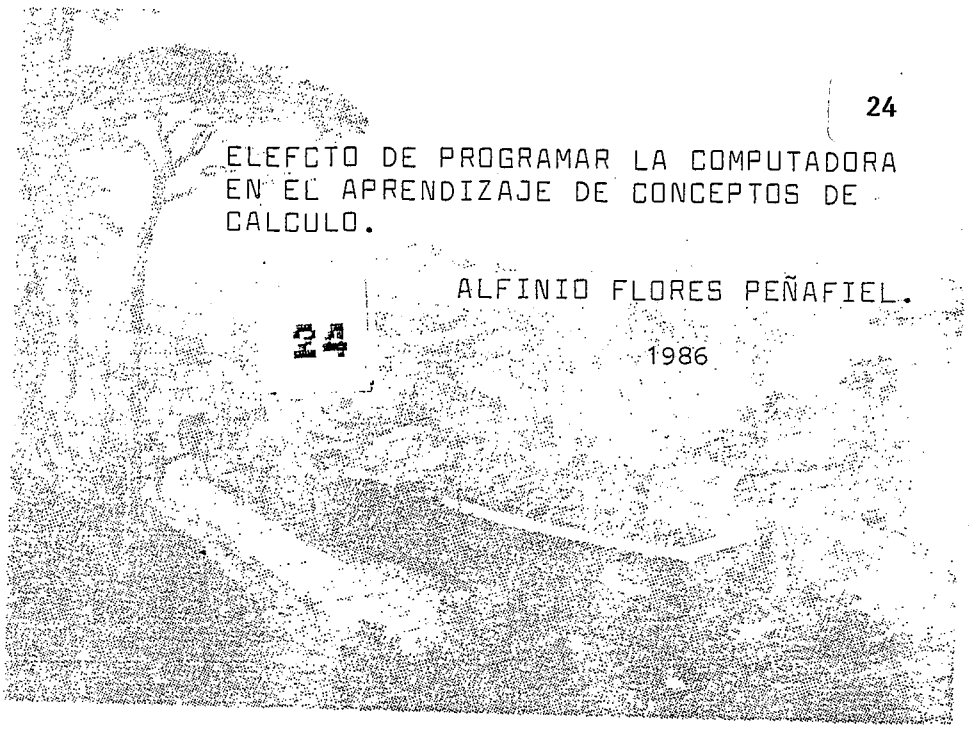
COMUNICACIONES DEL CIMAT



Aceptado para su publicación en Cuadernos de Investigación Programa Nacional de Formación y Actualización de Profesores de Matemáticas, enero 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 CONCEPTOS DE CALCULO

Alfinio Flores Peñafiel

Centro de Investigación en Matemáticas

Guanajuato, Gto. 1986

Reconocimiento: el presente trabajo es una versión abreviada de mi tesis doctoral, la cual fue dirigida por Richard J. Shumway, quien me brindó su apoyo total para que pudiera realizar la investigación en México.

INDICE

RESUMEN 1

REVISION BIBLIOGRAFICA 1

METODOS Y PROCEDIMIENTOS 6

Formulación del problema 6
Hipótesis 6
Estudios piloto 7
Descripción 8
Diseño 8
Tratamiento previo 9
Examen previo 10
Aleatorización 10
Tratamientos 11
Mediciones 13
Análisis estadístico 14

RESULTADOS 14

Resultados del examen previo 14
Resultados del examen subsecuente 15
Relación entre el examen previo y el subsecuente 18
Observaciones 20
Entrevistas 22

DISCUSION Y CONCLUSIONES 26

BIBLIOGRAFIA 30

APENDICES 35

A. Materiales de referencia y tratamiento previo 35
B. Examen previo 47
C. Tratamiento 1 57
D. Tratamiento 2 60
E. Examen subsecuente 63
F. Cuestionario entrevista 73

RESUMEN

El estudio se realizó con el propósito de determinar si programar la computadora mejora la comprensión de los alumnos de los conceptos de cálculo. Se utilizó la computadora de dos maneras para ayudar en el aprendizaje del concepto de derivada. La principal diferencia fue que en uno de los tratamientos los alumnos escribieron sus propios programas; en el otro, los alumnos recibieron programas ya hechos. Los alumnos de los dos tratamientos teclearon y corrieron los programas, analizaron los resultados, y modificaron parámetros y datos. El estudio fue conducido con dos grupos de cálculo de una Facultad de Química de una universidad del país. 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 2x2 (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.

REVISION BIBLIOGRAFICA

El uso de la computadora está modificando lo que percibimos como importante en las matemáticas (Appel y Haken, 1981); en la preparación de los matemáticos (CUPM, 1981); en la enseñanza de las matemáticas

(Conference Board of the Mathematical Sciences, 1982; 1984; Flores Peñafiel, 1986); y 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). La asociación de profesores de matemáticas de E.U., NCTM, recomienda que los programas de matemáticas deberían de utilizar todo el poder de las computadoras en todos los niveles (NCTM, 1980). También dedica el libro del año de 1984 al uso de las computadoras en la educación matemática. 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).

