UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

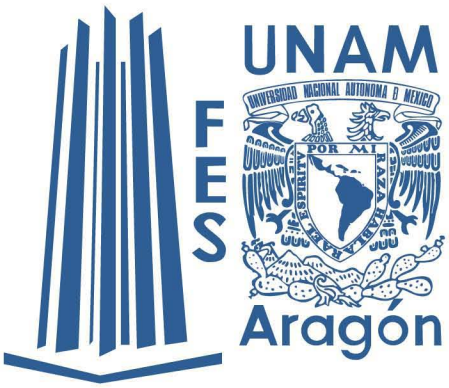
Diseño de Metodología para la Impartición de los Principios de Programación para la Carrera de Ingeniería en Computación.

TESIS

PARA OBTENER EL TÍTULO DE:
Ingeniero en Computación

PRESENTAN:
Medina Ramírez Nora
Quiroz Ruiz Mario

ASESOR:
MTRO. Gastaldi Pérez Juan



Ciudad Nezahualcóyotl, San Juan de Aragón, Estado de México
Mayo 2018

AGRADECIMIENTOS

A mi madre, M. Magdalena Ruiz Ramírez:

Nadie nos enseña a ser padres, uno va aprendiendo en el camino, no es fácil, y si hay algo que te debo agradecer es que nunca te rendiste. Gracias por apoyarme. Aprendí a ser fuerte e independiente gracias a ti. Gracias.

A mi amiga, Nora Medina Ramírez:

Te agradezco haber recorrido este camino conmigo, sé que no fue sencillo y que de haber sabido lo complicado que sería lo hubieras pensado mejor. Tus ideas, tus conocimientos, tu experiencia en el área docente, las correcciones que me hacías, todo lo que aportaste al proyecto hicieron de éste un trabajo mejor. Yo solo no hubiera podido concebir lo que tú planteaste. Gracias por todo, gracias por seguir, gracias por apoyarme. Gracias.

A la Mtra. A. Y. Aguilar Muñoz:

Te agradezco mucho, porque me enseñaste a ver las cosas de una manera diferente. Los caminos que he recorrido y que recorreré son gracias a ti. Al final, tú fuiste mi maestra. Tu aportación a este trabajo fue transcendental. Te agradezco de corazón.

A mi querida prima, Ernestina Quiroz Juárez:

Gracias por tu tiempo, tu esfuerzo, tus consejos y en especial tus palabras. Nos impresionó lo que hiciste en este trabajo. Gracias.

Gratias vobis valde

Mario Quiroz Ruiz

AGRADECIMIENTOS

Me gustaría empezar agradeciendo a la Facultad de Estudios Superiores Aragón, lugar donde no sólo tuve la oportunidad de crecer académicamente sino también como persona.

Especial agradecimiento al Mtro. Juan Gastaldi Pérez, asesor de esta tesis, por la orientación que nos brindo durante la elaboración de la misma. De igual modo a los asesores de tesis, por todas las sugerencias e interés mostrado en este trabajo.

Mi más sincero agradecimiento a mi compañero de tesis, Mario, quien sin su dedicación, entusiasmo, conocimientos, y todos esos momentos de risa, este trabajo no hubiera sido posible.

A mis padres, Carmen y Marcos, por su apoyo incondicional.

A mi hermano, Marco, por el animo y comprensión que me dio durante este proceso.

A todos los que han ayudado a terminar este trabajo brindándome su amistad y apoyo moral.

Finalmente a la persona más importante, mi esposo, Yan. Gracias por todo el amor, apoyo y criticas que me ha brindado durante todos estos años, ha sido un largo camino que sin su ayuda no hubiera podido recorrer.

Nora Medina Ramírez

ÍNDICE

INTRODUCCIÓN

La presente tesis tiene como objetivo principal diseñar una metodología que facilite enseñar los principios de programación a los alumnos de la carrera de Ingeniería en Computación, impartida en la Facultad de Estudios Superiores (FES) Aragón de la UNAM.

Se decidió realizar este trabajo de tesis, porque para algunos profesores resulta difícil enseñar temas de programación, pero no por la falta de conocimientos en el tema, sino por la falta de un método pedagógico adecuado, lo que genera problemas en el proceso de enseñanza-aprendizaje. En consecuencia, los alumnos y egresados tienen un nivel deficiente de conocimientos, situación indeseable puesto que en el perfil académico de un egresado de Ingeniería en Computación una parte de sus conocimientos se enfoca en la habilidad de diseñar sistemas informáticos.

En el campo laboral hay un déficit de profesionistas que se dedican al desarrollo de software, lo que refleja que la problemática del proceso de enseñanza-aprendizaje, con respecto a temas de programación, no es exclusiva de la FES Aragón, sino que está presente, de manera generalizada, en los diferentes centros educativos de nivel superior. Sin embargo, el alcance de este trabajo se reduce a la FES Aragón, pues al ser egresados de esta institución, conocemos la situación que se vive en la carrera.

Elegimos este tema porque consideramos que tenemos los conocimientos, la experiencia y el interés necesarios para proponer una posible solución a la problemática planteada. En el caso de la coautora Nora Medina Ramírez, ella cuenta con conocimientos en temas pedagógicos y tiene experiencia dando clases, además de conocer diferentes lenguajes de programación. Con respecto al coautor Mario Quiroz Ruiz, profesionalmente se ha dedicado al desarrollo de sistemas informáticos, además de haber impartido diferentes cursos de programación, tanto en el Centro de Actividades Extracurriculares

(CAE) de la FES Aragón, como en la Dirección General de Servicios de Cómputo Académico (DGSCA), hoy DGTIC, de la UNAM.

Para este trabajo se realizaron dos tipos de investigación: de campo y documental.

Para la investigación de campo, se recurrió a la observación directa de la problemática: el proceso de enseñanza-aprendizaje de la programación, al cual se enfrentan los alumnos y profesores. Los instrumentos de investigación fueron encuestas dirigidas a los alumnos y entrevistas para los profesores.

Para la investigación documental se recurrió a material bibliográfico, lo que permitió definir los conceptos más significativos de la enseñanza y el aprendizaje, así como de las teorías pedagógicas.

El trabajo se dividió en tres capítulos.

En el capítulo I, "Problemática de la enseñanza-aprendizaje de la programación en Ingeniería en Computación", se analiza la problemática en el proceso de enseñanza-aprendizaje de la programación, principalmente desde el punto de vista de los alumnos, mediante la aplicación de encuestas; en tanto, a los profesores se les entrevistó con el fin de conocer su opinión sobre esta problemática.

En el capítulo II, "Marco teórico", se definen los conceptos básicos de enseñanza y aprendizaje, así como los de las diferentes corrientes pedagógicas, poniendo especial énfasis en el constructivismo.

En el capítulo III, "Propuesta metodológica", se desarrolla una propuesta teórica enfocada en la implementación de una metodología para la enseñanza de los conceptos básicos de programación, en la carrera de Ingeniería en Computación de la FES Aragón.

CAPÍTULO I: PROBLEMÁTICA DE LA ENSEÑANZA-APRENDIZAJE DE LA PROGRAMACIÓN EN INGENIERÍA EN COMPUTACIÓN

Una de las funciones de las instituciones de educación superior, es la formación de profesionistas con el fin de generar gente capacitada que satisfagan las diversas necesidades que la sociedad requiere.

En las últimas décadas la computación ha crecido de forma exponencial dado que su campo de aplicación abarca muchas de las actividades del ser humano, desde el entretenimiento hasta la investigación, es tal la incursión de las Tecnologías de la Información (TI) que actualmente a nuestra sociedad se le considera como la Sociedad de la Información.

Los sistemas informáticos ayudan al desarrollo y al progreso del lugar donde se implementan, sin embargo, hay un gran déficit de profesionistas en esta rama, no sólo en México, sino en países de primer mundo, como los son los Estados Unidos de América, Alemania y España. Esta situación nos la confirma Ignacio Sainz de Baranda, presidente de la International Association of Microsft Chanel (IAMCP), el cual comentó con referente a la situación de España “Hay 700.000 puestos no cubiertos para este año (2016) y 2 millones para 2020 y lo mismo pasa en Estados Unidos e incluso en Latinoamérica” Hernandis, 2016.

Además de lo mencionado, con base en información de Daysoft empresa de reclutamiento en TI, el puesto de programador en México presenta las mayores carencias de talento.

Maribel Espinosa, directora de Reclutamiento y Selección en Daysoft en el año de 2011 aseguró que “la dificultad con la que nos encontramos al momento de reclutar gente del sector de TI es que no hay mucha gente joven que se quiera dedicar a la programación, y además carece de especialización” Chávez, 2011.

Antes de abordar la problemática debemos entender que es programar, sus características y fases.

Programar es el proceso de desarrollar programas informáticos a través de un lenguaje artificial que obedece a uno o más paradigmas, dicho programa en su forma más simple es un conjunto de sentencias o instrucciones y en una forma más abstracta es una representación de nuestro mundo ya sea físico o conceptual; el cual es ejecutado por una computadora o componente electrónico. Dicho proceso implica las fases de análisis, diseño, codificación y pruebas. Fuente: Elaboración propia (2017)

De la definición anterior se desprenden otros conceptos que debemos de explicar para crear un panorama más completo de lo que debe aprender el alumno.

Lenguaje de Programación. Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones, el cual es utilizado para crear programas informáticos.

Paradigma. Los lenguajes de programación están basados en uno o más paradigmas, entendiendo por paradigma como la manera de formular la resolución de un problema, es decir, a un mismo problema lo podemos resolver de diferentes maneras.

Un ejemplo de esto es la Programación Estructurada la cual establece mediante el teorema del mismo nombre que "*para desarrollar un programa, tan solo necesitamos 3 estructuras básicas: estructuras de bifurcación, estructuras de repetición y proceso*". Imagen 1-1

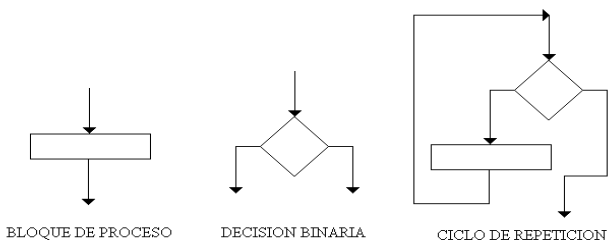


Imagen 1–1. Representación gráfica de las estructuras de control. Fuente: Elaboración propia (2017).

Otro ejemplo es la Programación Orientada a Objetos (POO), la cual se enfoca en crear entidades abstractas llamadas objetos los cuales tienen estado y comportamiento, y posteriormente se hacen interactuar entre sí para resolver el problema. Imagen 1-2.

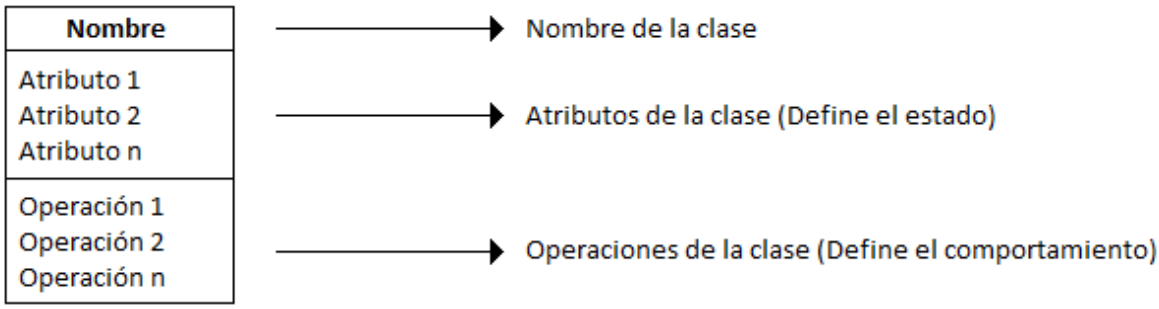


Imagen 1–2 . Representación gráfica de un objeto. Fuente: Elaboración propia (2017).

Análisis. En esta fase se debe comprender el problema, así como las entradas que se requieren y las salidas que se esperan.

Diseño. En esta fase se tiene que proponer un algoritmo que resuelva el problema. Este puede ser por ejemplo un diagrama de flujo o un pseudocódigo.

Codificación. En esta fase se convierte el algoritmo al código de un lenguaje de programación.

Pruebas. En esta fase se comprueba si el programa resuelve o no el problema planteado de forma satisfactoria.

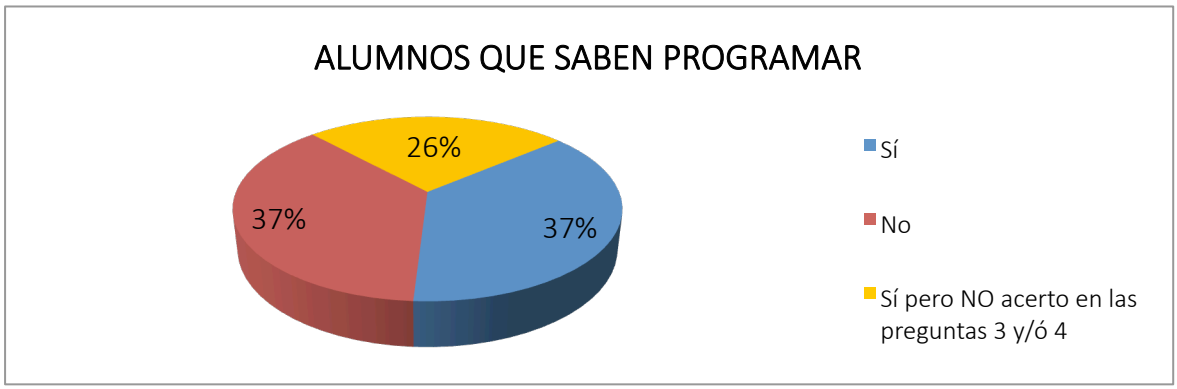
1.1 Problemática de la Enseñanza-Aprendizaje

Sabemos que existe una problemática referente al aprendizaje de las materias de programación, pero no sabíamos ni la magnitud ni las características de este. En consecuencia, elaboramos una encuesta (Anexo 1) que se aplicó a alumnos de tercer semestre, dado que para este momento ya habían cursado las materias de “Computadoras y Programación” y “Programación Orientada a Objetos”.

La encuesta se aplicó a 89 alumnos pertenecientes a la generación del 2008, que representan la primera generación del nuevo plan de estudios (Plan: 1279). A continuación, analizaremos los resultados obtenidos de las encuestas.

Se les preguntó si saben programar, no obstante, teníamos la inquietud de que algunos alumnos contestaran "Sí" ya sea por vergüenza o simplemente porque creen saber. Para "comprobar" que supieran programar, los que contestaban "Sí" tenían que dar definiciones de tres conceptos básicos de programación. Cabe señalar que aún con este filtro no podemos asegurar que los alumnos que contestaron correctamente las preguntas 3 y 4 de la encuesta verdaderamente saben programar.

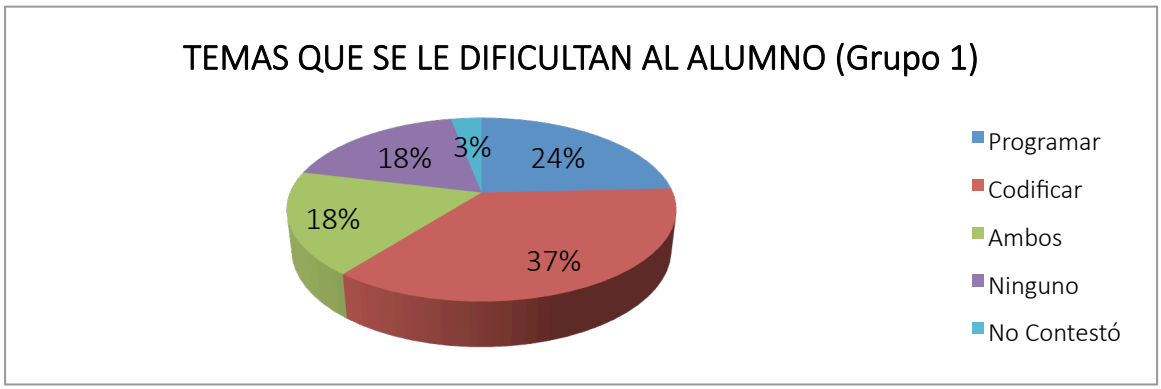
En la **Gráfica 1-1**, podemos observar que el 37% contestó que No, el 37% que Sí contestando correctamente las preguntas 3 y/o 4, y el 26% aunque contestó Sí, NO contestaron correctamente las preguntas 3 y/o 4, por lo que podríamos afirmar que el 63% de los encuestados no sabe programar o tiene deficiencias.



Gráfica 1-1. Alumnos que no saben programar. Fuente: Elaboración propia (2008).

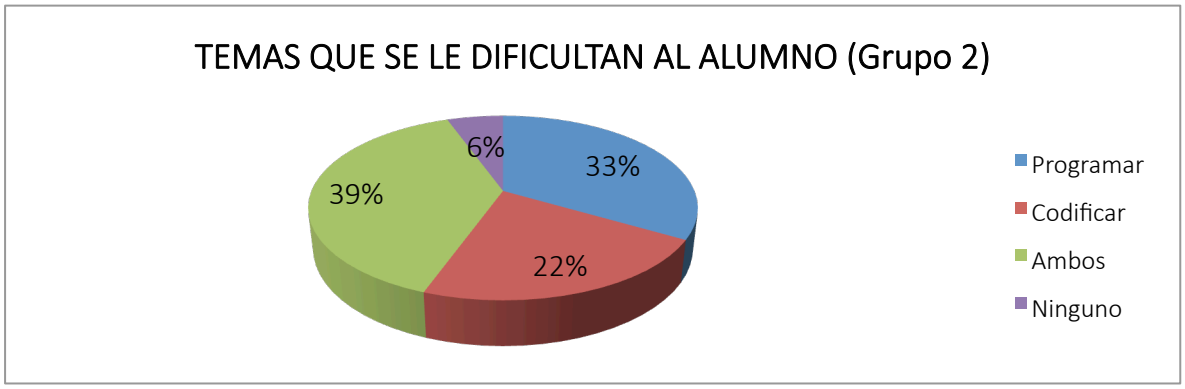
Con base en los resultados obtenidos y para un mejor procesamiento de la información, hemos dividido las encuestas en tres grupos. El **Grupo 1** representa a los que saben programar (37%), el **Grupo 2** representa a los que no y a los que no demostraron saber (63%) y el **Grupo Global** representa al total de los encuestados.

