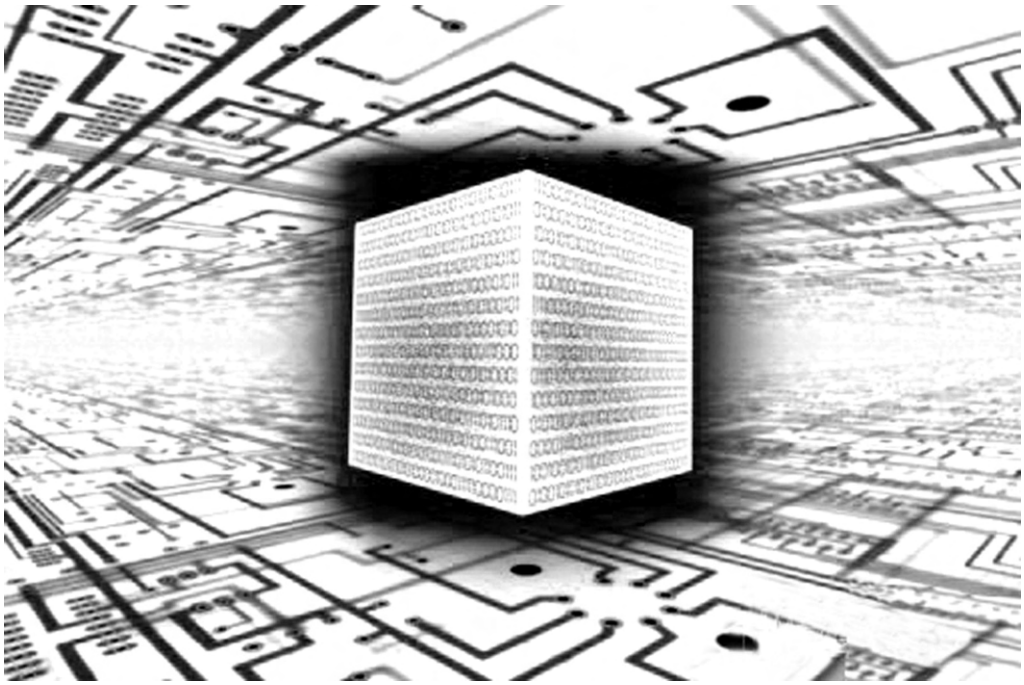
Cómo Inducir al Alumno en el Desarrollo de Habilidades en la Programación

Dora Araceli Cruz Huerta *



Introducción

Hoy día, el no contar con las habilidades necesarias que faciliten el desarrollo de programas en computadora, sigue siendo un problema relevante, ya que existe un alto índice de reprobación en los grupos que cursan asignaturas referentes a la programación, debido entre otros factores, a que los estudiantes no cuentan con una metodología computacional de enseñanza-aprendizaje adecuada, y en otros casos, por carecer de los conocimientos básicos para la solución de problemas lógicos asistidos por computadoras.

La enseñanza educativa en la programación es un área dedicada al servicio social, tomando en cuenta que la forma de enseñar requiere de una gran responsabilidad, dada su importancia, ya que implica una continua capacitación para todos, especialmente para los maestros, pues sin el estudio constante, los conocimientos y el desempeño se tornan obsoletos.

Por ello, es fundamental que los maestros y alumnos comprendan que no existe un método de enseñanza superior a otro, y que el mejor es el que permite lograr

Acerca del autor...

* Maestra en Ciencias, docente de la División de Informática del Tecnológico de Estudios Superiores de Ecatepec.

un aprendizaje significativo y duradero respecto a los objetivos específicos que se aborden en la clase. Mezclar diferentes métodos a través de la planeación didáctica, ayuda a mantener activo el interés de los estudiantes para programar adecuadamente en el mundo real, ya que pueden aplicar sus conocimientos y permitir que el profesor utilice el recurso más adecuado para desarrollar programas por computadora.