Sin embargo, aunque hay un consenso general entre los matemáticos y los maestros de matemáticas que la computadora tiene un gran potencial en la enseñanza de las matemáticas, hay menos acuerdo acerca de cómo usar la computadora en el salón de clases para realmente mejorar el aprendizaje de las matemáticas. Entre las formas de uso podemos mencionar la enseñanza asistida por computadora, la práctica de destrezas, las simulaciones, la computadora como herramienta, y programar. Taylor (1980) describe varios usos, sus ventajas y sus desventajas y los ha puesto en tres grandes grupos. Un uso es la enseñanza asistida por computadora. En esta forma de uso, el papel de la computadora es guiar al alumno. En esta modalidad la computadora ha sido usada principalmente en actividades de prácticas de destrezas y habilidades computacionales (Suydam, 1984). Una segunda forma de uso es utilizar la computadora como una herramienta. 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).

El tercer grupo de usos es cuando los alumnos programan la computadora.

En este estudio se puso atención a la relación específica que tiene programar con el aprendizaje de las matemáticas.

RELACION ENTRE PROGRAMAR Y DESEMPEÑO MATEMATICO

Varios matemáticos y científicos de la computación han afirmado que programar puede tener un efecto benéfico en el aprendizaje de las matemáticas. Kemeny (1966) menciona que al explicarle cosas a la computadora, esto es, al diseñar un algoritmo, los alumnos entienden mejor los conceptos fundamentales. Knuth (1974) afirma que una persona no entiende realmente algo hasta que puede enseñarselo a una computadora. La lista de matemáticos que afirman que programar la computadora puede tener un efecto benéfico puede extenderse (Birkhoff, 1972; Dijkstra, 1974; Forsythe, 1968; Friske, 1985; McKenna 1972; 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 dedicadas a la educación matemática también han expresado el punto de vista que los alumnos pueden mejorar su comprensión de conceptos matemáticos al programar una computadora. Existe un paralelismo entre el pensamiento necesario para formar, probar, corregir y refinar un programa de computación, y varios aspectos del pentsamiento 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. 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). 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. Varios otros autores han defendido que alumnos de todas las edades y habilidades usen activamente la computadora en el aprendizaje de las matemáticas (NCTM Commitee on Computer Oriented Mathematics, 1965; Bell, 1978; LaFrenz y Kieren, 1969; Hatfield, 1982; Luehrman, 1980; Shumway, 1983).

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.

OTRAS INVESTIGACIONES

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. Para un resumen de otros resultados ver las revisiones bibliográficas de Flores Peñafiel (1985) y de Shumway (1985a; 1985b).

Aunque los resultados no son concluyentes, hay suficiente evidencia para que sea interesante la línea de investigación sobre si programar la computadora tiene un efecto en el aprendizaje de las matemáticas.

Más aún, es necesario realizar investigaciones para entender el papel que desempeña la programación al estudiar un tema de matemáticas. La programación, que ofrece una oportunidad para generalizar, ¿mejora la habilidad de generalizar de los estudiantes? Las dificultades de programar, ¿interfieren con la comprensión de un concepto? ¿Entienden los alumnos mejor porque tienen que escribir el programa, porque ven los resultados o porque escriben el programa y ven los resultados? Estudiar dos formas de usar la computadora, donde la diferencia principal entre

ellas es escribir los programas, puede ayudarnos a entender la contribución de la programación en el aprendizaje.

METODOS Y PROCEDIMIENTOS

FORMULACION DEL PROBLEMA

El propósito del estudio fue determinar si programar la computadora mejora la comprensión de los alumnos de conceptos de cálculo. 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 teclearon los programas, analizaron los resultados y modificaron parámetros y datos.

HIPOTESIS

La hipótesis nula es que no hay diferencia en el aprovechamiento en cálculo entre:

1) alumnos que llevan un curso donde ellos escriben, corren y modifican sus propios programas;

o,

2) alumnos que llevan un curso de cálculo donde se les proporcionan programas ya hechos, y los alumnos tienen que cambiar parámetros y datos.

SUPOSICIONES

1. El lenguaje de programación usado en este estudio (BASIC) no afecta los resultados, esto es, cualquier efecto debido a la programación en el aprendizaje del cálculo se puede obtener usando cualquier otro lenguaje de programación de alto nivel.
2. La marca o modelo de la computadora no es importante, mientras pueda graficar con una resolución de 250 x 150 aproximadamente.
3. Como para los alumnos de los dos tratamientos la computadora es una experiencia nueva, cualquier efecto debido a la novedad afecta ambos grupos de la misma manera.

ESTUDIOS PILOTO

Antes del experimento se condujeron dos estudios piloto. Un experimento informal de enseñanza con alumnos de licenciatura de matemáticas sirvió para seleccionar los programas para el presente estudio (Flores Peñafiel, 1984c). Otra experiencia con alumnos de cálculo de bachillerato sirvió para desarrollar un conjunto de materiales de referencia y de actividades para familiarizar a los alumnos con el lenguaje BASIC. Estas actividades, ligeramente modificadas constituyeron el tratamiento previo del experimento. Así mismo se desarrollaron el examen previo, y actividades para ambos tratamientos. Durante esta experiencia también se probaron algunas de las preguntas del examen subsecuente. La experiencia ganada durante el estudio piloto sirvió de base para tomar decisiones concernientes con los métodos y procedimientos del experimento.

DESCRIPCION

El estudio fue conducido con dos grupos (29 y 26 alumnos) de estudiantes de química de primer año en un curso de cálculo en una universidad del centro del país. Se utilizaron 12 computadoras Timex-Sinclair 2068 (72 K de memoria). Cada grupo tenía tres horas de teoría a la semana, y una sesión con la computadora de dos horas a la semana. Un matemático dio la clase en el turno de la mañana. El experimentador (también un matemático) dió la clase en el turno de la tarde, en ambos casos la teoría fue una clase de tipo tradicional. El experimentador supervisó a ambos grupos en las actividades de computación. Para las sesiones de computación cada grupo fue dividido en dos secciones, de modo que no hubiera más de 15 alumnos con las 12 computadoras.