Las siguientes tres gráficas muestran los temas que se les dificulta más a los alumnos. En la **Gráfica 1-2** podemos observar que sólo el 18% no tiene dificultades, lo cual llama la atención dado que es un porcentaje muy bajo tomando en cuenta que la gráfica representa al conjunto de alumnos que saben programar.



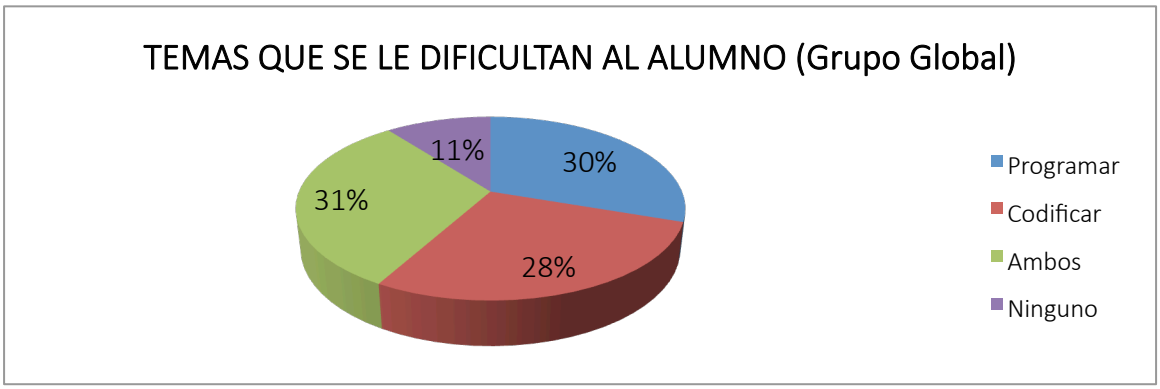
Gráfica 1-2. Temas que se le dificultan al alumno (Grupo 1). Fuente: Elaboración propia (2008).

La **Gráfica 1-3** nos muestra que el mayor porcentaje lo tiene la opción "Ambos", es decir, la mayoría de los alumnos tienen problemas al programar y codificar. Cabe mencionar que el 6% que contesto "Ninguno" está asociado a los alumnos que contestaron que sí saben programar pero que no lo demostraron.



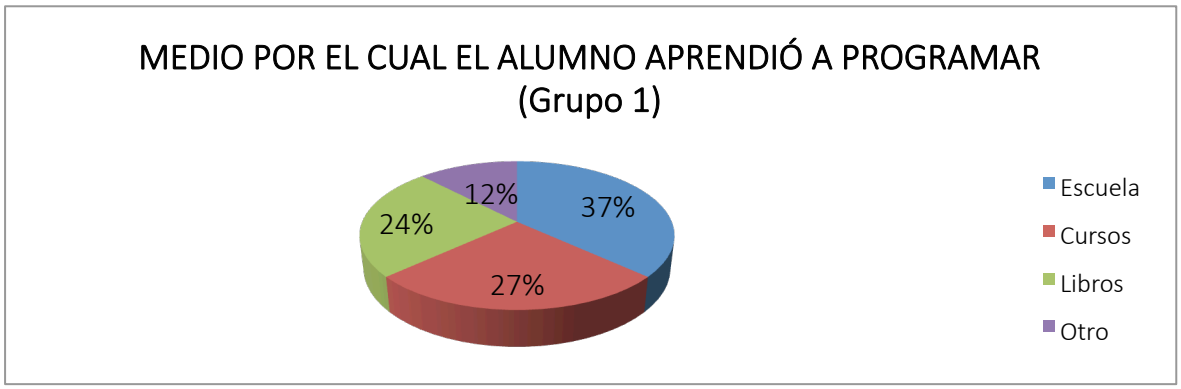
Gráfica 1-3. Temas que se le dificultan al alumno. Fuente: Elaboración propia (2008).

La siguiente gráfica (**Gráfica 1-4**) pertenece al **Grupo Global** y como podemos observar tan solo el 11% de los encuestados contestó que no tenía ningún problema. Este dato es muy interesante ya que nos puede acercar aun más al porcentaje de alumnos que saben programar.



Gráfica 1-4. Temas que se le dificultan al alumno (Grupo Global). Fuente: Elaboración propia (2008).

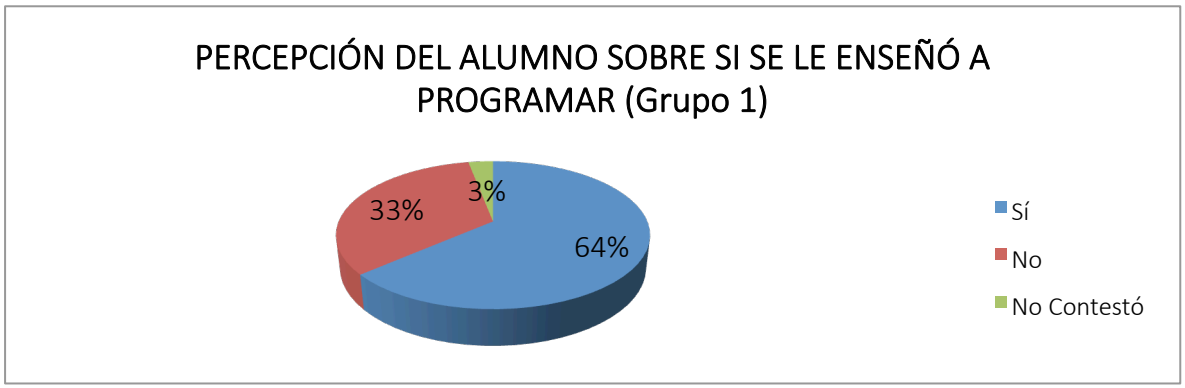
En la **Gráfica 1-5** podemos observar por cuál medio los alumnos del **Grupo 1** aprendieron. Si sumamos todos los porcentajes de las opciones diferentes a escuela, podemos observar que 63% de los alumnos que aprendieron a programar lo hicieron por un medio distinto.



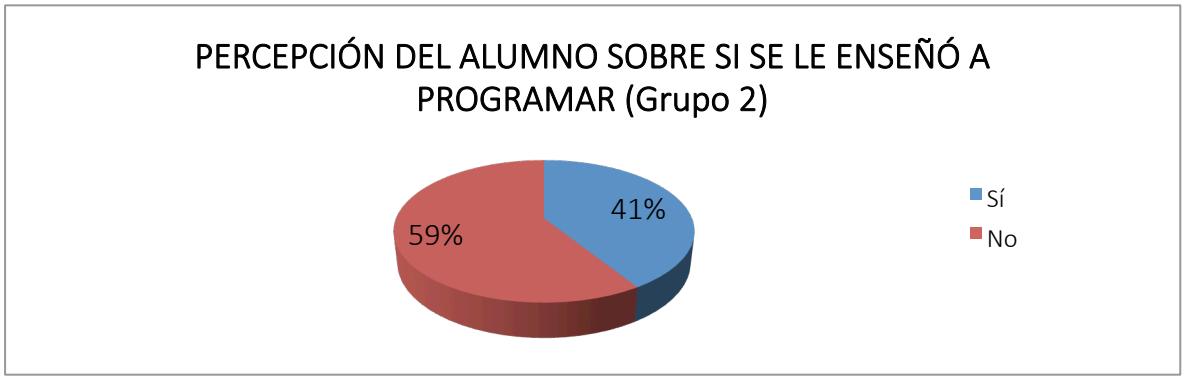
Gráfica 1-5. Medio por el cual el alumno aprendió a programar (Grupo 1). Fuente: Elaboración propia (2008).

Se les preguntó a los alumnos si consideraban que su profesor les había enseñado a programar, estamos conscientes que la respuesta es subjetiva pero consideramos que es importante conocer la opinión del alumno, estas respuestas se ven reflejadas en las siguientes tres gráficas (**Gráfica 1-6**, **Gráfica 1-7** y **Gráfica 1-8**)

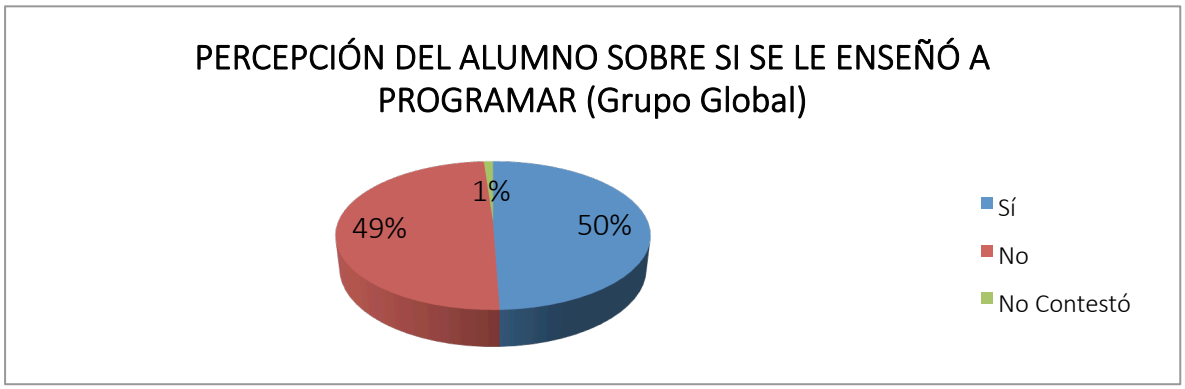
Es significativo mencionar que en la **Gráfica 1-6** representando al **Grupo 1** podemos observar que la mayoría considera que su profesor les enseñó a programar, mientras que en la **Gráfica 1-7** representando al **Grupo 2** ocurre lo contrario, mientras que en la **Gráfica 1-8** representando al **Grupo Global** estos porcentajes que equilibran.



Gráfica 1–6. Percepción del alumno sobre si se le enseño a programar (Grupo 1). Fuente: Elaboración propia (2017).



Gráfica 1–7. Percepción del alumno sobre si se le enseño a programar (Grupo 2). Fuente: Elaboración propia (2008).



Gráfica 1–8. Percepción del alumno sobre si se le enseño a programar (Grupo Global). Fuente: Elaboración propia (2008).

1.2 Ausencia de Análisis

Uno de los mayores problemas en los alumnos es el hecho de que no tienen el hábito de analizar el problema, pasando inmediatamente a la codificación, situación que entorpece el desarrollo de software. El análisis es fundamental dado que permite entender el problema, y el alumno al no comprenderlo en su totalidad, no diseñará una solución adecuada.

Si el alumno no realiza el análisis correspondiente, probablemente se enfrentará con las siguientes preguntas:

¿Qué escribo?

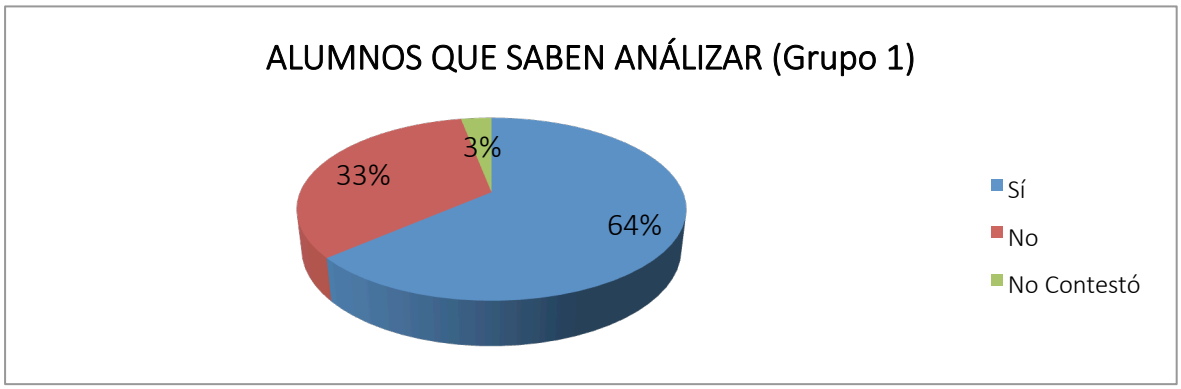
¿Por dónde empiezo?

Preguntas que reflejan que el alumno no está adquiriendo el conocimiento necesario para enfrentarse a los problemas de la materia, además de desarrollar un sentimiento de frustración.

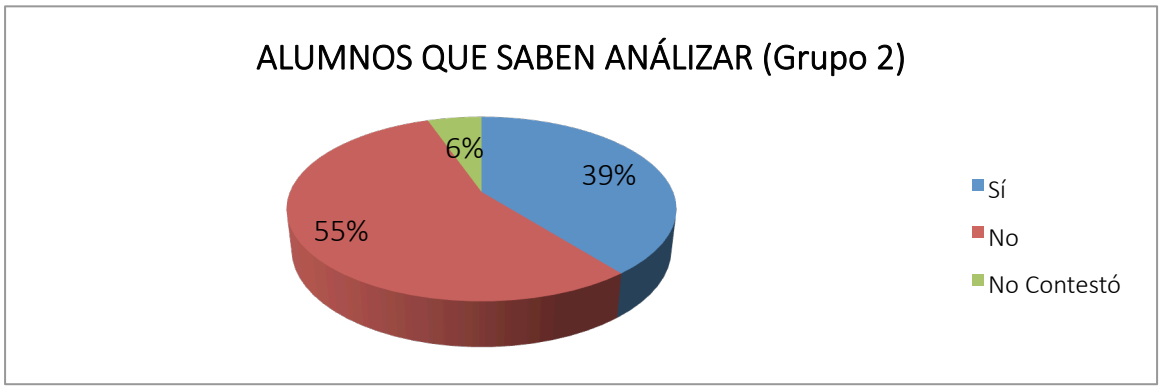
La falta de análisis no obedece a una falta de interés o falta de capacidad intelectual del alumno, sino más bien al desconocimiento de cómo hacerlo. Ahora bien, ¿qué implica hacer un análisis de un problema que deberá de ser codificado? A continuación, mencionamos los puntos que un alumno debería de tomar en cuenta cuando realice el análisis de un problema:

- Entendimiento del problema.
- Identificación de los datos de entrada.
- Tener claro los datos de salida que se esperan.
- Entender cómo fluye la información.
- Entender los diferentes flujos de información.
- Entender cómo interactúan los datos entre sí.

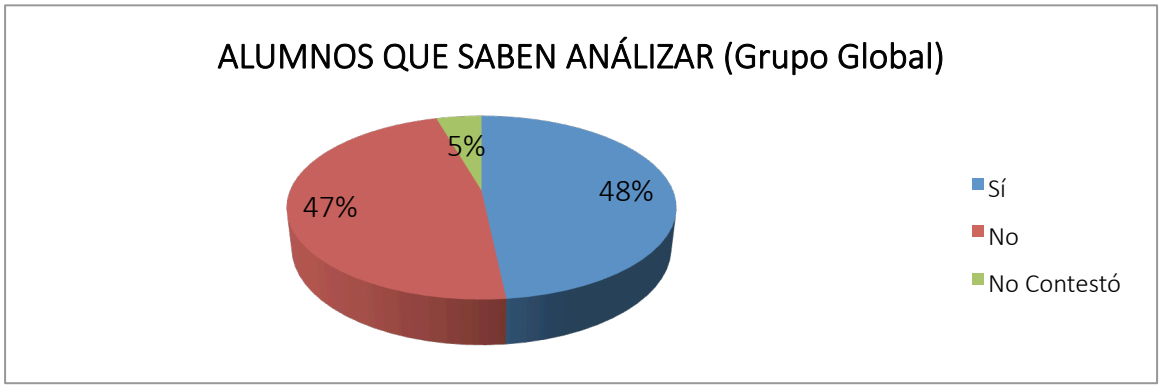
Les preguntamos a los alumnos si sabían analizar un problema, mostrándonos de igual manera que el **Grupo 1** contestó en su mayoría que Sí (**Gráfica 1-9**), el **Grupo 2** que No (**Gráfica 1-10**) y el **Grupo Global** muestra un equilibrio (**Gráfica 1-11**).



Gráfica 1–9. Alumnos que saben análizar (Grupo 1). Fuente: Elaboración propia (2008).



Gráfica 1–10. Alumnos que saben análizar (Grupo 2). Fuente: Elaboración propia (2008).



Gráfica 1–11. Alumnos que saben análizar (Grupo Global). Fuente: Elaboración propia (2008).

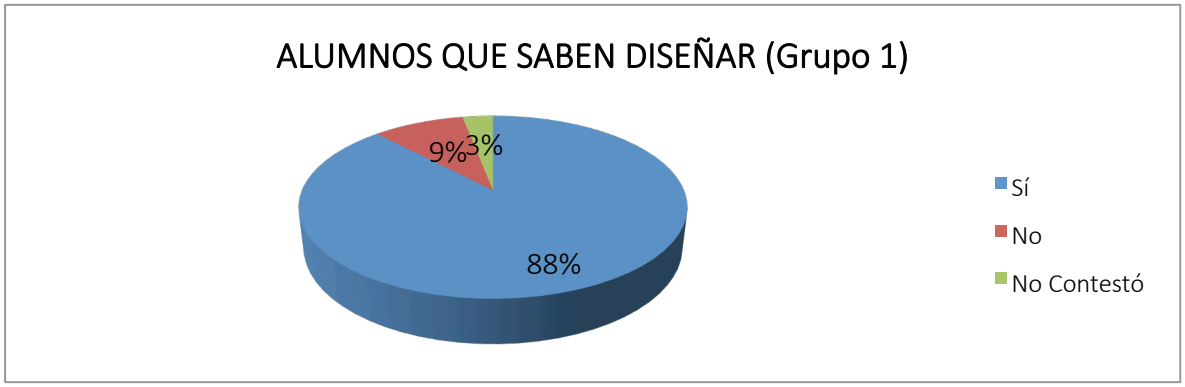
1.3 Ausencia de Diseño

El diseño da la pauta de cómo resolver un problema mediante un algoritmo o pseudocódigo y es el resultado del proceso de análisis, no obstante, al igual que ocurre con el análisis el diseño también es omitido por el alumno.