Se debe motivar a los estudiantes y dirigirlos hacia tareas que logren el éxito, pero sobre todo, acrecentar el deseo de saber, mediante estrategias de aprendizaje y formación de habilidades de estudio.

El propósito de este artículo, es dar a conocer que la implementación de diferentes estrategias de enseñanza, induce y facilita el aprendizaje del alumno en la programación asistida por computadora.

Desarrollo de habilidades en la programación

Continuamente los seres humanos realizamos una serie de pasos, procedimientos o acciones que nos permiten alcanzar un resultado o resolver un problema. La idea fundamental para aprender a programar, consiste en desarrollar la lógica, que es necesaria para solucionar problemas en forma algorítmica, independientemente del uso de algún lenguaje de programación; esto es, aprender a diseñar algoritmos empleando un pseudolenguaje, y no hacerlo directamente con un lenguaje.

Elaborar un programa por computadora implica seguir un conjunto de pasos secuenciales y cronológicos, que comienzan con la detección y definición del problema, para determinar la implementación del programa que lo soluciona. A continuación se describen en detalle:

1. Definición del problema

Este proceso inicia cuando surge la necesidad de resolver algún problema a través de la computadora. Primero debemos identificarlo y comprender la utilidad de la solución que se obtenga. Es preciso formarse una visión general del mismo, estableciendo las condiciones iniciales (los puntos de partida) y, además, sus límites, es decir, dónde empieza y dónde termina el problema.

Por ejemplo, si requerimos calcular el sueldo de un empleado, la solución señalará la cantidad que se debe pagar, y el beneficio que se logra es que servirá precisamente para remunerar a dicho trabajador. La situación anterior consiste en un pago de sueldos, involucrando para cada empleado atributos como: nombre, tiempo trabajado y sueldo que percibe por unidad de tiempo dedicada a su labor.

2. Análisis del problema

En este punto, es imprescindible entender a detalle el problema en cuestión, para lograr una estructura del mismo en términos de los DATOS disponibles como materia prima, y definir el PROCESO necesario para convertir los datos en la INFORMACIÓN requerida.

Etapas que permiten resolver un problema

Análisis profundo del problema.
Construcción del algoritmo.
Verificación del algoritmo.
Representar la operación o acción que permite el ingreso de los datos del problema.
Representar la operación o conjunto de operaciones secuenciales, cuyo objetivo es obtener la solución al problema.
Representar una operación o conjunto de operaciones, que permitan comunicar al exterior el o los resultados alcanzados.

Se debe ser cuidadoso, pues muchas veces no se toman en cuenta ciertas características que permiten realizar lo antes mencionado en forma adecuada.

Diseño del programa

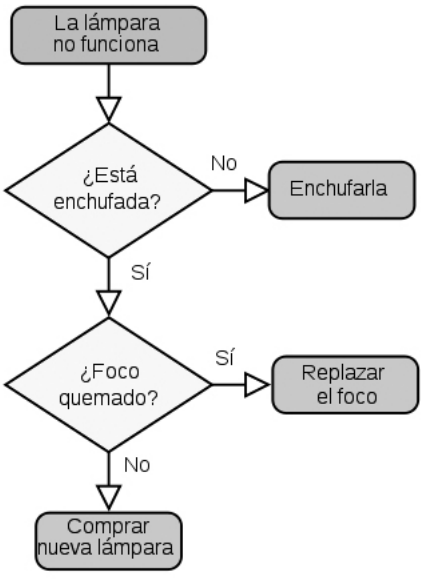
Durante este paso, se diseña la lógica para la solución del problema, con base en lo siguiente:

a. **Elaborar el algoritmo.** Se diseña el algoritmo de la solución del problema, es decir, se estructura la secuencia lógica y cronológica de los pasos que la computadora deberá seguir, utilizando alguna técnica convencional, como el pseudocódigo, los diagramas de flujo, diagramas de clase, los diagramas Warnier, entre otros.

b. **Prueba de escritorio.** Se simula el funcionamiento del algoritmo con datos propios respecto al problema, y se comprueban a mano los resultados, a fin de validar la correcta operación del algoritmo. Si quedamos satisfechos con los resultados de la prueba, habremos agotado este punto, pero en caso contrario, se deberá modificar el algoritmo y posteriormente volver a probarlo hasta que esté correcto.