DISEÑO

El diseño queda ilustrado en la figura 1.

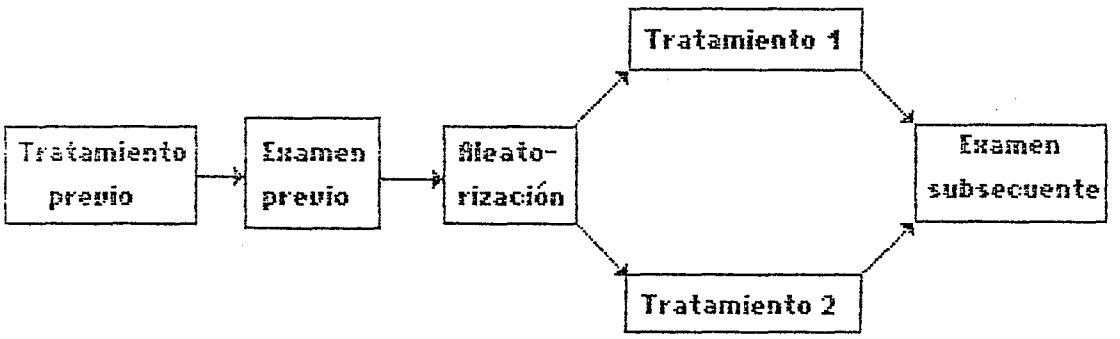


Figura 1 Diseño del experimento

Esto es, todos los alumnos tuvieron el mismo tratamiento previo, el mismo examen previo, fueron asignados aleatoriamente al tratamiento 1 o al tratamiento 2, y presentaron el mismo examen subsecuente.

TRATAMIENTO PREVIO

Antes de participar en el experimento, todos los alumnos tuvieron un curso de BASIC, y de cómo usar la computadora para aprender matemáticas. Se les proporcionaron materiales de referencia tales como:

Conociendo el tablero

Tabla traducida de comandos

Tabla resumida de comandos

Reportes de la computadora

Ver Apéndice A para un ejemplo de estos materiales.

El entrenamiento fue con temas matemáticos que no estaban directamente relacionados con los temas del experimento (cálculo). El contenido matemático del tratamiento previo fue de temas tales como aritmética, álgebra, geometría analítica y probabilidad.

La duración del tratamiento previo fue de cuatro semanas. Los programas de computación eran muy cortos, de modo que la programación en sí no fue enfatizada. Cada programa ilustraba un concepto matemático clave. Por ejemplo, el siguiente programa se puede ~~Se~~ modificar para mostrar la relación entre sumas repetidas y la multiplicación.

```
10 FOR N = 1 TO 12
20 PRINT N + N + N
30 NEXT N
```

Cambio: `20 PRINT 3 * N`

Los programas para graficar ilustran la idea clave de la geometría analítica: la asociación de pares ordenados de números con puntos en el plano, y de ecuaciones algebraicas con curvas. Con el siguiente programa, los alumnos pueden explorar las funciones lineales:

```
5 INPUT "coeficiente "; A
10 FOR X = 0 TO 255
20 PLOT X , A*X
30 NEXT X
```

La lista de las actividades del tratamiento previo es la siguiente:

Contando con la computadora

Programas para graficar

Cambio de escala y de coordenadas

Traslación del origen

Ciclos anidados

Otras formas de iterar

Procesos aleatorios

Ver Apéndice A para un ejemplo de estas actividades.

EXAMEN PREVIO

Ambos grupos presentaron un examen previo. Tal examen medía la familiaridad de los alumnos con los comandos y la sintaxis del BASIC a través de su

a) habilidad para recordar programas dados;

b) habilidad para corregir programas;

c) capacidad para predecir el resultado de un programa dado;

d) capacidad para escribir un programa que diera un resultado dado.

Ver el Apéndice B para el examen previo.

ASIGNACION ALEATORIA A TRATAMIENTO

Los alumnos fueron asignados a un tratamiento o al otro conforme iban entrando al laboratorio, de acuerdo a una lista de números aleatorios

generada por la computadora. La distribución de acuerdo a grupo y tratamiento quedó como sigue:

	1	2
A	n = 13	n = 16
B	n = 14	n = 12

Figura 2 Tamaños de las celdas

TRATAMIENTOS

El laboratorio de computación estaba dividido en dos salas, cada una con seis computadoras, una sala para cada tratamiento, separadas por una pared de vidrio y conectadas por una puerta. Se aislaron los tratamientos uno del otro, aunque hubo interacción entre los sujetos del mismo tratamiento. Las actividades para ambos tratamientos fueron escritas y supervisadas por el experimentador. Los alumnos tenían acceso a estos materiales únicamente durante las sesiones de laboratorio. De este modo se controló el tiempo dedicado a las actividades. Los sujetos en ambos tratamientos dedicaron el mismo tiempo (tres sesiones de laboratorio) a actividades relacionadas con derivada y con límite. El experimentador aclaró dudas de los alumnos, observó el comportamiento de los alumnos y controló que los dos tratamientos permanecieran ajenos.

Las semejanzas y diferencias entre los tratamientos están resumidas en la tabla 1.