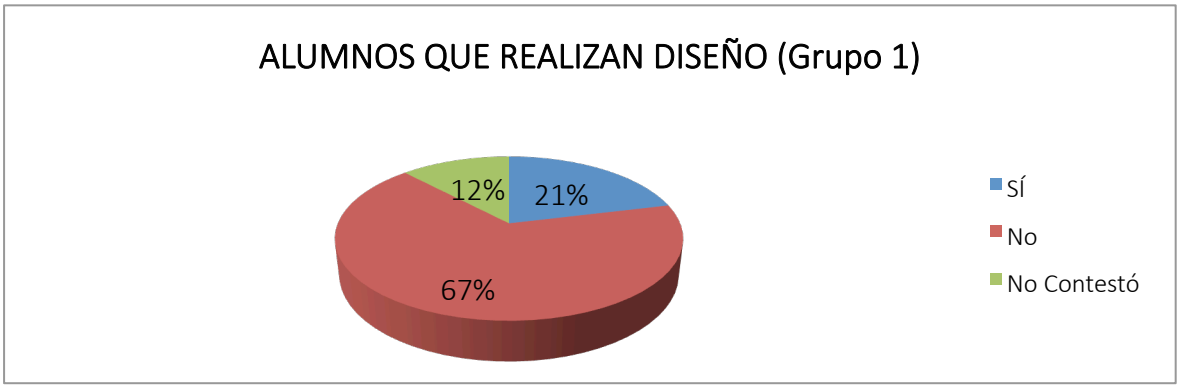
Cuando el alumno se encuentra por si solo a la tarea de programar, se enfrenta con la siguiente pregunta ¿Cómo le hago para…? Y no porque no se le haya enseñado las estructuras necesarias, sino más bien porque no tiene una guía que le indique lo que debe hacer y es precisamente el algoritmo el que responde a la pregunta.

Para conocer mejor la situación de los alumnos con respecto a este tema, se les preguntó si sabían diseñar, además, también se les preguntó si tenían el hábito de hacer algoritmos. A continuación, presentamos los resultados.

Es interesante ver como en el **Grupo 1** aunque la mayoría contestó que si sabe hacer un diseño (**Gráfica 1-12**), no tienen el hábito de realizarlo antes de comenzar a codificar (**Gráfica 1-13**)

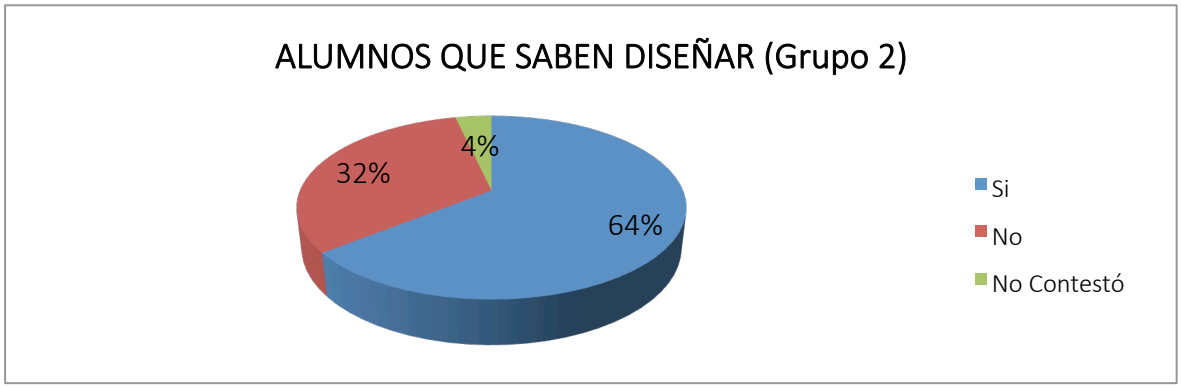


Gráfica 1–12. Alumnos que saben diseñar (Grupo 1). Fuente: Elaboración propia (2008).

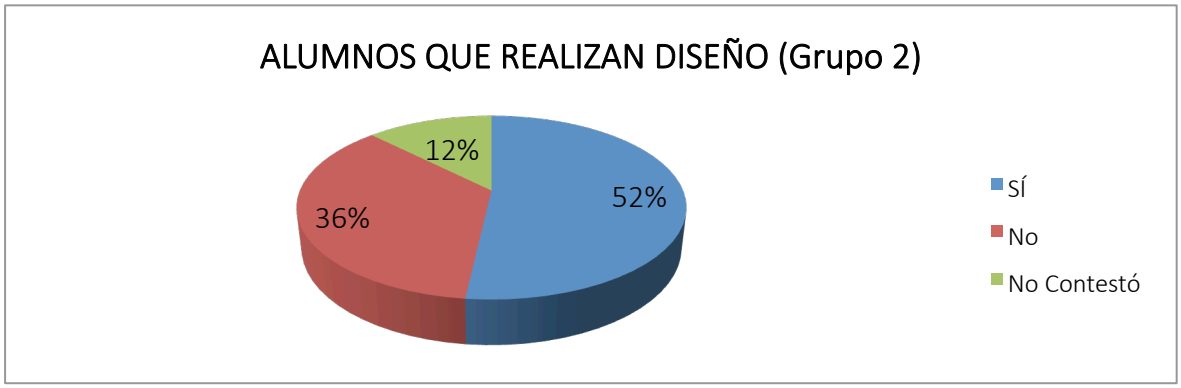


Gráfica 1–13. Alumnos que ealizan diseño (Grupo 1). Fuente: Elaboración propia (2008).

De igual manera el **Grupo 2** contestó que sí sabe realizar diseño (**Gráfica 1-14**), contradictoriamente contestaron que tienen el hábito de realizar diseño (**Gráfica 1-15**).

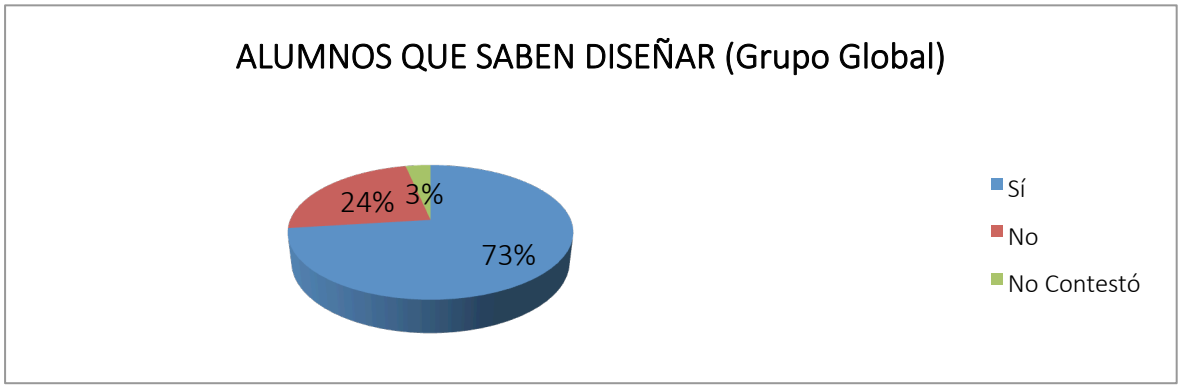


Gráfica 1–14. Alumnos que saben diseñar (Grupo 2). Fuente: Elaboración propia (2008).

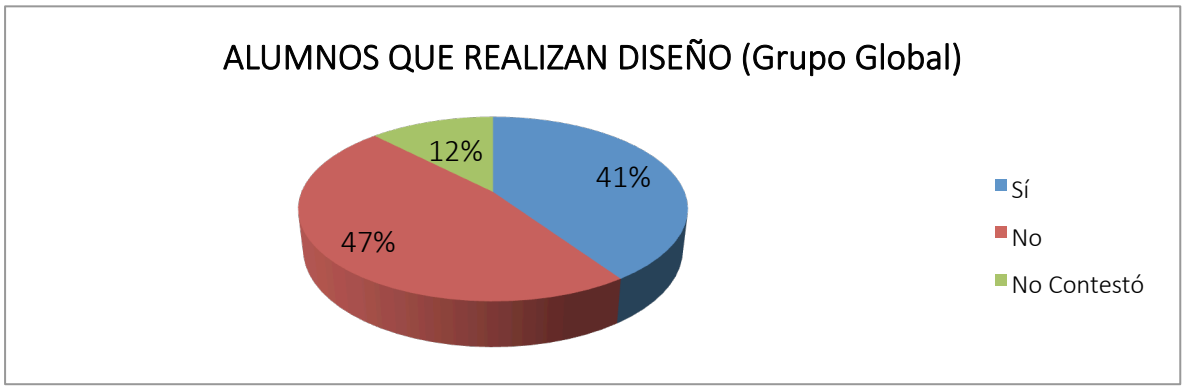


Gráfica 1–15. Alumnos que realizan diseño (Grupo 2). Fuente: Elaboración propia (2008).

Teniendo como resultado en el **Grupo Global** una mayoría en los alumnos que saben diseñar (**Gráfica 1-16**) pero una mayoría de alumnos que no realizan dicho diseño (**Gráfica 1-17**).



Gráfica 1–16. Alumnos que saben diseñar (Grupo Global). Fuente: Elaboración propia (2008).



Gráfica 1–17. Alumnos que realizan diseño (Grupo Global). Fuente: Elaboración propia (2008).

De los resultados mostrados podemos deducir que posiblemente los alumnos del **Grupo 1** al tener una mayor habilidad programando se saltan la fase del diseño, lo cual refleja un mal hábito que los afectará cuando se tengan que desarrollar en un ambiente laboral.

1.4 Desconocimiento del Lenguaje de Programación

Uno de los problemas que encontramos en las encuestas, es el desconocimiento del lenguaje de programación por parte de los alumnos, situación que afecta su aprendizaje, dado que, si los alumnos presentan deficiencias en su codificación, difícilmente podrán implementar un algoritmo. Los tres problemas principales que detectamos son:

1. Falta de entendimiento de la sintaxis del lenguaje. Al tener errores de sintaxis, el programa no compila, situación común incluso entre profesionistas durante el proceso de codificación. El problema radica cuando el alumno es incapaz de corregir los errores de sintaxis, lo cual no tiene que ver con la capacidad intelectual sino por falta de comprensión del lenguaje de programación. Un ejemplo típico de esto sería la finalización de las instrucciones con un punto y coma (;) y el cierre de llaves (}). Si bien es cierto que el ambiente de desarrollo (IDE) proporciona un detalle de los errores de compilación, también es cierto que el alumno no sabe cómo interpretarlos.

2. Falta de entendimiento de los elementos que conforman a las estructuras. Al no haber una comprensión del lenguaje por parte del alumno, este tendrá dificultades en la modificación de los elementos de las estructuras y en su adaptación a diferentes problemas, un ejemplo de esto sería la sintaxis del ciclo *for*, hemos observado una falta de comprensión de los elementos que la conforman, es decir, nos les queda claro la función de la variable de inicio o de su condición o del incremento. ¿Cuántos alumnos están conscientes de que es posible y correcto construir las siguientes definiciones de la estructura *for*?

- `for(i=1;i<=10;i=i+5)`
- `for(i=1;i!=5;i++ )`
- `for(i=1;i<10 && k>0;i++ )`
- `for(; i<10 ;)`
- `for(a=1;b<10;c--)`

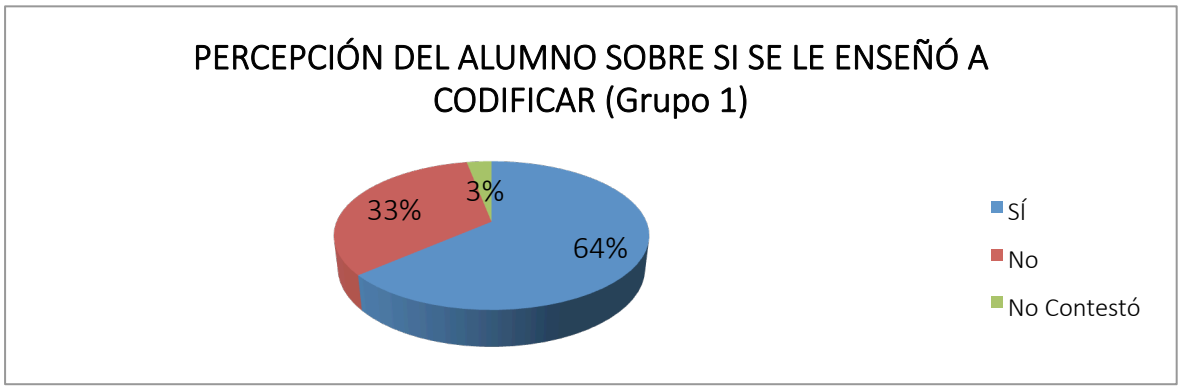
3. Falta de entendimiento en la estructuración de un programa. Otro de los problemas observados es el desconocimiento de la organización de los elementos que componen a un programa. Es tal el desconocimiento del lenguaje de programación que incluso se puede llegar a encontrar el siguiente fragmento de código, el cual fue tomado de

un desarrollo realizado por un alumno de la FES Aragón que se encontraba realizando su servicio social en la DGSCA (Hoy DGTIC) de la UNAM. Llama la atención que la definición de la función se encuentre dentro de una estructura *if* y que inmediatamente después de la definición de la función se realiza la llamada a ésta, lo cual evidentemente no es correcto dado que el alcance de la función queda definida dentro del cuerpo del *if*. Este código ejemplifica las deficiencias que tienen los alumnos, confirmando los problemas detectados.

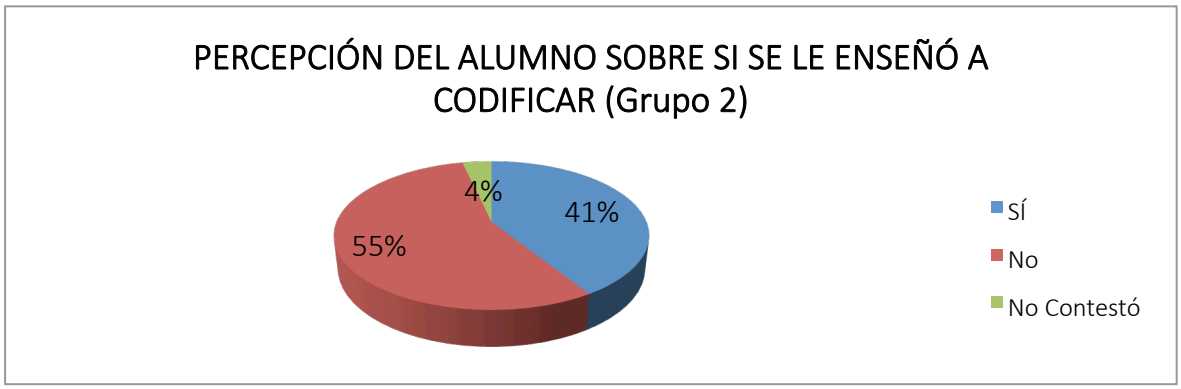
```
if (condición es verdadera){
      function nombreFuncion()
      {
            //Cuerpo de la función.
      }
      nombreFuncion();
}
```

Todas estas situaciones, generan en el alumno sentimientos de frustración provocándoles ideas negativas tales como "es difícil", "no puedo" o "no me gusta programar", ideas que perjudican el proceso de aprendizaje entre el alumno y el profesor.

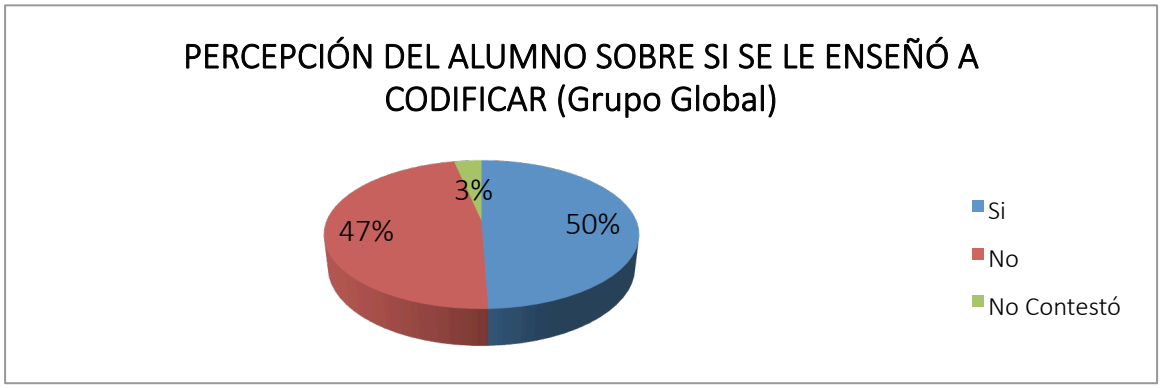
Al preguntar en la encuesta si se les enseñó a codificar a los alumnos, podemos observar una respuesta muy similar a las **Gráficas 1-6**, 1-**7** y 1-**8**. El **Grupo 1** indicó mayoritariamente que Sí (**Gráfica 1-18**), el **Grupo 2** que No (**Gráfica 1-19**) y encontramos nuevamente un equilibrio en el **Grupo Global** (**Gráfica 1-20**)



Gráfica 1–18. Percepción del alumno si se le enseño a codificar (Grupo 1). Fuente: Elaboración propia (2008).