Es posible que se deba retroceder a cualquier paso precedente.

Hasta aquí, se tiene ya diseñada la solución del problema, y estamos listos para pasar al siguiente punto: trascribirlo a un lenguaje de computadora.

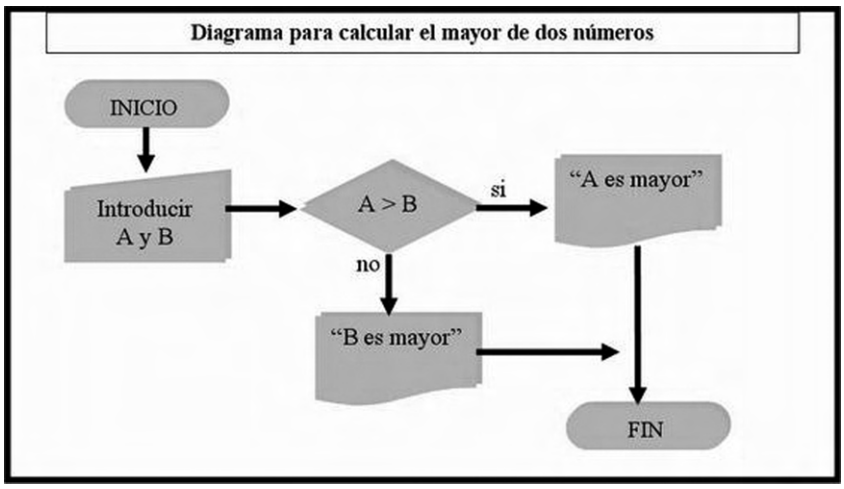


Elementos necesarios para documentar la solución de un problema

Algoritmo

Un algoritmo se define como un conjunto de pasos, procedimientos o acciones que nos permiten alcanzar un resultado o resolver un problema.

A continuación, se mencionarán las características que debe reunir un algoritmo en la primera fase de documentación:



Precisión: Los pasos deben de ser plasmados claramente.

Determinismo: Dado un conjunto de datos idénticos de entrada, siempre debe arrojar los mismos resultados.

Finitud: Independientemente de la complejidad del mismo, siempre debe tener una longitud finita.

Módulos o secciones de un algoritmo

Datos de entrada.
Procesamiento de datos.
Impresión de resultados.

Diagrama de flujo

Un diagrama de flujo representa la esquematización gráfica de un algoritmo para alcanzar la solución de un problema. Su correcta construcción es sumamente importante, porque éste nos facilitará más tarde la escritura del programa en algún lenguaje de programación. Si el diagrama de flujo está completo y en orden, el paso del mismo a un lenguaje de programación, será relativamente simple y directo.

Los símbolos se deberán colocar adecuadamente, permitiendo crear una estructura gráfica flexible, que ilustra los pasos o procesos a seguir para alcanzar un resultado específico.

Reglas para la construcción de diagramas de flujo

Se inicia de izquierda a derecha y de arriba hacia abajo (top-down).
Todo diagrama debe poseer un inicio y un fin.
Las líneas utilizadas para indicar la dirección del flujo del diagrama, deben ser rectas, verticales y horizontales.
Todas las líneas utilizadas para indicar la dirección del flujo del diagrama, deben estar conectadas. La conexión puede ser a un símbolo que exprese lectura, proceso, decisión, impresión, conexión o fin de diagrama.
La notación utilizada en el diagrama de flujo debe ser independiente del lenguaje de programación. La solución presentada en el diagrama puede escribirse posteriormente y con facilidad en diferentes lenguajes de programación.
Cuando se realiza una tarea compleja, es conveniente insertar comentarios que expresen o ayuden a entender lo que se hace.
Si el diagrama de flujo requiere más de una hoja para su construcción, debe utilizar conectores adecuados y enumerar las páginas convenientemente.
No puede llegar más de una línea a un símbolo.

