“Nivel de utilización de las técnicas de diagramación por parte de los estudiantes, en el diseño de algoritmos secuenciales, selectivos e iterativos”

Área de Conocimiento: Computación Educativa

Isaías Pérez Pérez [1], Citlali Anahí Monzalvo López [2]

[1] Universidad Autónoma del Estado de Hidalgo.
Instituto de Ciencias Básicas e Ingeniería
Carr. a Tulancingo s/n. Mineral de la Reforma, Hidalgo. México
e-mail: isaiasp@uaeh.edu.mx, isaiaspp7@hotmail.com

[2] Universidad Autónoma del Estado de Hidalgo
Instituto de Ciencias Básicas e Ingeniería
Carr. a Tulancingo s/n. Mineral de la Reforma, Hidalgo. México
e-mail: lex_any@hotmail.com

Resumen: *El presente trabajo exhibe la continuación del análisis de los resultados obtenidos en los estudios presentados por Pérez Pérez y Monzalvo López (2010), el cual exploró la problemática presente en el diseño de los algoritmos concebidos dentro del paradigma de la programación estructurada. Para ello, se diseño y aplicó un instrumento de evaluación que entre varios aspectos, realizó un análisis del nivel de utilización de las técnicas de diagramación, en el diseño de algoritmos con sentencias secuenciales, selectivas e iterativas, con el propósito de observar que tan importantes son estas técnicas en el diseño de algoritmos para los estudiantes de las asignaturas de programación estructurada.*

Palabras clave: *Programación Estructurada, diseño de algoritmos, técnicas de diagramación.*

Introducción

La Programación Estructurada, como lo menciona Joyanes (1990), se define como *“el conjunto de técnicas para desarrollar programas fáciles de escribir, verificar, leer (legibles para el usuario) y mantener (posibilidad de modificar)”.* Cuando se diseñan algoritmos, una de las técnicas más utilizadas, son las llamadas herramientas de diagramación de algoritmos, que buscan hacer una representación grafica de los mismos, con la finalidad de ofrecer un mejor entendimiento y su adecuada documentación.

Estado del arte y problemática presentada

Las técnicas de diagramación como los diagramas de flujo, el pseudocódigo, entre otras, son esquemas representativos además de un lenguaje para expresar algoritmos. El aprender a utilizar estas técnicas de diagramación es relativamente sencillo; lo que resulta más difícil es aprender a utilizarlas como herramientas efectivas para expresar ideas. Esto llega solo con la práctica (Forsythe, Skeenan, Organick, Stenberg, 1975).

El presente documento continúa con el estudio realizado por Pérez Pérez y Monzalvo López (2010), concretamente sobre el análisis de los comentarios generados por parte de los estudiantes, sobre la utilización de las diversas técnicas de diagramación disponibles, dentro del diseño de algoritmos de tipo secuenciales-selectivos e iterativos. El motivo de haber seleccionado estos dos tipos de problemas, es porque reflejan diferentes niveles de dificultad en la construcción de sus algoritmos, ya que los secuenciales-selectivos siguen un cierto patrón estructural y de funcionamiento diferente al de los algoritmos cíclicos; éstos últimos se consideran, en la experiencia cotidiana docente, difíciles de entender y desarrollar por parte de los estudiantes.

Las sentencias secuénciales y de selección son proposiciones que aunque controlan el flujo de los algoritmos, este solo se da de un punto (inicio) a otro (fin), sin que haya retrocesos o regresos en la secuencia de las instrucciones. En cambio, las sentencias cíclicas, manipulan el flujo del algoritmo de una manera más compleja, ya que las sentencias pueden empezar en un punto (entrada al ciclo), seguir un numero de pasos específicos y según la condición lógica que controla el ciclo, este puede volver a empezar o terminar para seguir con la secuencia de normal del algoritmo. Las sentencias cíclicas, desde el punto de vista de cómo se estructuran las sentencias, se pueden concebir como submódulos, dentro del algoritmo total, ya que presentan algunas entradas al ciclo, un proceso iterativo y una salida del mismo (Pérez Pérez y Moreno Gutiérrez, 2009).