Gráfica 1–19. Percepción del alumno sobre si se le enseño a codificar (Grupo 2). Fuente: Elaboración propia (2008).



Gráfica 1–20. Percepción del alumno sobre si se le enseño a codificar (Grupo Global). Fuente: Elaboración propia (2008).

1.5 Entendimiento de las estructuras

Otra situación que hemos observado está relacionada con las estructuras cuya función es similar, como son las estructuras de bifurcación y repetición. Los alumnos presentan dudas ya que no saben cuándo usar una u otra estructura.

El caso más típico es cuando el alumno le pregunta al profesor cuándo usar un ciclo *for* de un ciclo *while* o de un ciclo *do-while* y el profesor puede responder:

> *"El ciclo for lo utilizamos cuando de antemano conocemos el número de veces que se debe repetir el ciclo. El ciclo while cuando desconocemos el número de veces que deberá de repetir el ciclo pero con la posibilidad de que ni una sola vez se ejecute el ciclo y en el caso del ciclo do-while es igual al ciclo while pero*

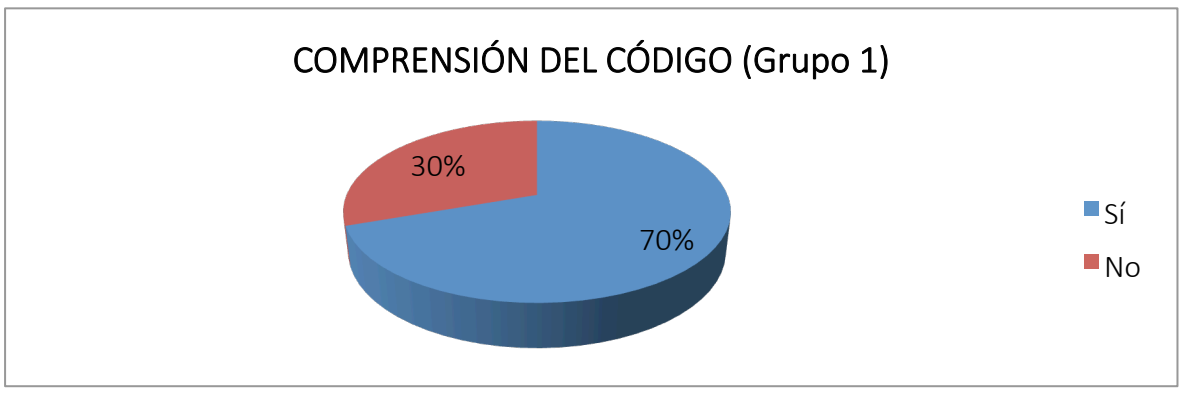
de antemano tenemos garantizado que el cuerpo del ciclo se ejecutará por lo menos una sola vez."

Aunque la respuesta anterior es correcta, esta sigue sin ser significativa para el alumno. En este punto es muy importante reflexionar la causa de esta situación, es decir, el por qué las explicaciones no son significativas para el alumno, una aproximación a este problema nos la puede dar la Doctora en Psicología Rocío Quesada:

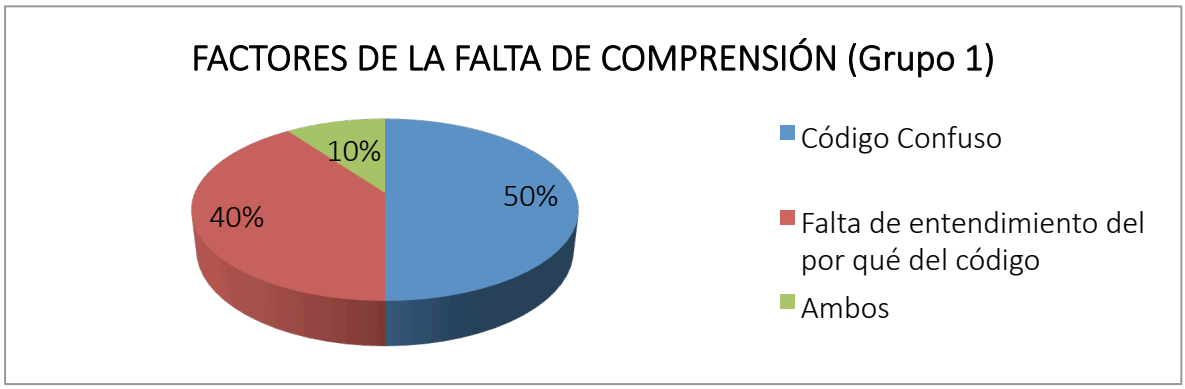
"Resulta indispensable iniciar la enseñanza a partir de lo que los alumnos saben acerca del tema y de sus ideas previas al mismo" y "A partir de la identificación de los conocimientos e ideas previos se prevé la forma como se preparará o dispondrá esta base de información para vincularla de la mejor manera con el conocimiento nuevo por aprender." Quesada, 2008.

Ahora bien, con base en este planteamiento el aprendizaje de un nuevo conocimiento implica vincular la información nueva con los conocimientos previos, siendo éstos lo que el alumno sabe acerca del tema en cuestión, pero ¿qué pasa si no existe conocimiento previo? O ¿cuáles conocimientos previos se deberían de tener para una clase de programación? Esta pregunta nos puede ayudar a acercarnos al origen del problema.

En la encuesta les preguntamos a los alumnos si comprendían el código que el profesor les enseñaba, la mayoría del **Grupo 1** contestó que Sí (**Gráfica 1-21**), resaltando que la mayoría que contestó No consideran como mayor dificultad que el código les resultaba confuso (**Gráfica 1-22**).

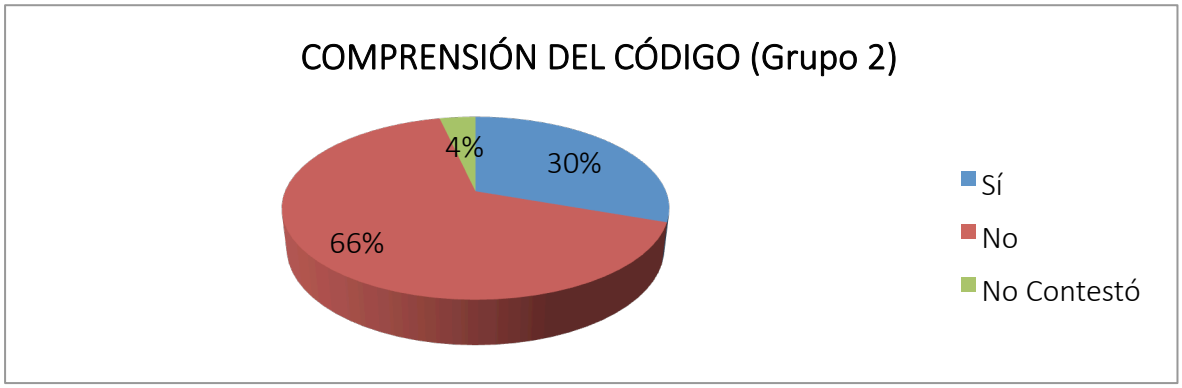


Gráfica 1–21 Comprención del código (Grupo 1). Fuente: Elaboración propia (2008).

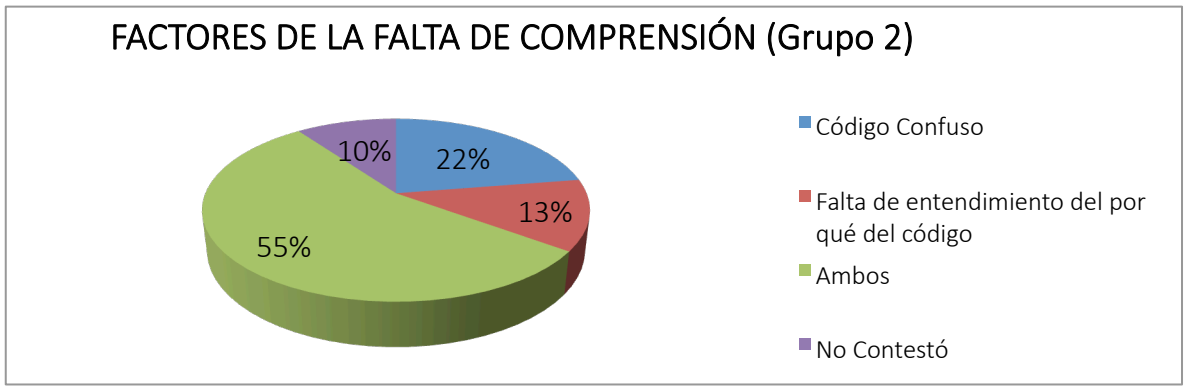


Gráfica 1–22. Factores de la falta de comprensión (Grupo 1). Fuente: Elaboración propia (2008).

En el **Grupo 2** la mayoría contestó que No (**Gráfica 1-23**) presentando problemas en la falta de entendimiento del porqué del código y al mismo tiempo resultándoles confuso (**Gráfica 1-24**).

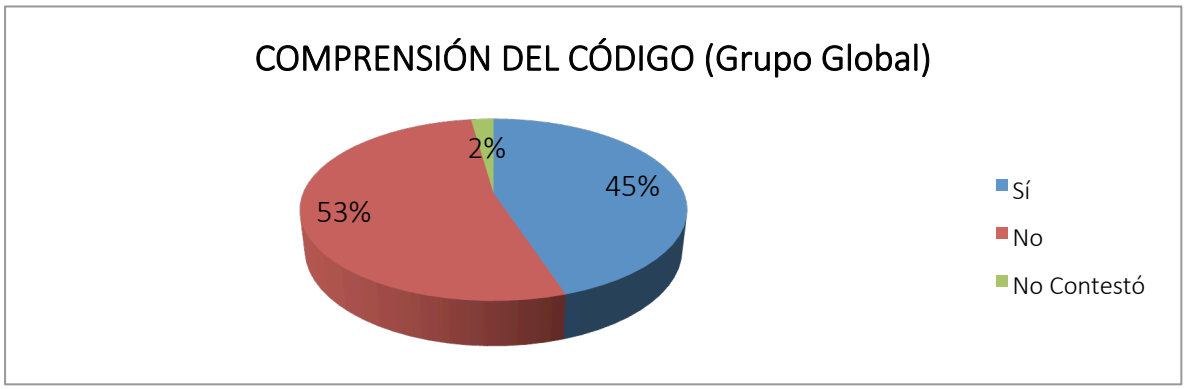


Gráfica 1–23. Comprención del código (Grupo 2). Fuente: Elaboración propia (2008).

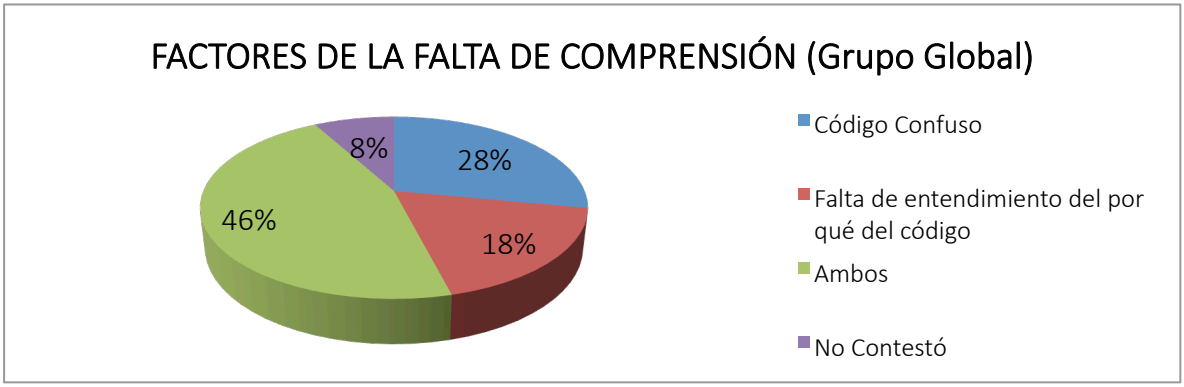


Gráfica 1–24. Factores de la falta de comprensión (Grupo 2). Fuente: Elaboración propia (2008).

Finalmente, en el **Grupo Global** volvemos a ver la mayoría contestando que No comprenden el código (**Gráfica 1-25**) por las mismas razones de falta de entendimiento del porqué del código y al mismo tiempo resultándoles confuso (**Gráfica 1-26**).



Gráfica 1–25. Comprención del código (Grupo Global). Fuente: Elaboración propia (2008).



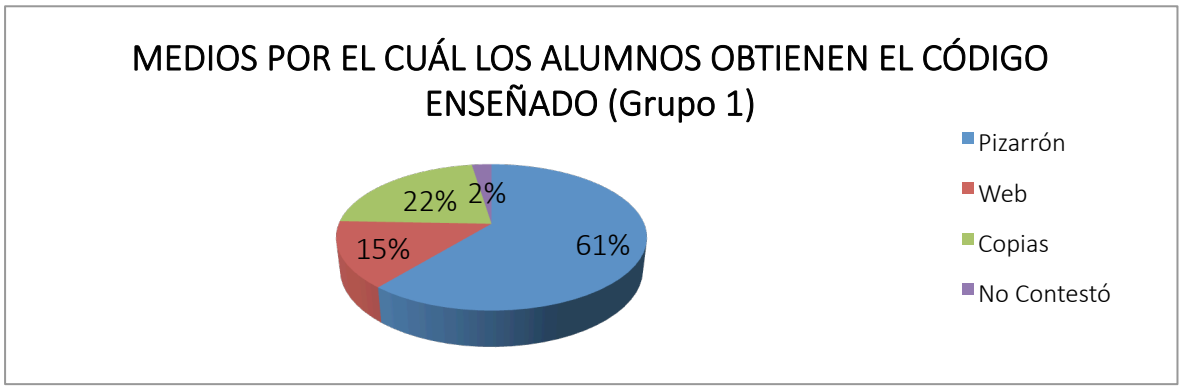
Gráfica 1–26. Factores de la falta de comprensión (Grupo Global). Fuente: Elaboración propia (2008).

1.6 Pérdidas en la adquisición del código

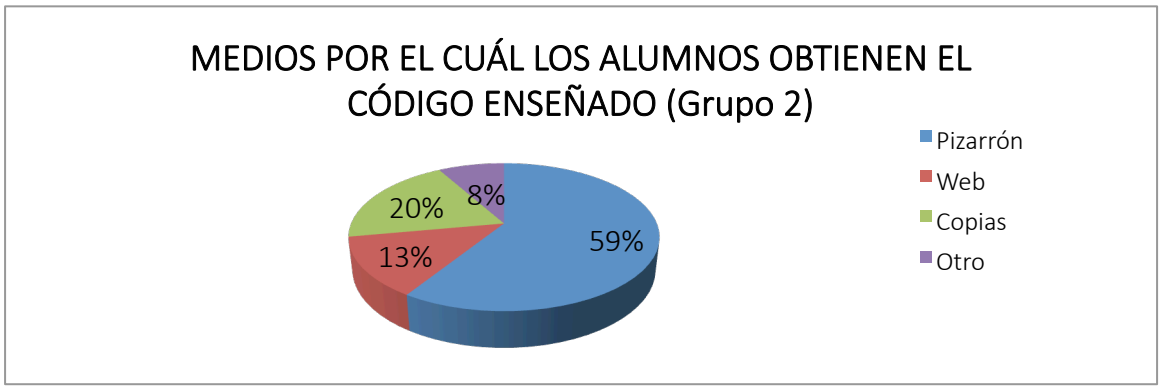
De acuerdo a la encuesta un problema común al que se enfrentan los alumnos tiene que ver con la adquisición de los ejemplos de código enseñado por el profesor, generalmente el alumno copia el código del pizarrón a su cuaderno y posteriormente lo transcribe al ambiente de desarrollo (IDE) en este proceso de transcripción suele haber pérdidas, lo que provoca que el código no compile o no realice su tarea apropiadamente. Esta situación afecta el proceso de aprendizaje del alumno dado que al no contar con un código íntegro éste no puede analizarlo para comprender su funcionamiento.

Identificamos dos causas del por qué el alumno copia mal el código desde el origen (generalmente pizarrón). Primeramente para un alumno que no está familiarizado con los lenguajes de programación, ver un código equivale a ver un conjunto de símbolos y palabras que no tienen un significado para él. La segunda causa está relacionada con el uso de un solo color para escribir el código en el pizarrón y en consecuencia al ser todo homogéneo resulta más difícil al alumno leer y distinguir la sintaxis del código.

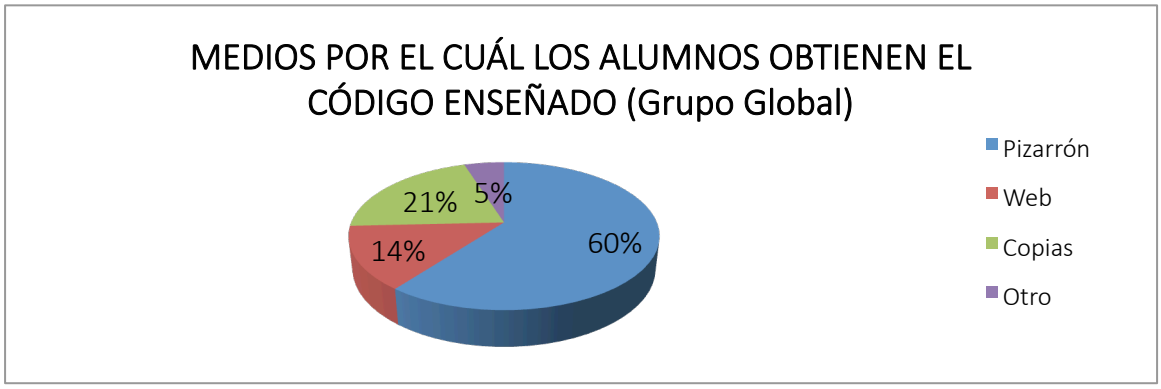
Preguntándole al **Grupo 1** (**Gráfica 1-27**) y al **Grupo 2** (**Gráfica 1-28**), teniendo como total al **Grupo Global** (**Gráfica 1-29**), cómo obtenían el código que el profesor les enseñaba en clase, podemos observar que el pizarrón sigue siendo el principal para los tres, teniendo de esta forma una mayor posibilidad de pérdida de código en la transcripción.