Pseudocódigo

Un pseudocódigo es una serie de normas léxicas y gramaticales parecidas a la mayoría de los lenguajes de programación, pero sin llegar a la rigidez de sintaxis de éstos ni a la fluidez del lenguaje coloquial. Esto permite codificar un programa con mayor agilidad que en cualquier lenguaje de programación, con la misma validez semántica; normalmente se utiliza en las fases de análisis o diseño de software, o en el estudio de un algoritmo. Forma parte de las distintas herramientas de la ingeniería de software.

El pseudocódigo describe un algoritmo utilizando una mezcla de frases en lenguaje común, instrucciones de programación y palabras clave que definen las estructuras básicas. Su objetivo es permitir que el programador se centre en los aspectos lógicos de la solución a un problema.

Ventajas de utilizar un pseudocódigo a un diagrama de flujo

1. Permite representar de forma fácil operaciones repetitivas o complejas.
2. Es más sencilla la tarea de pasar de pseudocódigo a un lenguaje de programación formal.
3. Si se siguen las reglas de identación, se pueden observar claramente los niveles en la estructura del programa.
4. En los procesos de aprendizaje, los alumnos de programación estarán más cerca del paso siguiente (codificación en un lenguaje determinado, que los que se inician en la modalidad Diagramas de Flujo).
5. Mejora la claridad de la solución de un problema.

Cualidades de un lenguaje de programación

A continuación se destacan algunas cualidades de los lenguajes de programación:

Simplicidad y claridad. ¿Es fácil para un programador escribir un programa en este lenguaje? ¿Hasta qué punto es inteligible este programa para lector medio? ¿Es fácil aprender y enseñar el lenguaje? Algunos lenguajes se diseñaron intencionalmente para facilitar la claridad de expresión, y otros para simplificar el aprendizaje y la enseñanza de los principios de una programación estructurada.

Uniones. Un elemento de un lenguaje se une a una propiedad en el momento en que se define dicha propiedad para él. Los principales momentos de unión de los elementos con sus propiedades son los siguientes:

1. **Definición del lenguaje.** Cuando definimos un lenguaje, los tipos de datos básicos se unen a palabras reservadas que los representan.
2. **Implementación del lenguaje.** Escribimos el compilador o el intérprete de un lenguaje, los valores se unen a las representaciones del equipo.
3. **Escritura del programa.** Cuando escribimos programas en determinados lenguajes, los nombres de variables se unen con tipos que permanecen asociados con dichos nombres a lo largo de la ejecución del programa.
4. **Compilación-carga del programa.** Cuando compilamos —cargamos los programas—, las variables "estáticas" se asignan a direcciones de memoria fijas, la pila de tiempo de ejecución se asocia con un bloque de memoria y se asigna el mismo código máquina a otro bloque de memoria.
5. **Ejecución del programa:** Cuando estamos ejecutando un programa las variables se unen con valores, como en el caso de la asignación x=3.

Hay un grupo de precauciones prácticas relacionadas que rodean el estudio de los lenguajes de programación. Las consideraciones siguientes proporcionan un conjunto de restricciones que afectan a todos los diseños de lenguaje de programación.

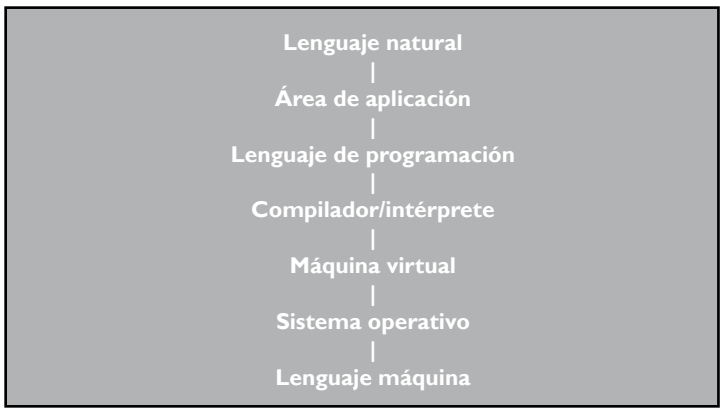
Restricciones de arquitectura. Los lenguajes de programación están diseñados para las computadoras.