Metodología o técnica usada

La metodología que sigue el presente estudio (planteada inicialmente en la investigación de Pérez Pérez y Monzalvo López (2010)), es la siguiente:

a) Selección de dos problemas tipo a plantear, definiendo cuidadosamente los enunciados de éstos

b) Planteamiento de los enunciados de los problemas, a un grupo de estudiantes de la asignatura de programación estructurada

c) Recolección de los resultados obtenidos

d) Análisis y generación de las conclusiones, derivadas de los resultados obtenidos

Resultados experimentales

Los dos enunciados de los problemas seleccionados, se aplicaron a 23 estudiantes de la materia de programación estructurada, de primer semestre, de la carrera en Ingeniería en Electrónica y Telecomunicaciones, impartida en la Universidad Autónoma del Estado de Hidalgo. Después del análisis del desempeño de los estudiantes en dicha prueba, se obtuvieron los siguientes resultados:

Se llevo a cabo un análisis de los aspectos más significativos a los estudiantes encuestados, sobre el diseño de algoritmos de tipo secuenciales-selectivos e iterativos, con respecto a la utilización de las diversas técnicas de diagramación disponibles. Como lo mencionaba el estudio de Pérez Pérez y Monzalvo López (2010), una cuarta parte de los estudiantes (26.1%), se les clasificaba en un nivel de desempeño alto, en cuanto al aspecto de diseñar algoritmos secuenciales-selectivos e iterativos. Poco más de la mitad de los estudiantes (52.2%), se clasificarían en un nivel de desempeño medio, y finalmente, poco menos de la cuarta parte (21.7%), serian considerados en un nivel de desempeño bajo.

Los resultados de las entrevistas a los estudiantes, sobre sus acciones y las opiniones subjetivas que aportaron, sobre sus experiencias con los problemas de tipo secuencial-selectivo y cíclicos presentados, son las siguientes:

a) Algoritmo Secuencial-Selectivo. El enunciado del problema se muestra a continuación. Algunos resultados obtenidos se muestran en la tabla 1:

"Construya un algoritmo que permita como entrada, una cantidad en grados Farenheit (°F), y que calcule su equivalente en grados centígrados (°C), por medio de la fórmula:

$$°C = 5/9 (°F - 32)$$

El algoritmo deberá dar como resultado la temperatura en grados centígrados y determinar si corresponde a alguno de los siguientes niveles:

Frío (temperatura menor a 10 °C)
Tibio (temperatura entre 10 °C y 50°C)
Caliente (temperatura mayor a 50 °C)

Nota: Utilice solo sentencias ***secuenciales y selectivas."***

b) Algoritmo Cíclico o Iterativo. El enunciado del problema se muestra a continuación. Algunos resultados obtenidos se muestran en la tabla 2:

"Desarrolle un programa que utilizando la función f(x) = x^2 – 2x + 1, genere como salida, la tabla de valores: x, f(x) en pantalla, la cual sólo comprende los valores de f(x) en el intervalo de -5 a 20; y para x, en el intervalo de los valores de -5 a 5 con pasos de 1.

Nota: Utilice sentencias ***secuenciales, selectivas y cíclicas."***

Nivel de desempeño del subgrupo de estudiantes	**Facilidad en el diseño del algoritmo (0 = muy fácil, 10 = muy difícil)**	**Técnica de diagramación utilizada**	**Forma de desarrollo del algoritmo**
Alto (puntuación de 60 a 100)	1.5 (sencillo)	Tablas NS	Se construye de forma secuencial, de manera que cada elemento se concibe como "los eslabones de una cadena"
Medio (puntuación de 20 a 60)	0 (muy fácil)	Pseudocódigo	(Igual que el anterior)
Bajo (puntuación de 0 a 20)	0 (muy fácil)	Ninguna (el algoritmo se codifica directamente en el lenguaje de programación)	(Igual que el anterior)

Tabla 1. Resultados de los estudiantes sobre el algoritmo secuencial-selectivo.

Los comentarios más relevantes, recogidos de las opiniones de los estudiantes, sobre el uso y utilidad de las herramientas de diagramación en el diseño de algoritmos, son las siguientes:

1) El uso de las técnicas de diagramación en el diseño de algoritmos largos, complejos o cíclicos, es complicado, difícil de aplicar y consume demasiado tiempo. Según los estudiantes, estas técnicas no ayudan de mucho a diseñar o comprobar un algoritmo. Consideran que éste se debe concebir inicialmente en la mente del diseñador y las técnicas de diagramación solo sirven para plasmarlo de manera gráfica, con el fin de aclarar y definir los detalles faltantes en el diseño, pero la "esencia del algoritmo", lo que éste recibe, procesa y da como resultado, se debe concebir con anterioridad.