Tabla 1 diferencias entre los tratamientos

	Tratamiento 1	Tratamiento 2
Escribir programas	sí	no
Teclear programas	sí	sí
Correr programas	sí	sí
Modificar programas	sí (propios)	sí (dados)
Analizar resultados	sí	sí
Datos dados	sí	sí
Escoger datos	sí	sí
Modificar parámetros	sí	sí
Tiempo en el laboratorio	6 horas	6 horas
Promedio de actividades hechas	3.3	4.3
Cantidad de ejemplos	menos	más
Preguntas hechas (programación)	más	menos

Para tomar en cuenta la diferencia en la rapidez de trabajo de los alumnos, éstos fueron provistos de actividades adicionales que usaban el concepto estudiado pero que no lo profundizaban más. Estas actividades están señaladas con un asterisco*. La lista del total de la actividades proporcionadas a los alumnos es la siguiente:

Velocidad promedio y velocidad instantánea

Método de incrementos

Cociente de incrementos y funciones relacionadas

Derivación numérica*

Encontrando la aproximación deseada

Encontrando la aproximación deseada a un límite.

Un ejemplo de una actividad del tratamiento 1 aparece en el apéndice C, un ejemplo de la misma actividad para el tratamiento 2 aparece en el apéndice D.

MEDICIONES

El desempeño de los alumnos fue medido por su comprensión del concepto de derivada, especialmente los siguientes aspectos:

a) velocidad instantánea como el límite de la velocidad promedio;

b) la pendiente de la tangente como el límite de la pendiente de las secantes;

c) la derivada como el límite del cociente de incrementos.

La relación de los conceptos puede representarse como sigue:

Concepto general:	Límite
Concepto central:	Derivada
Conceptos particulares:	Velocidad instantánea
	Pendiente de la tangente como límite de pendientes de secantes

La medición consistió de un examen escrito. Los estudiantes tuvieron que contestar 26 preguntas de opción múltiple y completar dos dibujos. El examen consistió de tres secciones, una relacionada con velocidad, otra con tangente y la otra con la derivada como un límite.

Tres matemáticos del Centro de Investigación en Matemáticas verificaron la validez de contenido del examen. Una versión previa del examen fue probada con alumnos de una licenciatura en matemáticas un año antes. La versión definitiva del instrumento fue probada con 17 alumnos de matemáticas. El coeficiente KR-20 obtenido fue 0.72.

Además del examen escrito, se hicieron observaciones en el laboratorio y se condujeron entrevistas con alumnos individuales. Las observaciones fueron sobre el tiempo dedicado a programar, tiempo dedicado a correr los programas con diferentes datos, y las interacciones entre los estudiantes. Se escogieron cuatro alumnos al azar de cada uno de los tratamientos para realizar las entrevistas, a fin de obtener una idea de los procesos de pensamiento de los alumnos al escribir y correr los programas y cómo los relacionaban con los conceptos matemáticos estudiados.

ANALISIS ESTADISTICO

Aunque el interés principal era comparar los tratamientos, se efectuó un análisis de varianza de 2 x 2 (tratamiento por grupo) sobre los resultados del examen, para detectar posibles interacciones y aumentar el poder del análisis. Se efectuó otro análisis de varianza usando los resultados del examen previo para formar bloques.

Las observaciones y las entrevistas no fueron cuantificados.

RESULTADOS

Primero se presentarán los resultados del examen previo, del examen subsecuente, y la relación entre estos dos. La información cualitativa de las observaciones y de las entrevistas será presentada después.

RESULTADOS DEL EXAMEN PREVIO

Un análisis de varianza de los resultados del examen previo no mostró diferencias por celda o por efectos principales. Ver tabla 2 para este análisis de varianza. El patrón de los resultados de las celdas se muestra en la figura 3.

Tabla 2 Análisis de Varianza Examen Previo

Fuente	grad. lib.	SC	MC	F
T	1	12	12	0.77
G	1	2.1	2.1	0.13
TG	1	0.8	0.8	0.05
S/TG	44	686.8	15.6	
Total	47	701.7	30.5	

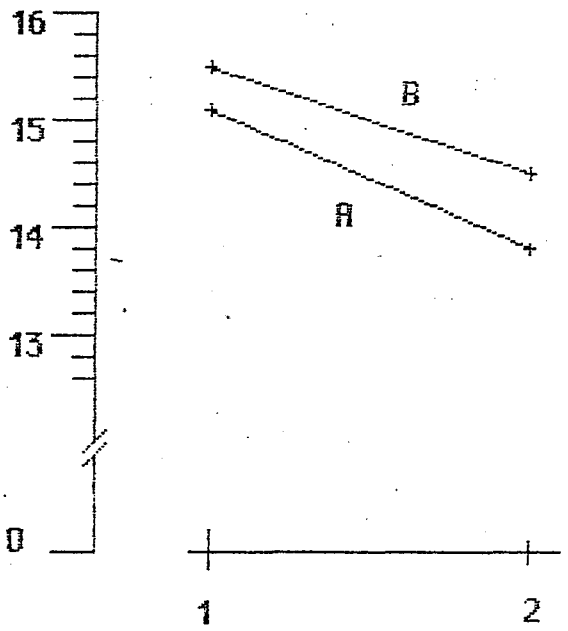


Figura 3 Resultados del examen previo

RESULTADOS DEL EXAMEN SUBSECUENTE

Los alumnos presentaron un examen de cálculo una semana después de que terminaron los tratamientos. Los resultados del análisis de las celdas se muestran en la tabla 3. El patrón de los resultados de las celdas se muestra en la figura 4.