Durante algunos años, se consideró a la arquitectura de computadoras como un subproducto, y no como un precursor del diseño del lenguaje.

Cuando se toman en cuenta las virtudes de las distintas opciones de diseños de lenguajes, siempre se limitan por la necesidad de implementar dichos lenguajes de forma eficiente y efectiva dentro de las restricciones impuestas por las variantes actuales.

La idea de un buen diseño de lenguaje puede conducir a una arquitectura de computadoras viable, radicalmente nueva y comercial, aunque probablemente no sea una buena opción.

Restricciones contextuales. Los lenguajes no están diseñados sólo para cuadrar dentro de las restricciones de una familia de arquitecturas determinada, también deben estarlo de modo que satisfagan otras restricciones impuestas por el contexto en el que se van a utilizar: el área de aplicación, el sistema operativo, la red y las preferencias de la comunidad de programación.



Dichos programadores trabajan en profesiones distintas que utilizan sus propios estilos de diseño de software, sus propias herramientas y sus propios lenguajes naturales, y para comunicarse entre ellos; este esquema superior de la configuración compleja del diseño de un lenguaje, está resumido de la siguiente forma:

Codificación de un programa

En este paso, se procede a codificar el programa en el lenguaje de programación que vaya a utilizar. Este proceso es sumamente sencillo, dado

que ya se tiene diseñado el programa, sólo se concreta uno a convertir las acciones del algoritmo en instrucciones de computadora. El programa codificado debe editarse, compilarse, probarse y depurarse; es decir, se ejecuta para verificar su buen funcionamiento y se hacen las correcciones o los ajustes pertinentes. Si aparecen errores difíciles de corregir en este paso, quiere decir que debemos retroceder.

Para que el programa se encuentre en condiciones de ser entendido y ejecutado por la computadora, debe estar en el leguaje máquina o código objeto, traducido por un compilador, a código accesible para la máquina mediante la compilación. El proceso de compilación es el siguiente: una vez que tenemos codificado el programa, debe ser introducido mediante el proceso de edición, para lo cual se utiliza un editor que permite crear un archivo en el cual se introduce el programa, creándose el programa fuente con las instrucciones que se elaboran en el lenguaje que se esté utilizando en este momento. El programa fuente es sometido al proceso de compilación, que mediante un compilador (traductor del lenguaje), se traduce instrucción por instrucción a código objeto, creándose un archivo con el programa objeto, el cual es entendible directamente por la máquina. Si en el proceso de traducción se encuentra algún error, se suspende el proceso; el programador debe corregir el error en el programa fuente y luego someterlo de nuevo al procesos de compilación.

Una vez que el proceso de compilación ha terminado con éxito, se tiene el programa objeto, el cual puede ser ejecutado por la computadora, misma que seguirá las instrucciones paso a paso llevando a cabo las acciones que se indican emitiendo los resultados correspondientes, si éstos no son satisfactorios o existen errores, el proceso de programación debe ser repetido desde alguno de los pasos precedentes.

Implantación del programa

Una vez que el programa está correcto, se instala y se pone a funcionar, entrando en operación normalmente dentro de la situación específica para la que se desarrolló. Debe ser supervisado continuamente para detectar posibles cambios o ajustes que sea necesario realizar.

Es importante contar con una metodología que permita conducir la enseñanza-aprendizaje de la programación, mediante el uso de un pseudolenguaje de diseño de programas o algoritmos orientados a objetos, los cuales guiaran a los alumnos que están iniciando sus estudios de programación.