Nivel de desempeño del subgrupo de estudiantes	Facilidad en el diseño del algoritmo (0 = muy fácil, 10 = muy difícil)	Técnica de diagramación utilizada	Forma de desarrollo del algoritmo
Alto (puntuación de 60 a 100)	5.5 (regular)	Ninguna (su uso es confuso, tardado e impráctico).	Sólo se conciben las posibles sentencias que contendrá, pero no se tiene claro cuál es la secuencia que siguen, su interrelación y el encadenamiento de unas con otras.
Medio (puntuación de 20 a 60)	5 (regular)	Ninguna. Las técnicas no se consideran útiles. Se codifica el algoritmo directamente en el lenguaje de programación.	Los algoritmos se construyen, poniendo en secuencia las diversas sentencias, pero al tratar de incluirles sentencias cíclicas, su diseño se dificulta.
Bajo (puntuación de 0 a 20)	2.5 (difícil)	No usaron alguna técnica de diagramación. Trataron de resolver el problema mentalmente, apoyándose en una representación informal del problema (tabla de valores) y codificaron directamente en lenguaje de programación, el algoritmo.	Cuando se diseñan los algoritmos, se visualizan algunas de las sentencias necesarias, pero sin relación entre ellas. Al plasmarlas concretamente en el código, se les va encontrando relación, así como define el orden en que deben estar acomodadas.

Tabla 2. Resultados de los estudiantes sobre el algoritmo cíclico o iterativo.

2) Según la opinión de algunos estudiantes, si los algoritmos se diseñan directamente en el código fuente de un lenguaje de programación, sin utilizar

alguna técnica de diagramación, es posible definir de manera más fácil la secuencia de instrucciones necesarias y la forma en que van relacionadas y ordenadas. Explican que esto se debe a dos razones: a) porque en la mente del diseñador ya se tiene la concepción del algoritmo a desarrollar, y b) porque el uso del compilador del lenguaje específico, permite llevar a cabo un método de prueba y error, el cual se basa en retroalimentar constantemente el diseño para mejorarlo, logrando que después de varias iteraciones de prueba–error, se obtenga un algoritmo correcto y funcional. El uso del código fuente directamente en el diseño de los algoritmos, permite rectificar la lógica del diseño en base a los errores que surgen, además de que permite visualizar más rápida y eficientemente, el diseño global del algoritmo.

3) Según los estudiantes, un algoritmo cíclico es más complicado de diseñar debido a que no solo consiste en una secuencia de pasos o sentencias en forma encadenada, como es el caso de un algoritmo secuencial-selectivo. En un algoritmo iterativo, hay que diseñar ciclos, los cuales implican desarrollar un grupo de sentencias ordenadas y vinculadas entre sí que comprenden una unidad en sí misma y que tienen un objetivo particular, lo cual no es una tarea fácil en muchos casos, ya que se necesita saber cuándo se utilizará alguno de los tipos y las variantes que presentan las sentencias cíclicas (for, do-while, while, etc.), dentro del contexto del algoritmo; pero este bloque de código cíclico es a su vez, un elemento más que colabora con el objetivo general del algoritmo, junto con las demás sentencias que lo componen; esto provoca que además de lidiar con la complejidad que genera el grupo de sentencias con características iterativas, hay que lidiar con la complejidad de todo el algoritmo, cosa que hace que el diseño total se convierta en una tarea bastante "embrollada".

Los resultados obtenidos, muestran que los algoritmos secuenciales-selectivos son relativamente fáciles de concebir por los estudiantes. Esto posiblemente se puede deber a que las personas piensan de manera muy similar a como estas sentencias se estructuran; no así en el caso de los problemas cíclicos, en donde los resultados son poco alentadores. Es evidente que en el uso de las sentencias repetitivas es en donde los estudiantes tienen más problemas al tratar de implementarlas en el diseño de algoritmos. Esto es debido probablemente a que las personas no tienden a pensar en forma iterativa. Scheid (1984), menciona a este respecto: *"El hombre a menudo encuentra aburrido repetir, pero los computadores no comparten esta emoción. En efecto, es precisamente en esto donde sobresalen. Desarrollando algoritmos que hagan uso eficiente de la repetición, las máquinas se pueden programar para que hagan trabajos a gran escala"*.

Conclusiones

Para comenzar, se debe decir (Pérez Pérez y Monzalvo López, 2010) que las técnicas y herramientas de diagramación clásicas para el diseño de algoritmos (diagramas de flujo, tablas N-S, pseudocódigo), parecen adecuarse adecuadamente a la forma de concebir los programas secuenciales y selectivos, en un poco más del 60% de los casos, pero parecen no aportar beneficios sustanciales en el diseño de programas repetitivos, ya que más del 90% de los estudiantes no las utilizaron para tal fin.