Tabla 3 Resultados del examen subsecuente

	Tratamiento 1 programar	Tratamiento 2 no programar
Grupo A	n = 13 $\bar{x}$ = 19.5 s = 2.9	n = 16 $\bar{x}$ = 18.1 s = 2.5
Grupo B	n = 14 $\bar{x}$ = 16.5 s = 3.7	n = 12 $\bar{x}$ = 16.6 s = 5.1

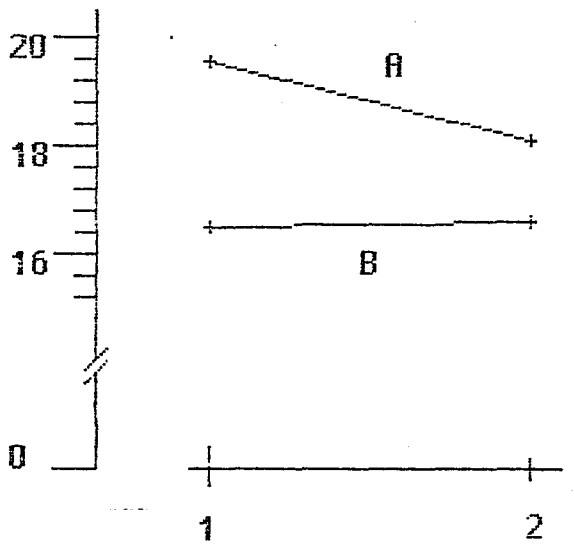


Figura 4 Resultados del examen subsecuente

Los resultados muestran un patrón similar a los resultados que se obtuvieron en el estudio piloto (ver tabla 4 y figura 5). En ambos casos, en el grupo con mejores resultados en el examen de cálculo, los programadores tuvieron un mejor desempeño que los no programadores.

Tabla 4 Resultados piloto

	Programar 1	No programar 2
grupo A	$n = 7$ $\bar{x} = 7.3$ $s = 3.5$	$n = 4$ $\bar{x} = 4.3$ $s = 5.9$
grupo B	$n = 9$ $\bar{x} = 4.2$ $s = 3.6$	$n = 8$ $\bar{x} = 3.9$ $s = 4.0$

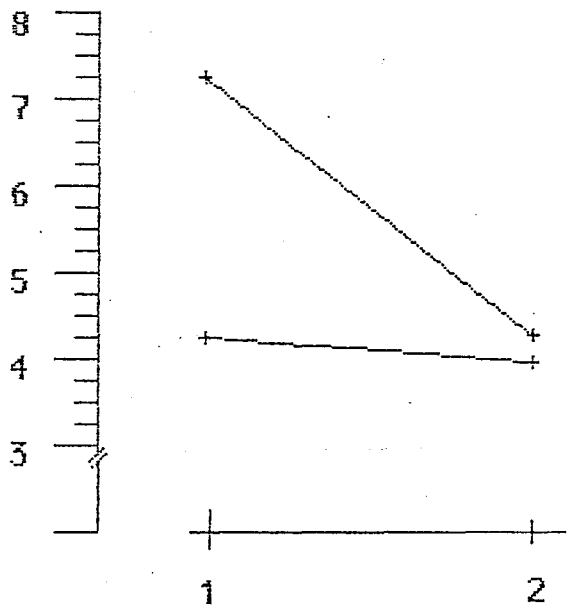


Figura 5 Resultados piloto

Usando sólo los datos del grupo A en este estudio, una prueba t dio un valor de α entre 0.05 y 0.10, lo cual es una indicación de que en estudios posteriores la habilidad matemática puede ser usada como un factor. En este estudio no hubo un examen común de matemáticas para ambos grupos que pudiera ser usado como una medida de habilidad matemática. Se efectuó un análisis de varianza de 2 x 2. No fue estadísticamente significativa la diferencia entre los tratamientos ni la interacción. La tabla 5 muestra los resultados.

Tabla 5
Análisis de Varianza Examen Subsecuente

Fuente	grad. lib.	SC	MS	F
T	1	3.9	3.9	0.29
G	1	65.5	65.5	4.84*
TG	1	11.7	11.7	0.86
S/TG	51	689.6	13.5	
Total	54	770.7		

* $\alpha < 0.05$

RELACION ENTRE EL EXAMEN PREVIO Y EL SUBSECUENTE

La correlación entre el examen previo y el examen subsecuente fue de 0.27. El diagrama de dispersión de estas dos mediciones se muestra en la figura 6.