Puntos a considerar

Existen diversos factores en donde las respuestas que aportaron tanto los maestros como los alumnos, respecto a su enseñanza-aprendizaje, son subjetivas.

Tomando en cuenta principalmente el respeto que deben tener, tanto los maestros como alumnos **(ética y valores)**, para concebir la enseñanza-aprendizaje.

A continuación se cita lo que puede considerar el profesor al iniciar una clase de programación:

1. Explicarle al alumno la estructura de la ciencia (**informática**). Platicar acerca de la concepción de la informática y cuáles son sus campos de estudio.

2. Dar a conocer el impacto de la ciencia en la sociedad, en la industria, la familia y en lo personal.

3. Despertar en los alumnos el interés para estar inmersos en ella.

4. Cómo aprender a crear programas que modifiquen y mejoren lo anteriormente expresado.

5. Hacer un examen diagnóstico de lógica para determinar sus capacidades, inteligencia, razonamiento y lógica.

6. Usar una metodología que se pueda aplicar en base a los resultados del punto anterior. Ejemplo: concepto, antecedente y objetivo, es decir, explicación del aspecto teórico-práctico, aplicado a la solución de problemas reales.

7. Diversidad del uso de herramientas como: tareas, programas a desarrollar, gráficas y pruebas de escritorio, entre otras.

8. Uso de técnicas que faciliten el aprender a programar a través de un orden de conceptos y elementos, desde lo más básico, hasta lo más complicado y abstracto.

9. Los temas deben ser explicados con detalle y análisis de cada algoritmo y programa.

10. Dar confianza y disponibilidad a los alumnos para que ellos se acerquen y poderles explicar todos los detalles, para que puedan constatar y probar los programas en la computadora.

11. No olvidar que se debe complementar la teoría con la práctica.

12. Es importante realizar una retroalimentación para ayudar a los alumnos, motivándolos a ayudar a sus compañeros y así poder reforzar lo aprendido.

13. Es primordial fomentar el aprendizaje independiente del alumno y su propia responsabilidad en la programación, sugiriéndoles ser autodidactas y que de acuerdo al propio interés le dediquen el tiempo que consideren adecuado.

A continuación se cita lo que puede considerar el alumno al iniciar una clase de programación:

1. Estar disponible física y mentalmente (**mente sana en cuerpo sano**).
2. Tener su propio criterio para no dejarse influenciar acerca de la opinión que se deriva del maestro y esperar a comprobarlo por sí mismo.
3. No mezclar los problemas personales con las clases.
4. Mantener un ambiente de respeto entre profesor y alumno.
5. Analizar y comprender el objetivo y contenido temático de la materia.
6. Ser organizado en el tiempo dedicado a entender la teoría para sustentarla con la práctica.
7. Disponibilidad para aprender, estar atento, ser participativo, tener interés en obtener generalizaciones.
8. Analizar y documentarse previamente acerca de los temas a tratar en clase.
9. Aprovechar los recursos que la institución ofrece, para fomentar el aprendizaje.

