of Mathematics.

Shumway, Richard J. (1987). 101 ways to learn mathematics using BASIC (K-8). Englewood Cliffs: Prentice Hall.

Smith, D. A. (1984). Interface: calculus and the computer. Filadelfia: Saunders College Publishing.

Stroyan, K. D. (1978). Computer laboratory manual for calculus and linear algebra. Iowa City: Conduit.

Suydam, M. N. (1984). Microcomputers in mathematics instruction. Arithmetic Teacher, 32(2), 35.

Taylor, Robert P. (Ed.). (1980). The computer in the school: tutor, tool, tutee. Nueva York: Teachers College Press.

Wells, G. W. (1981) The relationship between the processes involved in problem solving and the processes involved in computer programming. Dissertationa Abstracts International, 42, 2009A-2010A.

Wilf, H.S. (1982). The disk with the college education. American Mathematical Monthly, 89, 4-8.

Williams, G. (1982, Octubre). Software Arts' TK! Solver. Byte, 360-376.

Zbyszynski, Henry Michael. (1983). Abstract thought as a component of computer programming. Dissertations Abstracts International, 44, 1367A.