Gráfica 1–27. Medios por el cuál los alumnos obtienen el código enseñado (Grupo 1). Fuente: Elaboración propia (2008).



Gráfica 1–28. Medios por el cuál los alumnos obtienen el código enseñado (Grupo 2). Fuente: Elaboración propia (2008).



Gráfica 1–29. Medios por el cuál los alumnos obtienen el código enseñado (Grupo Globlal). Fuente: Elaboración propia (2008).

1.7 Herramientas de desarrollo

Los alumnos se enfrentan al problema de no conocer las herramientas de desarrollo, para ser más precisos nos referimos al Entorno de Desarrollo Integrado, mejor conocido como IDE. Los conocimientos básicos que los alumnos deberían de saber al momento de trabajar con un IDE son:

- Creación de un proyecto.
- Compilación.
- Depuración.
- Interpretación de errores del compilador.

Asimismo, hay que tomar en cuenta que cada vez los IDE´s son más complejos con forme pasa el tiempo y estos pueden resultar intimidantes para los alumnos. A continuación se muestra la evolución de los IDE's.

- Turbo C: Fue el IDE más popular para el desarrollo en C en la década de los años 80. Se le considera el primer IDE para C disponible para MS-DOS. Por su simplicidad era fácil de aprender y usar. (Imagen 1-3)

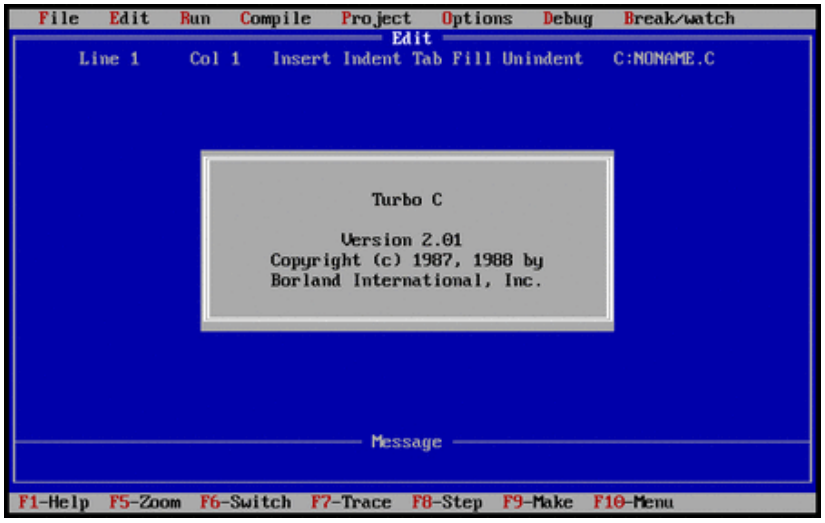


Imagen 1–3 . Turbo C. Fuente: Elaboración propia (2017).

- Borland C++. IDE para el desarrollo en C++ en la década de los años 90. Este IDE fue diseñado para Windows. Por su simplicidad era fácil de aprender y usar. (**Imagen 1-4**)

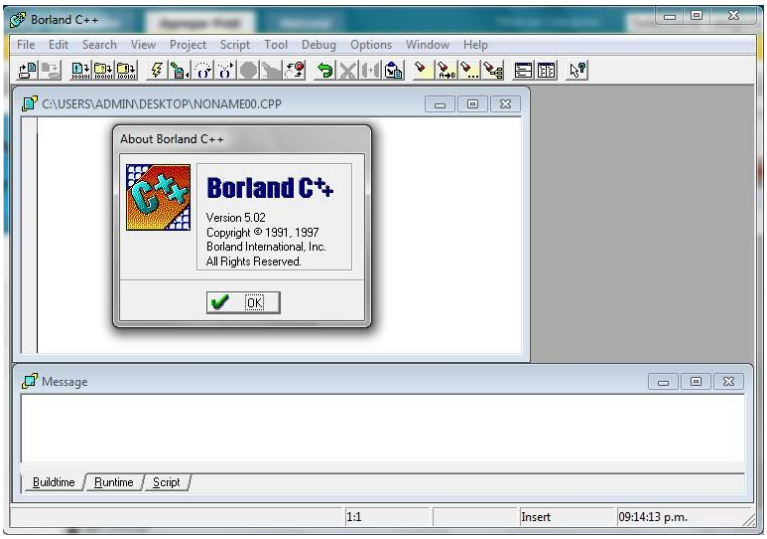


Imagen 1–4. Borland C++. Fuente: Elaboración propia (2017).

- Eclipse. IDE muy popular para el desarrollo en Java. Su uso inició a partir del año 2004 y actualmente se mantiene vigente como IDE de Java. Este IDE fue diseñado para ser Multiplataforma. Contiene muchas herramientas para el programador lo que lo convierte en un IDE complejo y difícil de aprender a usar. (**Imagen 1-5**)

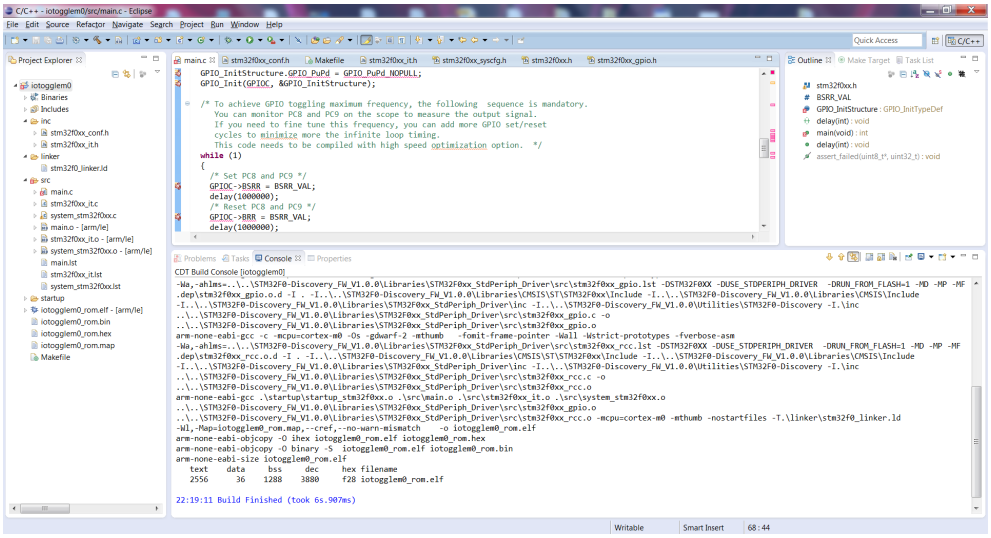


Imagen 1–5. Eclipse. Fuente: Elaboración propia (2017).

- Visual Studio. IDE multilenguaje muy popular. Su uso inició a partir del año 2002 y actualmente se mantiene vigente. Contiene muchas herramientas para el programador lo que lo convierte en un IDE complejo pero fácil de aprender. En la imagen siguiente se muestra el IDE desplegando un diagnóstico de memoria de la aplicación en ejecución. (**Imagen 1-6**)

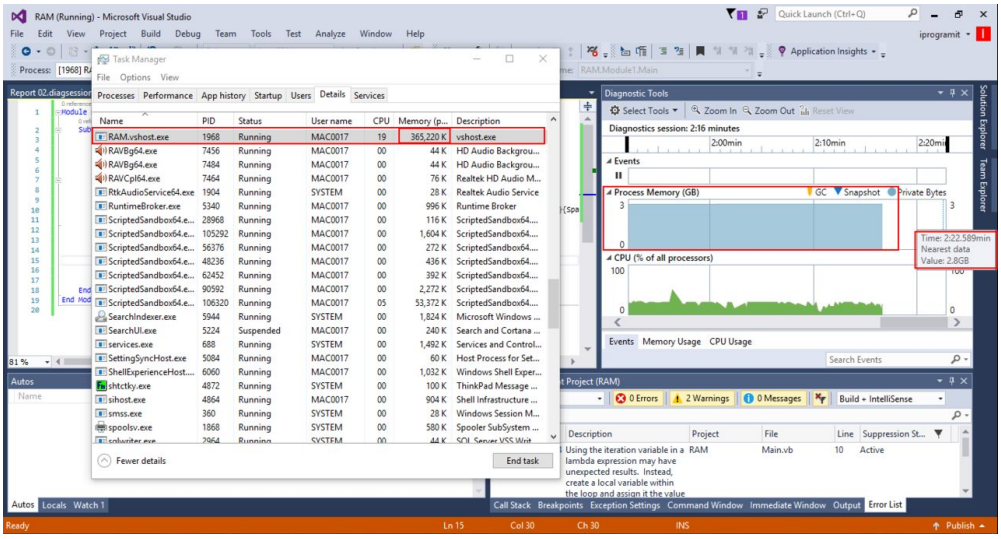
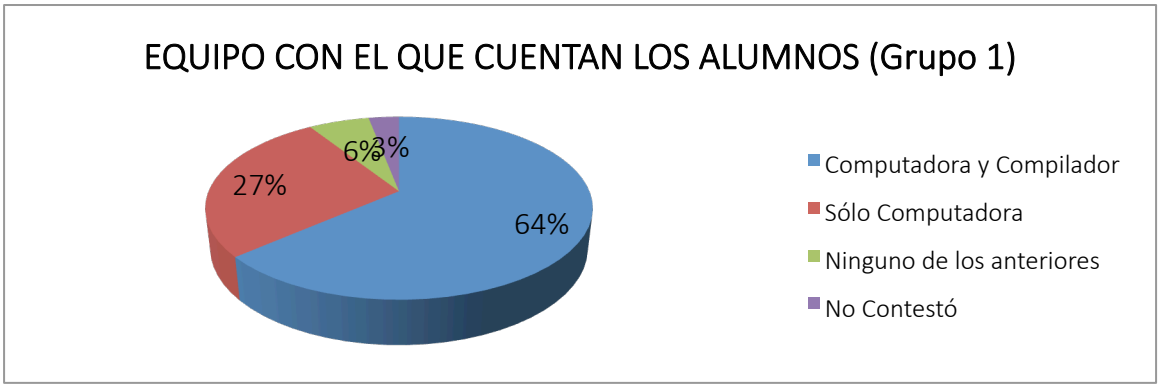
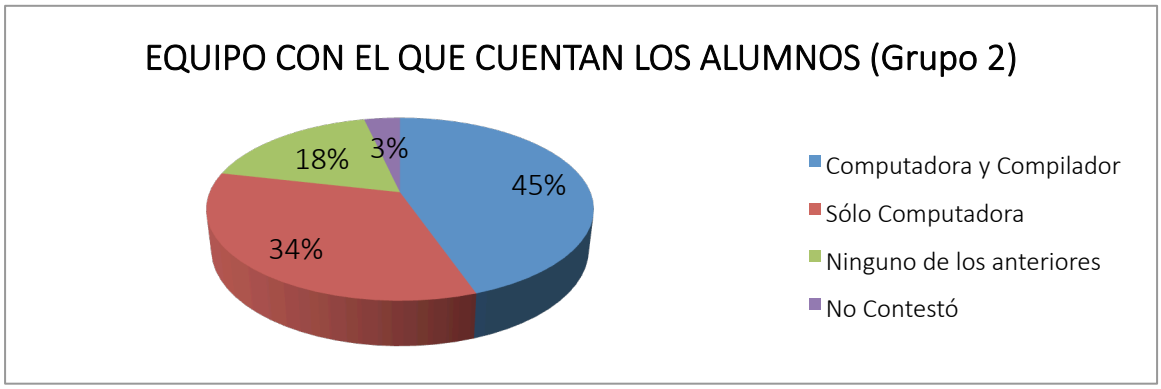


Imagen 1–6. Visual Studio. Fuente: Elaboración propia (2017).

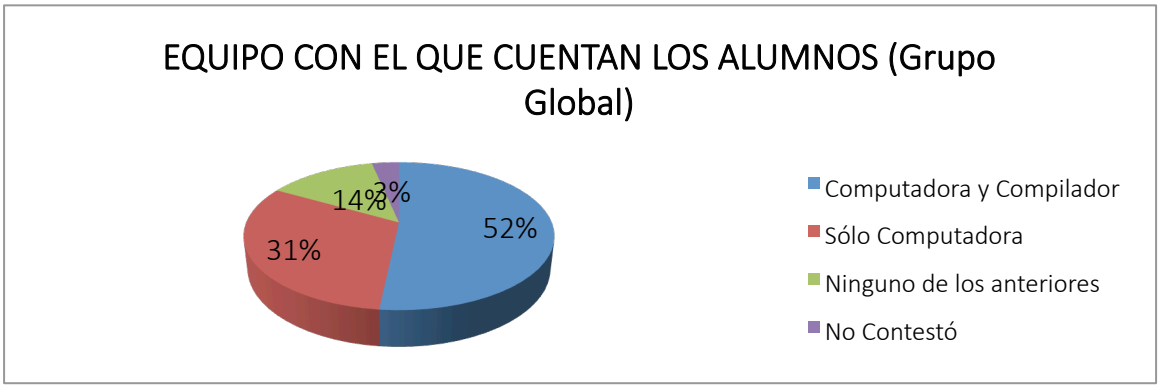
Sumando el problema de desconocer los IDE's nos encontramos que no todos los alumnos cuentan con las herramientas para trabajar en casa (computadora y compilador), lo cual puede causar una menor probabilidad de aprendizaje del alumno. (**Gráfica 1-30**, **Gráfica 1-31** y **Gráfica 1-32**)



Gráfica 1–30. Equipo con el que cuentan los alumnos (Grupo 1). Fuente: Elaboración propia (2008).



Gráfica 1–31. Equipo con el que cuentan los alumnos (Grupo 2). Fuente: Elaboración propia (2008).

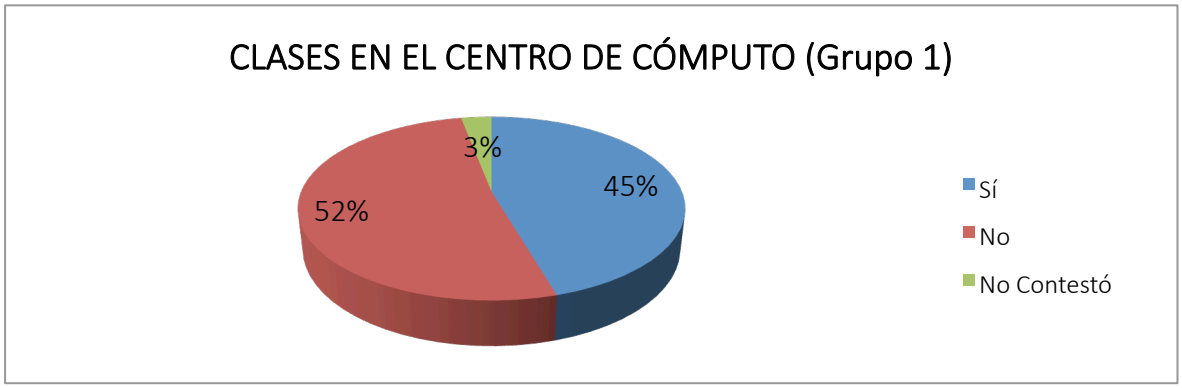


Gráfica 1–32. Equipo con el que cuentan los alumnos (Grupo Global). Fuente: Elaboración propia (2008).

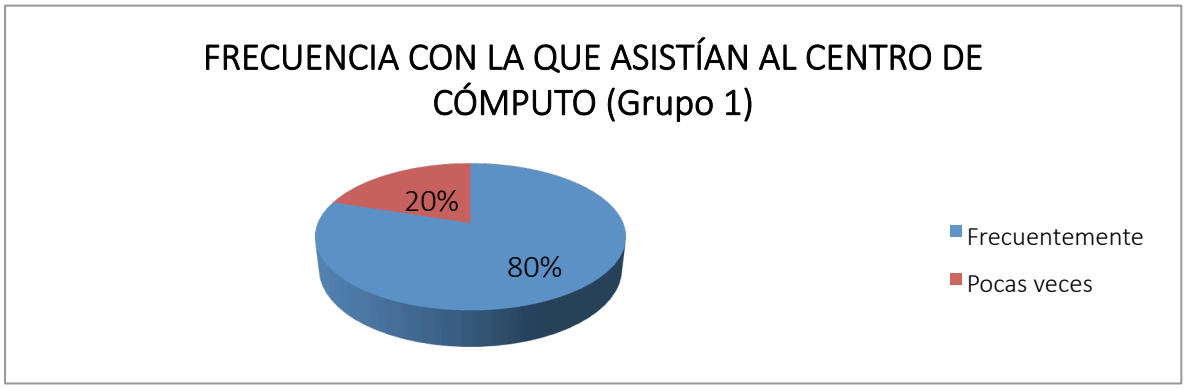
1.8 Clases en Centro de Cómputo

De acuerdo con las encuestas, los alumnos consideran como problema la falta de clases en un centro de cómputo, sin embargo, analizando los problemas que presentan antes de la codificación (mencionados en este capítulo) el tomar clases en el centro de cómputo no representaría una mejora significativa.

En el **Grupo 1** podemos observar un equilibrio entre los alumnos que asistían a clases en el centro de cómputo (**Gráfica 1-33**), y los que lo hacían asistían frecuentemente (**Gráfica 1-34**).