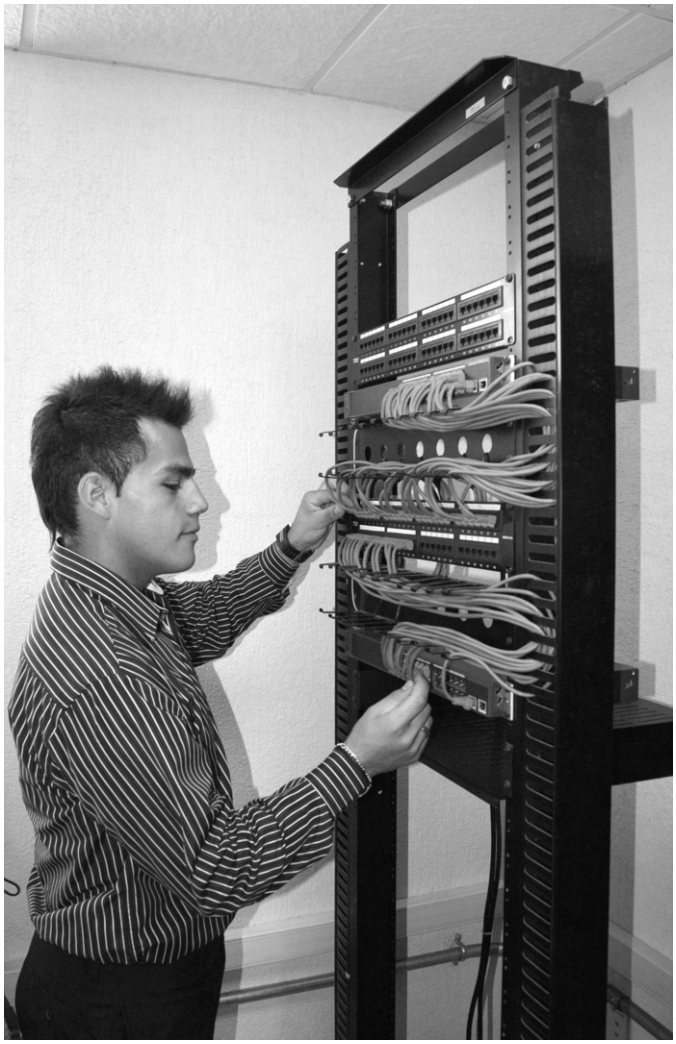
Conclusiones

Con base a lo antes mencionado, se concluye que una teoría educativa es aquella que sirve como una guía educativa, pero sobre todo como un catalizador para respetar el interés de los alumnos y motivarlos a reflexionar, discutir y profundizar sobre el aprendizaje.

El objetivo es motivar y proporcionar estrategias, técnicas y métodos, con el fin de inducir a los alumnos a generar sus propias habilidades en la programación, mediante el uso de los lenguajes actuales de programación de computadoras, para solucionar problemas informáticos.

Recomendaciones

Es importante reflexionar acerca de la relevancia que tiene crear nuevas estrategias de aprendizaje, como herramientas de apoyo para el proceso de enseñanza aprendizaje, ya que los resultados obtenidos permitirán observar desde la perspectiva de los docentes, el rendimiento escolar del alumno y que éste debe ser actualizado e implementando como recurso didáctico a través del uso de la tecnologías educativas.



Desde el punto de vista del alumno, se considera que el uso de estrategias y tecnologías educativas en el proceso de aprendizaje, ayudará a mejorar su rendimiento y desenvolvimiento escolar.

El contar con un material didáctico interactivo, facilitará el entendimiento de los temas expuestos por el docente en clase.

Por lo tanto, es importante el uso de materiales y estrategias didácticas interactivas, que permitan facilitar las evaluaciones por parte del docente, generando en el alumno interés por el aprendizaje de la materia de programación del área de computación.

También es importante que el docente se capacite periódicamente acerca del uso de tecnologías educativas como herramientas de apoyo para el aprendizaje y que conjuntamente con los alumnos puedan elaborar dichos materiales didácticos interactivos, ya que de esta forma se podrá lograr un mejor aprendizaje y rendimiento de los alumnos, a fin de que puedan desenvolverse adecuadamente en el ámbito laboral, recordando que “la práctica hace al maestro”.

Bibliografía

Amador, Rocío. *Comunicación Educativa Nuevas Teologías*, México, CISE UNAM, 1994.

Bernard, J. Poole (1999), *Tecnología Educativa*, España, Mc. Graw Hill.

Cairo Battisstutti, Osvaldo, *Metodología de la Programación, Algoritmos, Diagramas de Flujo y Programas*, Tercera Edición, México, Ed. Alfaomega.

López Román, Leobardo, *Metodología de la Programación Orientada a Objetos*, México, Ed. Alfaomega.

Tucker, Allen y Noonan, Robert, *Lenguajes de Programación Principios y Paradigmas,* Ed. Mc Graw Hill.