Como ya lo habían mencionado Pérez Pérez y Moreno Gutiérrez (2009), al utilizar alguna técnica de representación de algoritmos, se debe comprender adecuadamente no solo las relaciones entre las sentencias a utilizar, sino además la dependencia entre unas y otras: es necesario definir una jerarquía entre ellas, poniendo en primer lugar las sentencias encargadas del manejo de la entrada de datos, pasando por las sentencias que llevan a cabo el proceso de cálculo o gestión de datos en el algoritmo, y finalmente, las sentencias encargadas de generar la salida de resultados.

A este respecto, Forsythe, Skeenan, Organick y Stenberg (1975) mencionaban: *"La construcción de algoritmos y sus diagramas….. es esencialmente un proceso de solución de problemas. Para enseñar a resolver problemas, tenemos que proceder en forma muy diferente a la que estamos acostumbrados. No basta simplemente presentar el desarrollo directo de un elegante diagrama….. Debemos, por el contrario, mostrar cómo llegó el creador de éste a la solución final. Usted debe ver cómo escogemos el lugar para iniciar un problema. Debe ver también algunos de los falsos comienzos y omisiones, algunos de los algoritmos ineficientes obtenidos en nuestros primeros intentos. Sobre todo, usted debe saber que en la construcción de algoritmos primero intentamos obtener alguna clase de solución del problema (buena o mala), después estudiamos y criticamos nuestra solución y tratamos de encontrar formas de mejorarla. El tratar de obtener el mejor diagrama….. en el primer intento lo puede llevar a usted a confundirse y exasperarse".* Definitivamente, el conocimiento de cómo utilizar las herramientas de diagramación, no implica el conocimiento de cómo programar (Farina, 1983).

A este respecto, el conocimiento y aplicación adecuada de las técnicas de representación de algoritmos, no parece ayudar sustancialmente a resolver el problema que presentan los estudiantes, sobre su ineficiente capacidad para desarrollar algoritmos que reflejen correctamente las soluciones solicitadas. Hay que señalar que las técnicas de representación de algoritmos son sólo parte de la compleja estructura metodológica pendiente por definir y ajustar en futuras investigaciones (Pérez Pérez, Fuentes Gálvez y Moreno Gutiérrez, 2008).

Referencias

Farina, M. V. (1983). *"Diagramas de flujo"*. Editorial Diana. Treceava impresión. México. ISBN: 968 -13-0863-8. pp. 15, 33, 56 y 57.

Forsythe, A.; Skeenan, T.; Organick, E.; Stenberg, W. (1975). "*Lenguajes de diagramas de flujo*". Editorial LIMUSA. Segunda reimpresión. México. pp. 66.

Joyanes Aguilar, L. (1990). *"Problemas de metodología de la programación"*. Editorial Mc Graw-Hill. Primera edición. México. pp. 115 a 116. ISBN 84 -7615-462-3.

Pérez Pérez, I., Fuentes Gálvez, A., Moreno Gutiérrez, S. S. (2008). *"Estudio de la problemática presente en el diseño de algoritmos por computadora".* III Congreso Universitario de Tecnologías de Información y Comunicación 2008. Escuela Superior de Tlahuelilpan. UAEH. México.

Pérez Pérez, I., Moreno Gutiérrez, S. S. (2009). *"Estudio de la problemática relativa al uso de las sentencias secuénciales, selectivas e iterativas, en el diseño de algoritmos"*. IV Congreso Universitario en Tecnologías de Información y Comunicaciones 2009. Área académica de Computación. ICBI-UAEH. México. ISBN: 978-607-482-058-4.

Pérez Pérez, I.; Monzalvo López, C. A. (2010). *"Análisis del desempeño de los estudiantes en el diseño y construcción de algoritmos secuenciales, selectivos e iterativos"*. V Congreso Universitario en Tecnologías de Información y Comunicaciones 2010. Área académica de Computación. ICBI-UAEH. México.

Pérez Pérez, I.; Monzalvo López, C. A. (2010). *"Valoración del nivel de desempeño de los estudiantes en el diseño de algoritmos por computadora"*. V Congreso Universitario en Tecnologías de Información y Comunicaciones 2010. Área académica de Computación. ICBI-UAEH. México.

Scheid, F. (1984). *"Introducción a la ciencia de las computadoras"*. Segunda Edición. Editorial McGraw Hill. México. ISBN: 968-451-399-2. pp. 40 a 42.