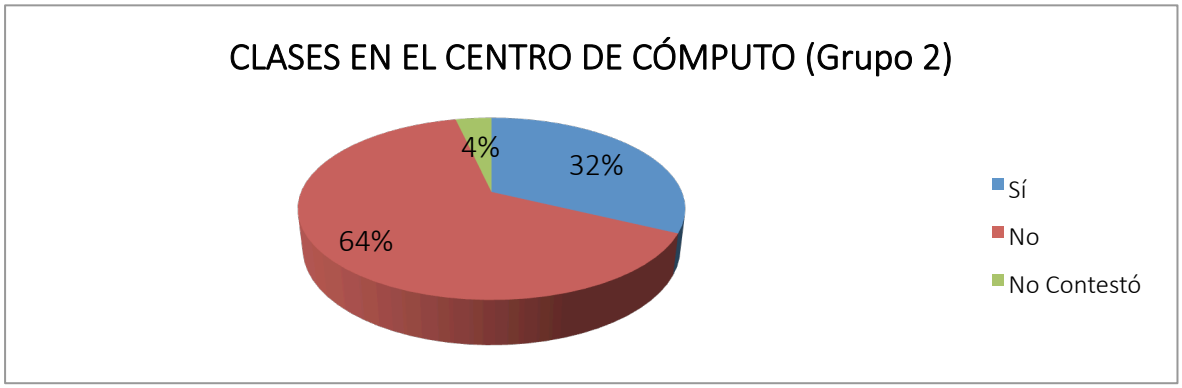


Gráfica 1–33. Clases en el centro de cómputo (Grupo 1). Fuente: Elaboración propia (2008).

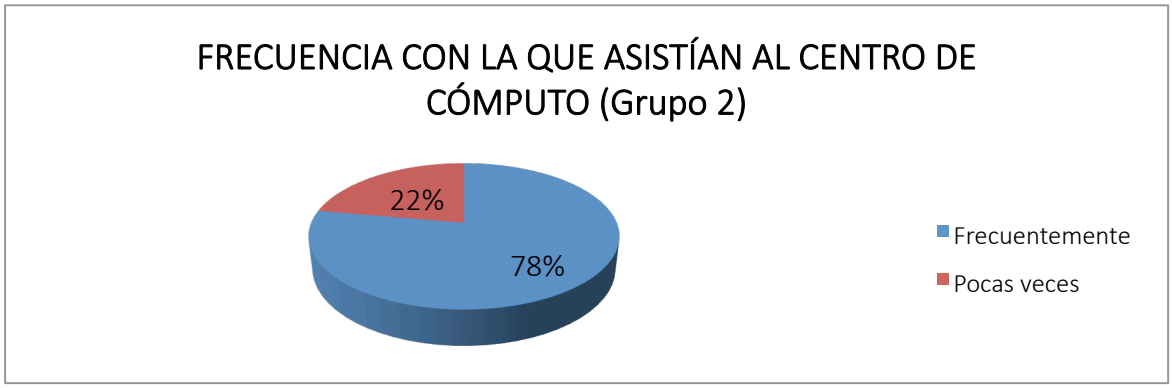


Gráfica 1–34. Frecuencia con la que asistían al centro de cómputo (Grupo 1). Fuente: Elaboración propia (2008).

En el **Grupo 2** podemos observar que la mayoría no asistían a clases en el centro de cómputo (**Gráfica 1-35**) mientras que los que lo hacían, asistían frecuentemente (**Gráfica 1-36**).

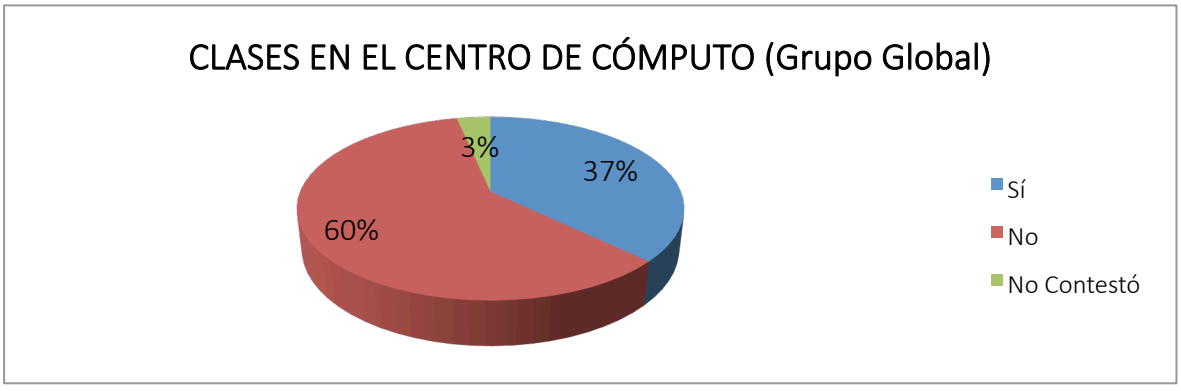


Gráfica 1–35. Clases en el centro de cómputo. Fuente: Elaboración propia (2008).

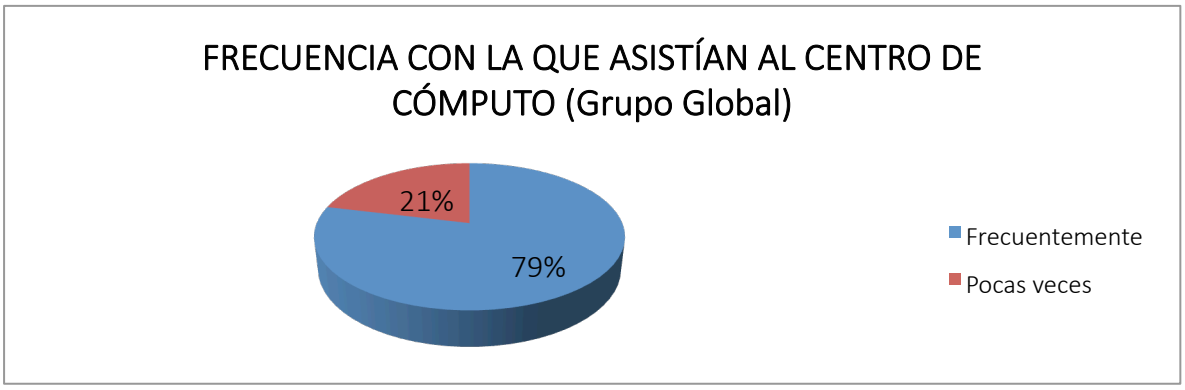


Gráfica 1–36. Frecuencia con la que asistían al centro de cómputo (Grupo 2). Fuente: Elaboración propia (2008).

En el **Grupo Global** observamos que al igual que el **Grupo 2** la mayoría no asistía a clases en el centro de cómputo (**Gráfica 1-37**), y los que lo hacían asistían frecuentemente (**Gráfica 1-38**).



Gráfica 1–37. Clases en el centro de cómputo (Grupo Global). Fuente: Elaboración propia (2008).



Gráfica 1–38. Frecuencia con la que asistían al centro de cómputo (Grupo Global). Fuente: Elaboración propia (2008).

1.9 Desmotivación

Podemos definir la motivación la como la atracción que siente el alumno por alcanzar un objetivo, que en este caso sería aprender a programar, sin importar el esfuerzo requerido.

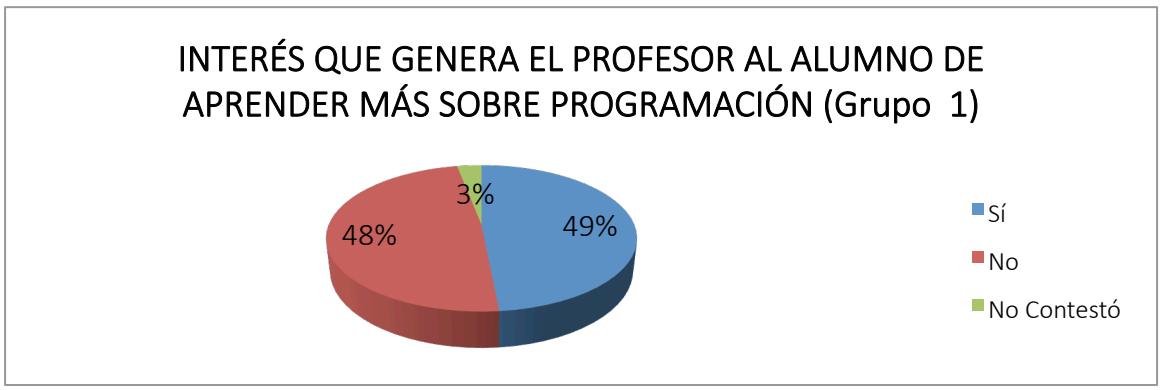
La motivación juega un papel muy importante porque constituye un factor que condiciona la capacidad de aprender y además es el motor del mismo. La ausencia de esta hace complicada la tarea del profesor y del alumno.

Con base en los problemas mencionados anteriormente no es de sorprender que exista una desmotivación entre los alumnos, la cual crea limitaciones. Entre las consecuencias de esto tenemos:

- Ausencia de las expectativas de éxito.
- Falta de incentivos para el estudio.
- Aburrimiento crónico.
- Apatía escolar.
- Decepción constante.
- Disminución de la propia autoestima.

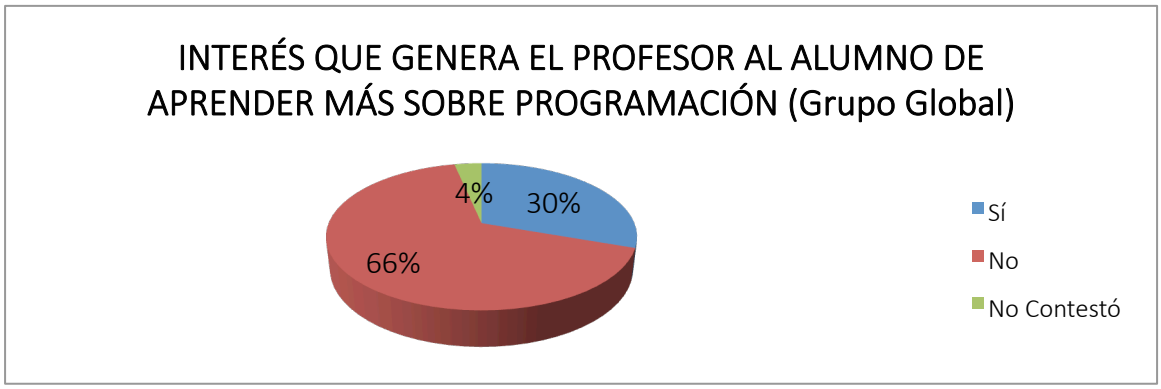
Aunque es cierto que la motivación debe ser un factor interno del alumno también ayuda que los profesores generen este interés por aprender.

Al preguntarle a los alumnos si su profesor les generaba este interés por aprender el **Grupo 1** se mostró equilibrado (**Gráfica 1-39**) mientras el **Grupo 2** (**Gráfica 1-40**) y el **Grupo Global** (**Gráfica 1-41**) se inclinó a contestar que No. Podemos observar una desmotivación considerable por parte de los alumnos, no es de extrañar que se tenga un porcentaje de alumnos con carencias de programación.



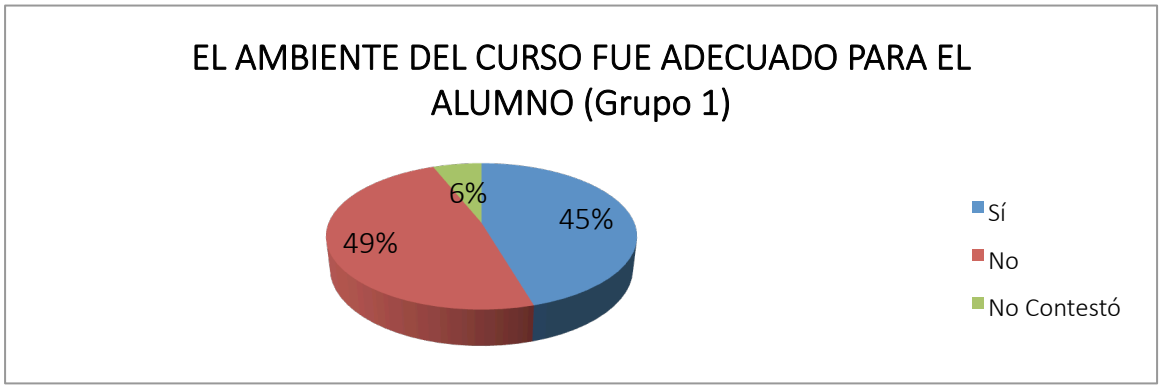
Gráfica 1–39. Interés que genera el profesor al alumno de aprender más sobre programación (Grupo1). Fuente: Elaboración propia (2008).

Gráfica 1–40. Interés que genera el profesor al alumno de aprender más sobre programación (Grupo 2). Fuente: Elaboración propia (2008).

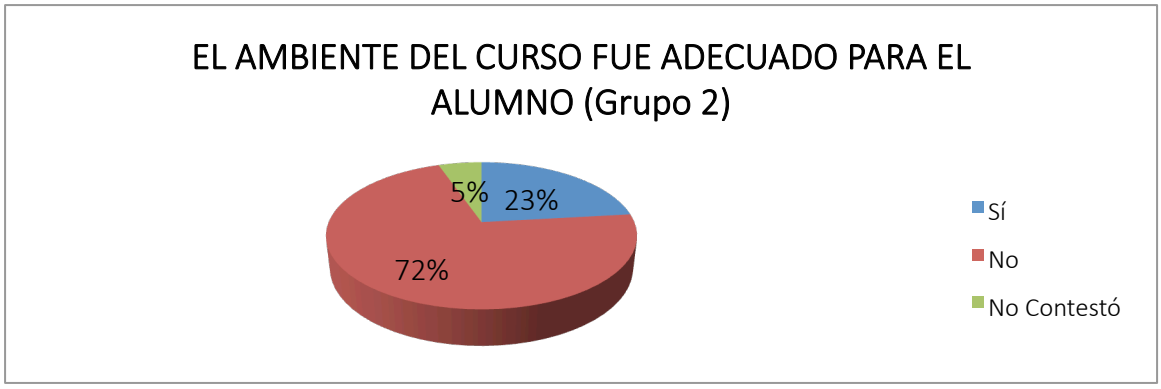


Gráfica 1–41. Interés que genera el profesor al alumno de aprender más sobre programación.

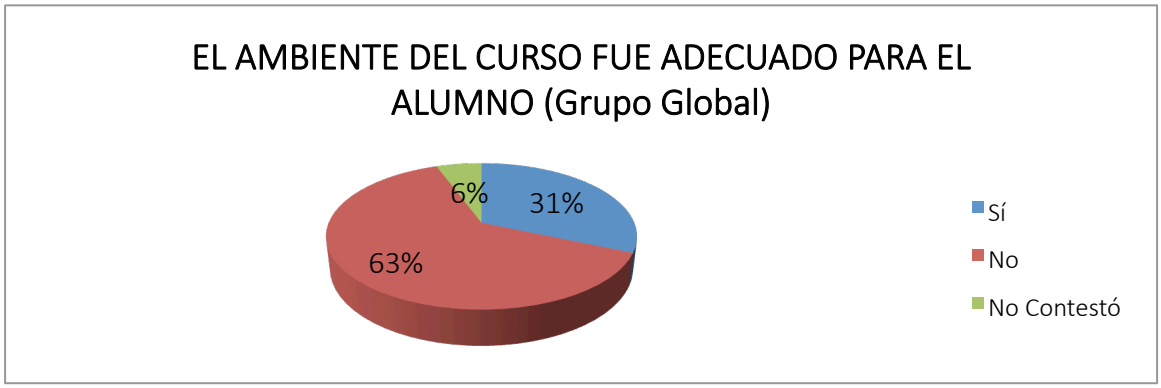
Preguntando si consideraban que el curso había sido adecuado para ellos, tanto el **Grupo 1** (**Gráfica 1-42**), **Grupo 2** (**Gráfica 1-43**) y **Grupo Global** (**Gráfica 1-44**) contestaron que No.



Gráfica 1–42. El ambiente del curso fue adecuado para el alumno (Grupo 1). Fuente: Elaboración propia (2008).

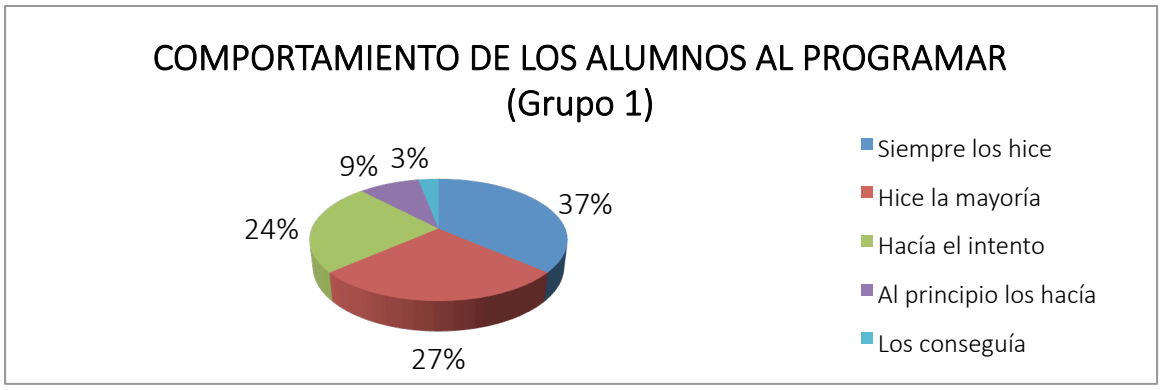


Gráfica 1–43. El ambiente del curso fue adecuado para el alumno (Grupo 2). Fuente: Elaboración propia (2008).