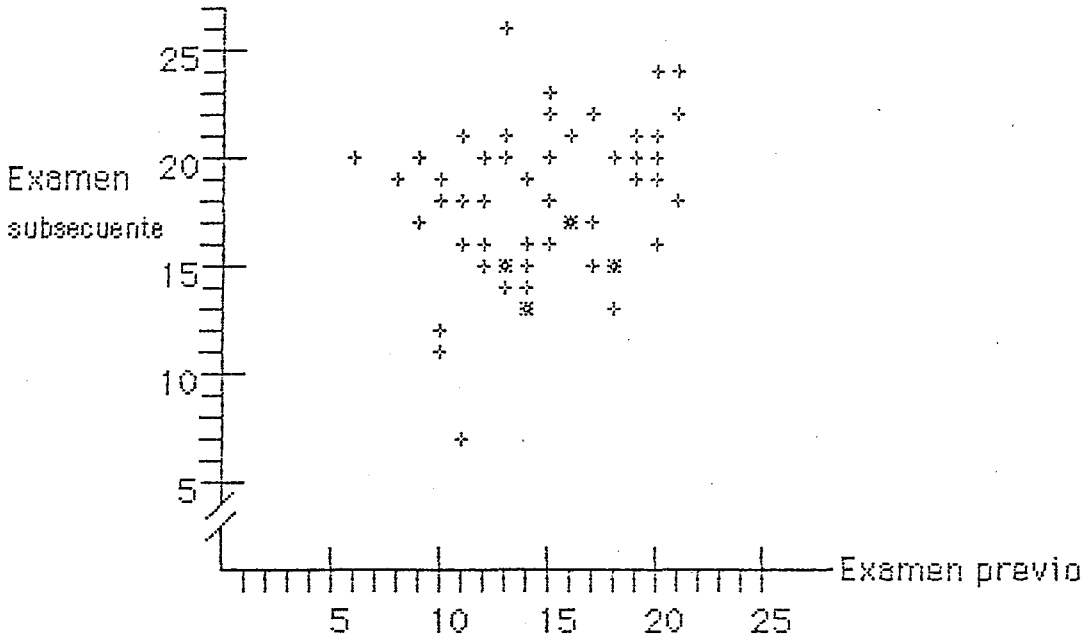


Figura 6 Diagrama de dispersión

Los resultados de los estudiantes de cada tratamiento fueron acomodados por bloques de acuerdo al desempeño en el examen previo, según hubieran estado en la mitad superior o inferior de la escala. La tabla 6 muestra los resultados de un análisis de varianza de 2 x 2 (agrupando por habilidad para programar según el examen previo). No fue estadísticamente significativa la diferencia entre los tratamientos ni la interacción.

Tabla 6 Análisis de Varianza Tratamiento por Habilidad para Programar

Fuente	grad. lib.	SC	MC	F
T	1	7.9	7.9	0.62
H	1	28.2	28.2	2.20
TH	1	13.0	13.0	1.01
S/TH	50	640.4	12.8	
Total	53	689.5		

De acuerdo con los resultados obtenidos en ambos análisis de varianza (tratamiento por grupo, y tratamiento por habilidad para programar) no se rechaza la hipótesis nula.

OBSERVACIONES

Las siguientes observaciones fueron hechas por el experimentador. No se hizo ningún intento por cuantificarlas. Esta información y la información obtenida en las entrevistas es anecdótica y se ofrece para que el lector tenga una idea de lo que los alumnos hacían y experimentaban durante las sesiones de laboratorio.

Familiaridad con el equipo de cómputo. Los estudiantes superaron facil y rapidamente cualquier sentimiento de incomodidad o desconfianza con las computadoras. Los alumno trabajaron en grupos pequeños (tres o cuatro alumnos) en las primeras sesiones del tratamiento previo, luego por parejas o solos.

Familiaridad con los comandos en BASIC. Al finalizar el tratamiento previo la mayor parte de los alumnos sabían qué comandos usar, aunque algunas

veces era sólo por imitación de actividades semejantes.

Entusiasmo. Los alumnos permanecieron entusiasmados en las sesiones durante todo el semestre, aunque la asistencia fue un poco menor hacia el final (ya se habían concluído los tratamientos).

Habilidad para escribir programas cortos. La mayor parte de los alumnos hacían el esfuerzo por entender el problema antes de escribir cualquier cosa. Sin embargo, había poca planeación de su parte. No todos los alumnos escribían programas elegantes, por ejemplo, algunos escribían un programa con un ciclo infinito, en vez de usar un ciclo controlado. Sin embargo, este hecho no parecía interferir con la comprensión del concepto matemático. Los programas eran suficientemente cortos así que las consideraciones de cómo estructurarlos no eran de primordial importancia. El supervisor se aseguró que todos los alumnos hubieran escrito un programa antes de terminar la sesión y que lo corrieran al menos un par de veces para ver diferentes ejemplos. El supervisor proporcionaba tantas pistas como necesitaran los alumnos para escribir el programa.

Documentación. Los programas que se proporcionaban a los alumnos en el tratamiento 2, estaban mejor documentados que los programas que los alumnos escribían en el tratamiento 1. Sin embargo, conforme los programas se iban haciendo más complejos, mejoraba el cuidado de los alumnos para documentarlos.

Errores. Pocos alumnos se daban cuenta, al ver el resultado en la pantalla, que el programa tenía un error, a menos que apareciera un mensaje de error, o el programa nunca se detuviera. Muchos de los alumno anotaban los resultados erróneos, a menos que el supervisor notara el error y lo señalara a los alumnos. 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. Algunos pocos alumnos escribían de manera espontánea programas más generales o poderosos que los que se les pedían, en general rara vez trataban los alumnos de escribir un programa más general a menos que se les indicara.

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.

Interacción entre los estudiantes. Dentro del mismo tratamiento, era frecuente la interacción entre los estudiantes, debido en parte a que algunos de ellos tenían que compartir una computadora. Cuando tenían problemas o aparecía un error, los alumnos preguntaban a sus vecinos, así como al supervisor.

Preguntas. Durante la sesión de laboratorio los alumnos hacían muchas preguntas, muchas más de las que usualmente se preguntan en una clase normal. La mayor parte de las preguntas estaban relacionadas con la programación, más que con el contenido matemático del programa.