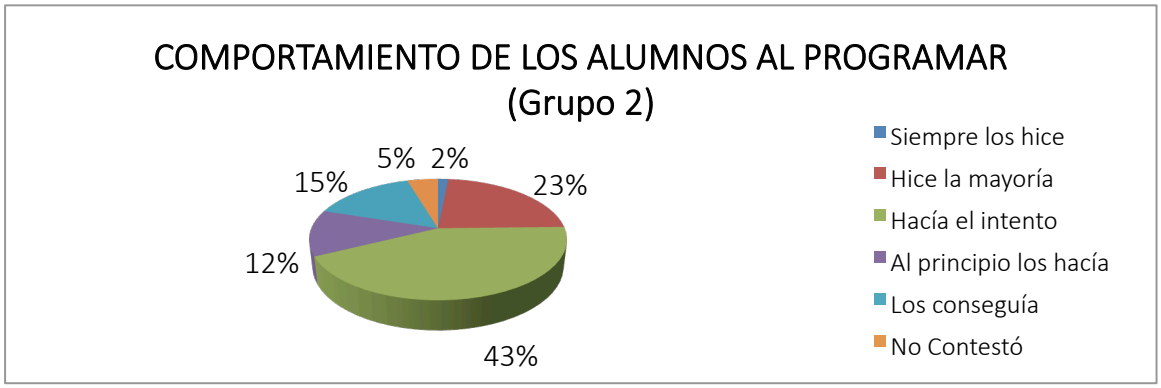


Gráfica 1–44. El ambiente del curso due adecuado para el alumno (Grupo Global). Fuente: Elaboración propia (2008).

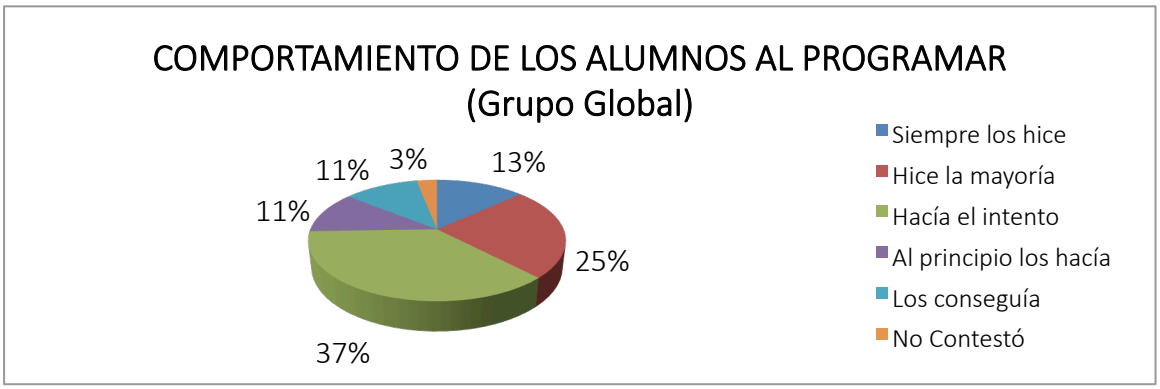
Al preguntarles si intentaban realizar los programas que los profesores les dejaban encontramos un desinterés general tanto en el **Grupo 1** (**Gráfica 1-45**), el **Grupo 2** (**Gráfica 1-46**) y el **Grupo Global** (**Gráfica 1-47**).



Gráfica 1–45. Comportamiento de los alumnos al programar (Grupo 1). Fuente: Elaboración propia (2008).

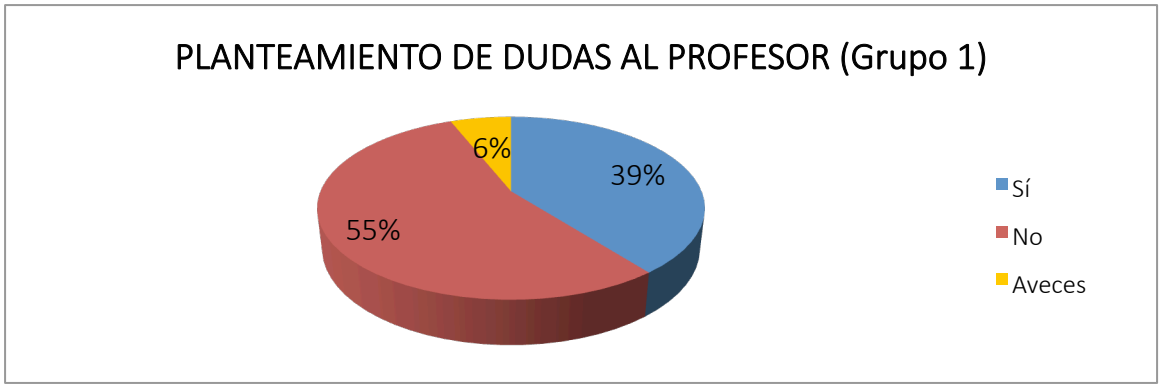


Gráfica 1–46. Comportamiento de los alumnos al programar. Fuente: Elaboración propia (2008).

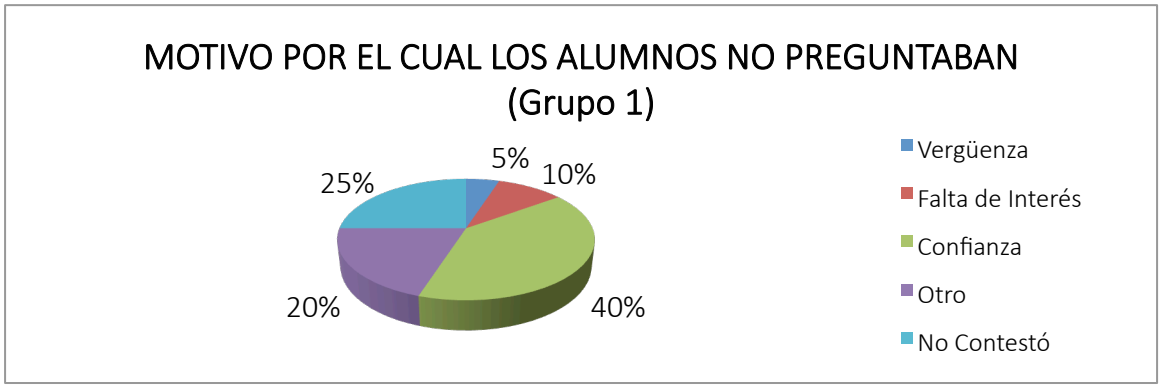


Gráfica 1–47. Comportamiento de los alumnos al programar (Grupo Global). Fuente: Elaboración propia (2008).

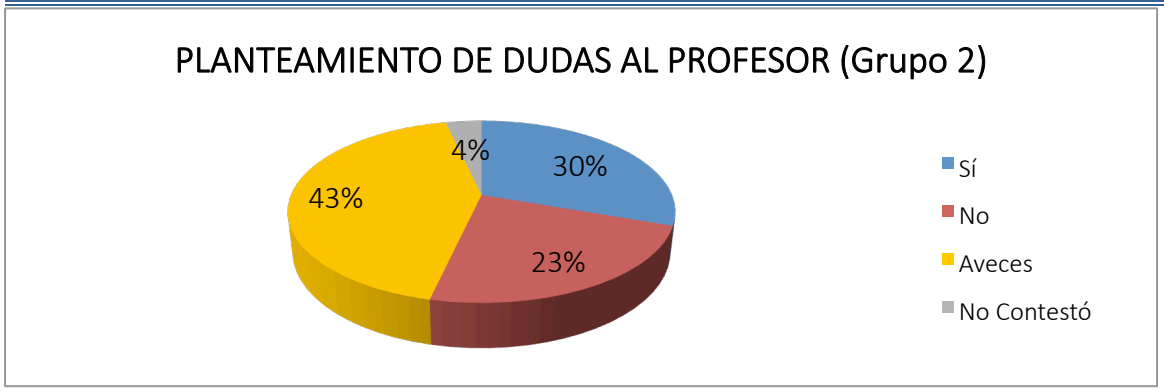
Algo alarmante con lo que nos encontramos es que los alumnos de los tres grupos al tener dudas durante las clases, preferían no preguntar, más alarmante aún es que la principal razón para no hacerlo fue la falta de confianza que les daba el profesor, como lo muestran las siguientes seis gráficas del **Grupo 1** (**Gráfica 1-48** y **Gráfica 1-49**), del **Grupo 2** (**Gráfica 1-50** y **Gráfica 1-51**) y del **Grupo Global** (**Gráfica 1-52** y **Gráfica 1-53**)



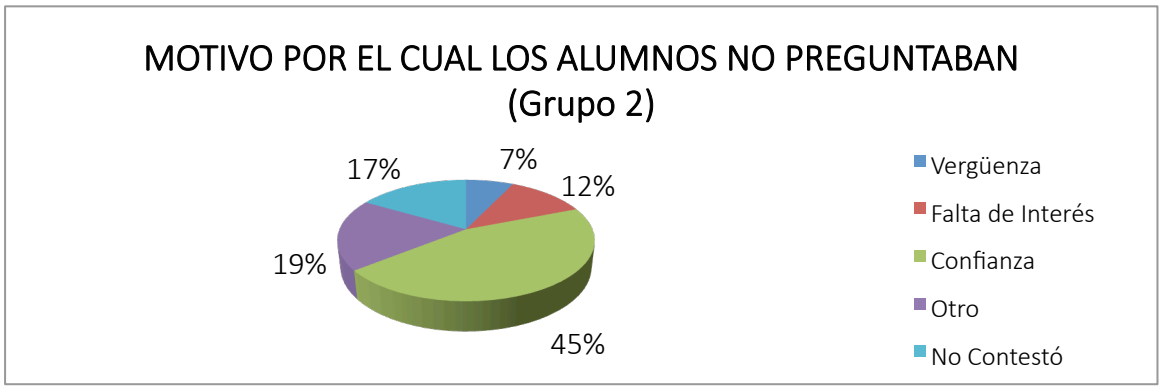
Gráfica 1–48. Planteamiento de dudas al profesor (Grupo 1). Fuente: Elaboración propia (2008).



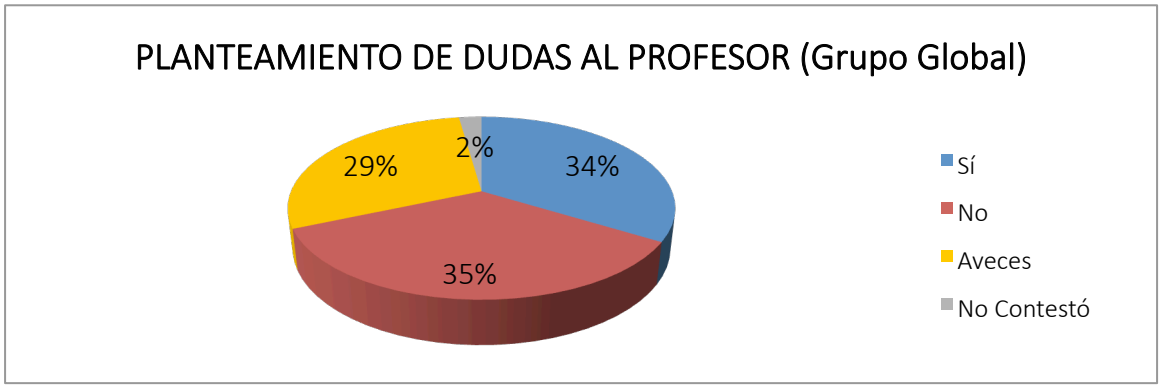
Gráfica 1–49. Motivo por el cual los alumnos no preguntaban (Grupo 1). Fuente: Elaboración propia (2008).



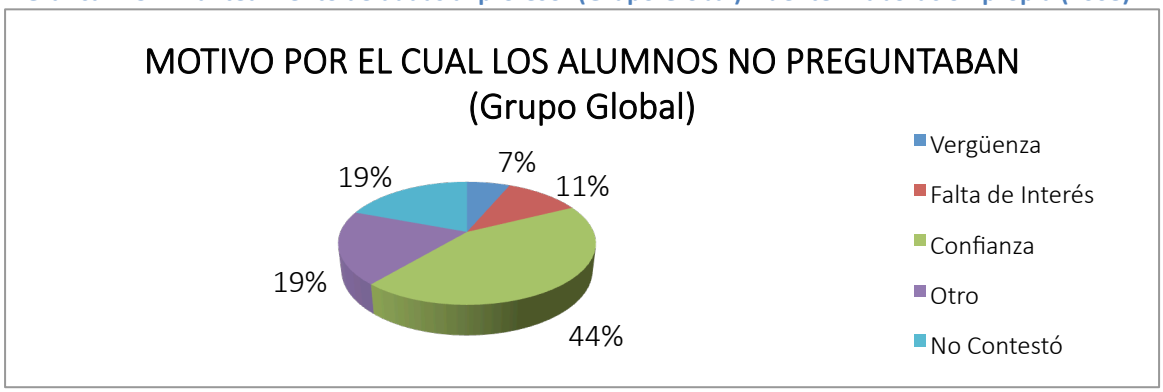
Gráfica 1–50. Planteamiento de dudas al profesor (Grupo 2). Fuente: Elaboración propia (2008).



Gráfica 1–51. Motivo por el cual los alumnos no preguntaban (Grupo 2). Fuente: Elaboración propia (2008).



Gráfica 1–52. Planteamiento de dudas al profesor (Grupo Global). Fuente: Elaboración propia (2008).



Gráfica 1–53. Motivo por el cual los alumnos no preguntaban (Grupo Global). Fuente: Elaboración propia (2008).

1.10 Perspectiva del Alumno del Problema.

En la encuesta se dejaron dos preguntas abiertas para permitirle al alumno expresar su opinión de los problemas que tuvo durante su proceso de aprendizaje. A continuación enlistamos sus respuestas.

¿Cuáles son los problemas que enfrentas o enfrentaste al aprender programar?

- Mis cursos han sido buenos, no me quejo de ellos.
- No haber obtenido información clara del cómo desarrollar el algoritmo y pseudocódigo.
- Los problemas son cuando tienes que pasar tu pseudocódigo al lenguaje de programación.
- Cómo entender, cómo estructurar un programa, cómo saber que poner para que corra el programa.
- Pues más que nada, mi principal problema es pasar los algoritmos o en este caso la idea en la que pienso hacer el programa y pasarlo a la computadora.
- En estructurar el programa.
- El diseño de algoritmos.
- El estudio y memorización de los lenguajes, el no saber cómo utilizar de forma correcta los ciclos y demás sintaxis.
- La lógica de programar y desconocimiento de varias funciones del lenguaje.
- Pues la lógica que debó de seguir.
- El código en sí y las estructuras en que deben ir.
- Los errores de sintaxis y lógicos.
- Reconocer el lenguaje de programación.
- Que no saben explicar cómo y para que usar estructuras.
- El código se me hacia un poco confuso.
- Me confundo con un sin fin de líneas.
- El entender los códigos y ver cuál es el mejor método para atacar el problema.
- A desarrollar los pseudocódigos y a programar en java.

- No entender el problema o lo que se pide al programa.
- Me falta conocimiento del lenguaje.
- Se me dificulta entenderlo (me confundo).
- Cambiar mi forma de pensar, todas las funciones y cada sentencia y cómo y cuándo aplicar cada sentencia.
- Que el profesor escriba el código en el pizarrón, pero cuando llega uno a programarlo no compila y no podía preguntarle a nadie.
- Que al leer un libro o tener el código de algún programa no pueda entender el porqué de las cosas, el orden o la función de algo, me es confuso.
- Entender lo básico ¿qué es un compilador? ¿Cómo programar? ¿Cuántas directivas?
- Que el primer curso era malo pues no te enseñaban lo básico y de ahí querían que ya supieras programar.
- Varios, porque los maestros llegan a la clase queriendo que uno ya sepa programar y se supone que por eso entre a la carrera para aprender.
- Que los profesores dan por hecho que sabes programar y solo dejan programas sin antes haberte enseñado las bases.
- Los maestros daban por hecho que ya sabíamos programar y tuve que aprender de los libros porque nunca nos enseñaron en la FES.
- No tener suficientes bases desde la prepa.
- Que no sabía nada sobre el tema pues casi no vi nada de eso en prepa.
- Los problemas son la falta de estrategia de parte de los profesores al tratar de enseñarnos un lenguaje.
- Pues una extraña manera de enseñar por parte de los profesores donde quedaba más en el alumno una extra responsabilidad por aprender lo que no se veía en clase por otros medios (libros, internet, conocidos).
- Que se necesita mucha paciencia por parte del profesor, la forma de enseñar del profesor, no hay acceso al equipo de cómputo.
- Al no entender bien y no practicar al momento que se está enseñando.
- El no practicar como debería y aprender el dónde aplicar los conocimientos.

- Para mi es que no nos dan la explicación de cada sentencia y por lo mismo no sé en donde usarla y para que o en que me hará más fácil su uso.
- No explicaba las funciones el profesor.
- Pues como ya dije los profesores no enseñan realmente y sus códigos tenían errores.
- Falta de equipo para realizar prácticas, así como un profesor que en verdad enseñara.
- La forma de enseñanza del profesor.
- El no tener un buen profesor y solo confundía en vez de ayudar.
- Falta de preparación de la clase.
- Que siempre los maestros dan el problema, pero casi nunca dan ejemplos del programa (semejantes) es decir siempre es teoría.
- Más bien no hay orientación por parte de los profesores.
- La poca claridad de los temas, la poca coherencia del profesor para explicarse.
- Que no se veían temas con un profesor que eran necesarios para el siguiente curso.
- No tuve una persona que me dijera cómo se hacían las cosas.
- Más que aprender la sintaxis de cada lenguaje y tratar de entender lo que algunos maestros querían que hiciera y lo que intentaba enseñar.
- Tuve que aprender fuera de la facultad.
- La resolución de un problema real.
- Que no entendía perfectamente, falta de información, no hay libros suficientes en la biblioteca, los profesores no son abiertos a explicaciones, aclaraciones o dudas.
- El acceso a programas o compiladores y que el maestro no podía proporcionarlos.
- La falta de uso del laboratorio en algunas ocasiones, también porque es un lugar muy reducido.
- A la falta de una computadora para practicar.
- Falta de explicación de algunos temas, principalmente en POO
- Que el programar es tanto una ciencia como un arte y para poder hacerlo adecuadamente se deben de pasar muchas horas practicando, repasando e

investigando y la mayor parte de veces solo hacía lo necesario para que el programa (en cuestión) funcionara.