ENTREVISTAS

Para comprender mejor los procesos de pensamiento de los alumnos durante la experiencia con la computadora, se entrevistaron algunos alumnos. Estas entrevistas se hicieron cuatro semanas después de terminado el experimento. La razón de retrasar las entrevistas fue dar a los alumnos la experiencia alternativa, es decir, los estudiantes que durante el experimento tuvieron que escribir sus propios programas, recibieron después durante tres semanas actividades con los programas ya

escritos; los alumnos que durante el experimento recibieron programas ya escritos, tuvieron luego que escribir sus propios programas. Los temas cubiertos en esta parte del curso fueron aproximación lineal; aplicaciones de la aproximación lineal, tales como el método babilónico para extraer raíces cuadradas, el método de Newton para encontrar ceros de funciones; y otros procesos iterativos, tales como el método de bisección.

Se escogieron ocho alumnos (dos de cada celda) al azar para responder a un cuestionario, aunque de hecho sólo se presentaron a la entrevista cuatro alumnos del tratamiento 1 y dos del tratamiento 2. El cuestionario aparece en el apéndice F. Antes de hacer las preguntas, el entrevistador explicó que la finalidad de la entrevista era aprender de esta experiencia de laboratorio para corregir errores con miras a futuras experiencias y saber lo que los alumnos percibían.

Cinco de los alumnos prefirieron contestar las pregunta verbalmente, el otro por escrito. La impresión del entrevistador es que los alumnos tomaron en serio su papel y contestaron honestamente. Ninguno de los alumnos entrevistados había tenido experiencia previa con la computadora.

Lo que sigue es un resumen de las percepciones de los alumnos. Primero se presentarán la apreciaciones que fueron comunes, independientemente del tratamiento que les haya tocado. Luego se presentarán las respuestas agrupadas de acuerdo al tratamiento.

Todos expresaron una actitud positiva con respecto a la idea de usar las computadoras en cálculo, por la oportunidad de aprender a usarlas al principio de su carrera.

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.

El hecho de que los comandos de BASIC estuvieron en inglés no representó un problema. Las características del teclado de la Timex-Sinclair no causaron ningún problema.

Cuando se les preguntó qué tipo de actividades preferían, aquellas donde ellos tenían que escribir programas o aquellas en las que se les daban programas ya hechos, cinco de los seis contestaron que preferían escribir los programas. La razón era que ellos sentían que aprendían más acerca de la programación. Ninguno mencionó que aprendiera más cálculo.

Las computadoras ayudaron a los alumnos a percibir las matemáticas como menos abstractas, más fáciles (cálculos), tener aplicaciones, y a ver ejemplos.

Ninguno de los seis alumnos mencionó que la computadora les ayudara a percibir aspectos exploratorios o de descubrimiento en las matemáticas. Ninguno dijo que hubiera cambiado su forma de estudiar cálculo debido a la computadora. Ninguno de los seis alumnos mencionó ningún concepto de cálculo que hubiera entendido mejor por haber tenido que escribir un programa. Ninguna dificultad específica de cálculo fue mencionada por los alumnos para escribir un programa.

Los alumnos no tenían una estrategia clara y bien estructurada para escribir programas. Todos los estudiantes entrevistados querían que hubiera laboratorio de computación en su siguiente clase de cálculo. Cuando se les preguntó qué era lo que más les gustaba del laboratorio, tres contestaron que aprender a usar la computadora, dos mencionaron graficación y uno la rapidez con la cual se puede ejecutar todo. Ninguno mencionó ningún tema o concepto directamente relacionado con derivada.

Los cambios sugeridos por los alumnos fueron: dar más explicaciones antes las actividades, enseñar diagramas de flujo, coordinar mejor el laboratorio

con la teoría.

A la pregunta de a cuáles conceptos de cálculo ayudó la computadora para una mejor comprensión, las respuestas de los alumnos del tratamiento 1 fueron:

- velocidad (variación en los resultados)
- límite, derivada (aproximaciones)

Dos estudiantes sentían que el uso de la computadora no había hecho una contribución especial a la comprensión de ningún concepto de cálculo.

Del tratamiento 2 los conceptos mencionados fueron:

- la diferencia entre velocidad promedio y velocidad instantánea (ya que los programas usados para calcular cada una eran diferentes)
- derivada.

De los resultados del examen previo en computación, las observaciones hechas durante las sesiones de laboratorio y los resultados de las entrevista, 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.

DISCUSION Y CONCLUSIONES

Aunque las diferencias en favor de los programadores no fueron significativas, los resultados de este estudio, junto con los resultados del estudio piloto, sugieren hacer más investigación en esta línea. Los resultados indican que la habilidad matemática puede ser un factor a ser considerado en estudios futuros. En dos estudios independientes las diferencias en favor de los programadores fueron mayores en el grupo con mayores calificaciones. Estratificar por habilidad matemática es un medio para ver si esto es cierto en otros grupos. Sin embargo algunas de las suposiciones, y el diseño deben ser reexaminados para mejorar la posibilidad de encontrar diferencias significativas en futuros estudios.

Se esperaba que los alumnos al programar desarrollaran algunos cambios en sus procesos de pensamiento. Algunos de los resultados cognitivos que se esperaban de programar son:

- 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.

Se esperaba que los alumnos que hubieran adquirido o desarrollado estas habilidades entendieran mejor los conceptos de cálculo que encontraran al escribir los programas.

Sin embargo, los alumnos no siempre desarrollaron esas habilidades cognitivas. Por ejemplo, no todos los alumnos se daban cuenta al correr el programa que habían escrito que este tenía un error. Así uno de los posibles beneficios de escribir los programas, el de aprender de los errores, no fue totalmente aprovechado.

No se enseñaron estrategias para resolver problemas explícitamente o de manera sistemática durante el laboratorio. Los alumnos no desarrollaron estas habilidades por sí mismos de manera evidente durante el tratamiento previo o el tratamiento.

Con la variedad de ejemplos para cada programa y las sugerencias para modificar los programas se pretendía dar a los alumnos una idea de cómo explorar y descubrir con la computadora, sin embargo, tal comportamiento fue observado únicamente en algunos alumnos.

Se observó durante el experimento que los programadores tenían problemas no sólo con los aspectos matemáticos del programa, sino también con la programación. Es deseable que antes de empezar el tratamiento los alumnos hayan superado estos obstáculos.

En este experimento los alumnos tenían que escribir programas cortos y sencillos. La hipótesis era que el escribir programas como esos diferenciaría a los programadores de los no programadores. Pea y Kurland (1983) dudan que los estudiantes ganen comprensión de otros dominios sobre la base de habilidades limitadas de programar. Ellos sugieren que se necesita un nivel más alto de habilidades de programación (nivel 4: desarrollo de software) para esperar evidencia amplia de transferencia cognitiva.

Los alumnos en este estudio alcanzaron el nivel dos en habilidad para programar, como se describió en la sección anterior. Como Pea y Kurland afirman, las habilidades que hacen más factible una transferencia cognitiva a otros dominios, son los ricos esquemas de conocimiento, las estrategias, reglas y organización de la memoria que muestran los programadores expertos. Estas habilidades rara vez se enseñan directamente.

Sin embargo es dudoso que todos los alumnos puedan o deban alcanzar el nivel cuatro de habilidad para programar. Tal vez la meta del tratamiento previo sería que los alumnos alcanzaran el nivel tres (generar programas). Los alumnos a este nivel dominan los comandos básicos y empiezan a pensar en términos de unidades á más alto nivel. El estudiante conoce secuencias de comandos que logran las metas de los programas. El estudiante puede leer un programa y explicar su propósito. Puede entender las diferentes partes de un programa y cómo están conectadas. El estudiante puede localizar errores que impidan el funcionamiento apropiado del programa, o errores que lo hagan interrumpan. El estudiante ve la necesidad de documentar el programa, sin embargo el programa no está diseñado para ser usado o mantenido por otros. A este nivel de competencia, los alumnos empiezan a desarrollar cierta apreciación por el proceso de diseñar un programa de computación exitoso. Como Pea y Kurland señalan, tal comprensión tiene en potencia implicaciones importantes para el trabajo de los alumnos en otros dominios, en especial si tales relaciones le son señaladas explícitamente al alumno por el profesor. Una sugerencia para nuevos estudios es hacer la experiencia de programación suficientemente intensa, extensa y estructurada para que los alumnos alcancen al menos el nivel tres de habilidades de programar antes

del tratamiento.

Un tratamiento previo extendido podría abarcar temas tales como graficación de funciones en coordenadas cartesianas, incluyendo cambio de escala y de origen, ecuaciones paramétricas, coordenadas polares, así como los temas que en este estudio se cubrieron durante el tratamiento: velocidad, secantes y tangente a una curva, y la derivada como un caso especial de límite.

Algunos autores afirman que se necesitan tratamientos más largos para esperar diferencias significativas, por ejemplo De Corte señala 50 horas. En este estudio el tratamiento fue de seis horas. Una posibilidad para hacer el tratamiento más extenso es dedicar el primer semestre al tratamiento previo, y hacer el tratamiento durante el segundo semestre (cálculo integral).

BIBLIOGRAFIA

Appel, K., y Haken, W. (1981). The nature of proof: limits and opportunities. Two Year College Mathematics Journal, 12, 116-119.

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, TX.

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.

Committee on the Undergraduate Program in Mathematics. (1981). Recommendations for a general mathematical sciences program. Washington: Mathematical Association of America.

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. Arithmetic Teacher, 32 (8), 14-18, 60.

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

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.

Dugger, 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.

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 de la Sociedad Matemática Mexicana, Mérida, Yucatán.

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

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

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

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

Hansen, Viggo P., y Zweng, Marilyn. (1984). Computers in mathematics education: 1984 Yearbook. Reston, VA: National Council of Teachers of Mathematics.

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

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.

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, 3463.

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

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.

McKenna, J. E. (1972). Computers and experimentation in mathematics.

American Mathematical Monthly, 79, 294-295.

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.

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

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

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.

Shumway, Richard J. (1984). Exploring computer programming in BASIC with kindergartners. En Moser, J. (Ed.) Proceedings of the Sixth Annual Meeting of PME-NA. Madison, Wisconsin: PME.

Shumway, Richard J. (1985a). Computer programming research (draft bibliography). Manuscrito no publicado. Centro de Investigación en Matemáticas, Guanajuato.

Shumway, Richard J. (1985b). Mathematical concept learning through computer programming: a survey of related research (draft copy). Manuscrito no publicado. Centro de Investigación en Matemáticas, Guanajuato.

Shumway, Richard J. (1987). 101 ways to learn mathematics using BASIC

(K-8). Englewood Cliffs: Prentice Hall.

Smith, David 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.

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.