¿Cómo te gustaría que te enseñaran programación?

- Explicando la teoría desde lo más básico y después analizando línea por línea que es lo que está haciendo el programa, solo aprenderíamos programación de verdad.
- Desde el principio y con detalles claros y específicos de cada elemento que se utiliza en el programa.
- De una forma clara y concisa con profesores y que den confianza.
- Iniciar desde lo básico y haciendo programas en las computadoras y no en pizarrón.
- Con una persona que me transmita los conceptos claros, su experiencia y material de apoyo (p. ej. Programas, código fuente, etc.).
- De una manera didáctica en la que se explicara a fondo la función de cada parte del código.
- Mediante teoría, pero mayoritariamente practicando en conjunto maestro-alumno.
- Con sus definiciones, pero más bien que vallan armando el código y nos vallan diciendo porqué.
- Que en el salón se programe y se haga más pruebas de escritorio.
- Paso a paso y aunque sea ridículo quisiera que me enseñaran como niño chiquito para poder entenderle.
- Con peras y manzanas, con buenos ejemplos.
- Explicando básicamente con manzanitas.
- Desde el principio y dejándolo CLARO cada uso y el porqué de la sentencia para poder aplicarlo a la programación.
- Empezar desde los conceptos básicos y tener una computadora donde practicar y que sea frecuente.
- Pues que explicara o mínimo dieran las bases de los diferentes lenguajes de programación.

- Con un profesor que sepa perfectamente el tema y que sepa explicar con claridad. Cursos prácticos.
- De acuerdo al temario y en un centro de cómputo.
- Más didácticamente y un poco más estricto.
- Pues que fuera más didáctico y que enseñaran cada una de las funciones a utilizar.
- Que pusieran y explicaran más a fondo con más ejemplos.
- En un laboratorio de cómputo para que la clase sea teórica-práctica.
- "La práctica hace al maestro", que los profes te den la teoría, pero bien definida y nosotros llevarla a la práctica.
- Mediante prácticas en laboratorios de computación.
- Con clases puramente prácticas.
- Sería muy bueno que todas las clases de programación se realizaran en salas de cómputo, con un proyector para que en la práctica uno aprenda mejor.
- De manera práctica durante todo el curso, con ejercicios enfocados a problemas comunes dentro de la carrera.
- En un salón de cómputo, con realización de ejercicios básicos para después usar códigos más complejos.
- Pues en aulas, y con más claridad en la forma de hacer el algoritmo.
- En el centro de cómputo para llevar todo a la práctica y poder despejar las dudas y que los profesores se interesen porque realmente aprendamos.
- Pues en un laboratorio adecuo al número de alumnos.
- Enfrente de una computadora y definiendo línea por línea.
- Con una computadora por alumno que el profesor vaya haciendo el programa junto con los alumnos y si hay errores se detenga para ayudar a solucionarlos.
- En una sala de computo bien equipada con el cupo de la sala y con un profesor que sea interesado, que tenga conocimientos y que explique bien.
- Poniéndonos frente a un ordenar para ir probando las cosas...
- Que dejara impartirse en un salón y se ocupara un laboratorio de cómputo donde pudiéramos practicar y resolver dudas.

- En un aula de computo con problemas reales, menos teoría.
- Que los grupos fueran más reducidos o con 2 profesores.

1.11 Perspectiva del Profesor del Problema.

De la misma manera diseñamos una entrevista para los profesores (Anexo 2) con la finalidad de conocer su opinión sobre el problema de enseñanza-aprendizaje de la programación. Para dicho propósito se entrevistó a los profesores que habían impartido la clase de "Programación Orientada a Objetos" de la generación de alumnos del presente estudio. Desafortunadamente de los tres profesores que impartieron la materia de dicha generación, sólo se logró aplicar la entrevista a dos profesores. A continuación, se muestran sus respuestas.

Profesor 1

1. Al inicio del curso ¿usted realiza un muestreo para ver si sus alumnos vienen con suficientes bases?, es decir, si los alumnos manejan las bases de la Programación Estructurada.

   Si, les dejo programas (ejercicios) para ver el nivel de conocimientos de los alumnos.
2. ¿Considera que los alumnos tienen las suficientes bases de programación para cursar la materia de Programación Orientada a Objetos?

   Si, tienen las nociones aproximadamente el 80% de los alumnos pero no las saben aplicar.
3. En caso de que un número considerable de alumnos no cuente con los conocimientos suficientes, usted ¿qué hace?

   Al dejar los programas los alumnos van recordando los conocimientos previos.
4. ¿Tiene identificados los temas que se les dificultan a sus alumnos?, ¿Cuáles son? y ¿Por qué cree que se les dificulten?

   Los alumnos no entienden el problema, tampoco saben analizar.
5. ¿Considera suficiente el tiempo del semestre para cubrir el temario?

   Si.

6. ¿Con cuántos alumnos le sería más sencillo enseñar?

   30.

7. ¿Cuándo usted enseña un nuevo tema explica el objetivo de este?

   Si, ya sea primero explicando y luego reafirmando con ejercicios, o dejando un ejercicio relacionado con el tema y lo voy explicando.

8. ¿Les enseñaba como analizar un problema de tal forma que ellos puedan implementar la solución mediante la Programación Orientada a Objetos?

   Si, se les ofrece situaciones y comparaciones de situaciones.

9. ¿Cuándo daba un código para explicar un tema sus códigos eran sencillos o elaborados?

   De las dos formas, les daba un código sencillo y posteriormente les daba uno más elaborados, pero con errores.

10. ¿Cuándo les pone un código a sus alumnos, cómo da la explicación de este, de forma general o explica detalladamente línea por línea?

    Primero se da una explicación general y después dejo que los alumnos lo analicen línea por línea.

11. Para un alumno es difícil ya sea leer o copiar el código del pizarrón, ¿usted de alguna forma le facilitaba al alumno este?

    Doy copias de los códigos preparados para la clase, aunque los códigos improvisados durante la clase tienen que ser copiados del pizarrón.

12. ¿Usted que prefería, dejar muchos ejercicios sencillos o dejar pocos, pero más elaborados?

    Varios sencillos para crear uno complejo.

13. ¿Lleva a sus alumnos al CC? ¿Con qué frecuencia?, en caso de que no por qué no?

    No.

14. ¿Considera que los alumnos aprenderían mejor la materia si esta se llevara a cabo en un CC?

    Sería igual.

15. ¿Ha trabajado en la industria programando con java?

No.

16. A su criterio ¿cuál es la problemática de la enseñanza-aprendizaje de la programación?

    Que los alumnos no saben analizar, falta de interés de los profesores, están acostumbrados a la enseñanza tradicional por medio de pizarrón.

17. A su criterio ¿cuál sería una posible solución a la problemática de la enseñanza-aprendizaje de la programación?

    - *Que los alumnos aprendan a analizar.*
    - *Interés de los alumnos para entender los problemas.*
    - *Encontrar ejemplos.*
    - *Interés del alumno y profesor.*
    - *No transmitir información incorrecta.*
    - *Preparar mejor a los profesores.*
    - *Actualizar a los profesores para dar clase y encontrar nuevos elementos para enseñar.*

Profesor 2

1. Al inicio del curso ¿usted realiza un muestreo para ver si sus alumnos vienen con suficientes bases?, es decir, si los alumnos manejan las bases de la Programación Estructurada.

   Si, preguntas básicas, antecedentes, como que es un bit, funciones de computadora, etc. y conocimientos previos, tipos de datos, etc.

2. ¿Considera que los alumnos tienen las suficientes bases de programación para cursar la materia de Programación Orientada a Objetos?

   No, hay que regresar.

3. En caso de que un número considerable de alumnos no cuente con los conocimientos suficientes, usted ¿qué hace?

   Bajar el nivel del curso.

4. ¿Tiene identificados los temas que se les dificultan a sus alumnos?, ¿Cuáles son? y ¿Por qué cree que se les dificulten?

   Si, son:

   - *Visión, no entienden que son los objetos, el enfoque, el cambio de paradigma, y el de filosofía.*
   - *Los punteros, al no tener las bases desconocen las funciones de memoria.*
   - *Instrucciones, ya que no saben el funcionamiento del procesador.*

5. ¿Considera suficiente el tiempo del semestre para cubrir el temario?

   No.

6. ¿Con cuántos alumnos le sería más sencillo enseñar?

   Un ideal sería de 30.

7. ¿Cuándo usted enseña un nuevo tema explica el objetivo de este?

   Si.

8. ¿Les enseñaba como analizar un problema de tal forma que ellos puedan implementar la solución mediante la Programación Orientada a Objetos?

   Si.

9. ¿Cuándo daba un código para explicar un tema sus códigos eran sencillos o elaborados?

   Sencillos.

10. ¿Cuándo les pone un código a sus alumnos, cómo da la explicación de este, de forma general o explica detalladamente línea por línea?

    Primero general, cuál es el problema, cómo hacer la estructura, el diseño, y después línea por línea.

11. Para un alumno es difícil ya sea leer o copiar el código del pizarrón, ¿usted de alguna forma le facilitaba al alumno este?

    Dándoles material en sitio en el centro de cómputo.

12. ¿Usted que prefería, dejar muchos ejercicios sencillos o dejar pocos, pero más elaborados?

    Dependiendo del perfil de los alumnos, si es bajo se dejan pocos.

13. ¿Lleva a sus alumnos al CC? ¿Con qué frecuencia?, en caso de que no por qué no?

    Sí, la mitad del semestre.

14. ¿Considera que los alumnos aprenderían mejor la materia si esta se llevara a cabo en un CC?

    Sí.

15. ¿Ha trabajado en la industria programando con java?

    Sí.

16. A su criterio ¿cuál es la problemática de la enseñanza-aprendizaje de la programación?

    No hay perfil en los alumnos.

17. A su criterio ¿cuál sería una posible solución a la problemática de la enseñanza-aprendizaje de la programación?

    - *Filtros de los alumnos de actitud y aptitud.*
    - *Un plan de estudios diferente y capacitación de los profesores.*
    - *Ingeniería en computación es una carrera sin objetivos.*

Habiendo expuesto en el presente capítulo la problemática de la enseñanza-aprendizaje en la FES Aragón ahora es necesario conocer los conceptos pedagógicos con la finalidad de comprender lo que esta área humanística aporta, es por ello que en el siguiente capítulo se realiza una síntesis de estos con la finalidad de fundamentar la metodología planteada en el capítulo III.

CAPÍTULO II: MARCO TEÓRICO

Al analizar la razón del fracaso escolar, nos encontramos con varios elementos que van desde problemas en la enseñanza, como la masificación de las aulas, profesores no capacitados pedagógicamente, etc. hasta problemas en el aprendizaje como, alumnos sin el perfil, falta de motivación por aprender, etc.

Para poder diseñar un método que nos ayude a mejorar el proceso de enseñanza de las bases de programación en los alumnos de la FES Aragón, es necesario partir de los conceptos básicos de la enseñanza y aprendizaje.

2.1 Enseñanza

La enseñanza es el proceso mediante el cual se transmite un conjunto de conocimientos, técnicas, principios, normas, ideas y/o habilidades generales o específicas sobre una materia. Basado en un método, elaborado a través de una serie de reglas y preceptos. Fuente: Elaboración propia (2017).

Al ser el hombre un ser sociable la enseñanza resulta ser un efecto de la condición humana, siendo un medio por el cual la humanidad hace su existencia perdurable. De esta forma, la enseñanza se convierte tanto como en un deber como en un derecho, buscando siempre los medios que faciliten su adquisición.

Para que haya una enseñanza deben de existir tres elementos: el profesor, docente o maestro; el alumno o estudiante; y el objeto de conocimiento.

En la percepción clásica de la enseñanza el profesor es la fuente del conocimiento y el alumno un receptor del mismo. No obstante, para las corrientes actuales, como la cognitiva, el profesor es un facilitador del conocimiento que al interactuar con el alumno actúa como nexo entre el conocimiento y el alumno. Fomentando el compromiso del alumno con su propio aprendizaje y tomando la iniciativa por la búsqueda del saber.

Clasificar los métodos de enseñanza generalmente es un proceso subjetivo, ya que nos basamos en experiencias e investigaciones propias. Por razones de estandarización hemos decidido utilizar en esta tesis la clasificación tradicional de dichos métodos, que está basada por la utilización del lenguaje y la terminología.

Sin embargo, nos hemos permitido hacer algunos pequeños ajustes adaptándolos a los avances en el conocimiento del aprendizaje en relación con las nuevas tecnologías de la educación.

2.1.1 Métodos de Enseñanza en cuanto a la forma de razonamiento

Se clasifica en tres métodos, deductivo, inductivo y comparativo o analógico.

El método deductivo se refiere a cuando el tema por enseñar va de lo general a lo particular. El profesor primero presenta los conceptos, definiciones o principios, de los cuales se analizan los casos particulares basándose en los conceptos ya presentados. Tradicionalmente este método se utiliza más en la enseñanza ya que evita trabajo extra y ahorra tiempo. Sin embargo, este método es recomendado cuando los conceptos, definiciones o principios ya están bien entendidos por los alumnos, ya que a partir de ellos se "deducen" los casos particulares.

El método inductivo se emplea cuando el tema a enseñar va de lo particular a lo general. El profesor presenta primero un caso particular, incitando a que se llegue al principio general que lo rige. Por excelencia el método más utilizado para la mayoría de los descubrimientos científicos, ya que está basado en la observación.

El método comparativo o analógico se usa cuando el tema a enseñar va de lo particular a lo particular. Es decir, el profesor presenta al menos dos casos particulares, he incita a los alumnos a comparar las similitudes de estos. Siendo este método el más utilizado para la educación de los más pequeños.

2.1.2 Métodos de Enseñanza en cuanto a la Organización de la Materia

Estos métodos se clasifican en dos, basado en la lógica de la tradición o de la disciplina científica, y basado en la psicología del alumno.

En el primero los datos se presentan originados en una estructuración de hechos, basados en el razonamiento adulto, en orden de lo menos a lo más complejos o del origen a lo actual. Generalmente esta estructura es la utilizada en los libros de texto.

En el segundo los datos se presentan basados en los intereses y experiencias del alumno, va de lo conocido a lo que desconoce. Este método aboga más por motivar a la intuición del alumno que a la memorización.

Cabe destacar que algunos profesores se oponen a cambiar el "orden lógico" de enseñar, ya que requiere más esfuerzo y tiempo. Sin embargo, Bruner[1] maneja el orden de presentar el contenido como un elemento didáctico que ayuda a generar una motivación por el aprendizaje por parte de los alumnos.

2.1.3 Métodos de Enseñanza en cuanto a su Relación con la Realidad

Estos métodos se dividen en dos, simbólico o verbalístico, e intuitivo.

En el método simbólico o verbalístico el tema se presenta solo por lenguaje oral o escrito. Es un método utilizado por muchos profesores, aunque es muy criticado cuando se utiliza como único método ya que dificulta la motivación del alumno y provoca que pierda el interés, y el profesor tiende a olvidar formas diferentes de la presentación de contenidos.

Mientras que en el método inductivo el tema se presenta tratando de acercarse lo más posible a la realidad inmediata del alumno. Esto fomenta el interés del alumno, ya que se le enseña basándose en su propia experiencia.

[1] Psicólogo y pedagogo estadounidense, fundó el Center for Cognitive Studies, considerado el primer centro de psicología cognitiva

2.1.4 Métodos de Enseñanza en cuanto a las Actividades Externas del Alumno

Estos métodos son dos, pasivo y activo.

El método pasivo sucede cuando el profesor fomenta que los alumnos no participen en la clase, permaneciendo de forma neutral, esto se consigue por medio de exposiciones del maestro, cuestionarios, dictados, etc.

El método activo es cuando se cuenta con la participación dentro de la clase del alumno, así el método y las actividades de este fomentan la motivación del alumno. Mientras el profesor se convierta en un orientador del aprendizaje, todas las técnicas de enseñanza se pueden volver activas.

2.1.5 Métodos de Enseñanza en cuanto a la Sistematización de Conocimientos

Estos son dos, globalizado y especializado.

En el método globalizado, la clase se desarrolla basándose en un grupo de asignaturas de acuerdo a las necesidades del tema principal. Se vuelve interdisciplinario cuando son varios los profesores que apoyan o rotan en su especialidad.

Al contrario, en el método especializado el tema se trata independientemente, es decir, sin vincularlo con otras asignaturas.

2.1.6 Métodos de Enseñanza en cuanto a la Aceptación de lo Enseñado

Se clasifican en dos, dogmático y eucarístico o de descubrimiento.

En el método dogmático el profesor enseña sin permitir la discusión con el alumno, en la suposición de que lo que enseña es la verdad.

En el método eucarístico o de descubrimiento el profesor presenta los elementos del aprendizaje para que el alumno descubra antes de aceptarlo como verdad.

2.2 Aprendizaje

El aprendizaje es el proceso en el cual se adquieren o modifican conocimientos, valores, habilidades y actitudes por medio del análisis y comprensión de información externa, siendo esta aplicada a la propia existencia. Fuente: Elaboración propia (2017).

"El aprendizaje nos obliga a reflejar los nuevos conocimientos, al cambiar nuestro comportamiento en las experiencias presentes y futuras. Se necesitan tres actos para aprender: observar, estudiar y practicar." Kahneman, 2014

Vale la pena señalar que todos los seres humanos, salvo aquellos con alguna discapacidad, al momento de nacer poseemos el mismo intelecto y de acuerdo al proceso individual de aprendizaje, se utilizará en menor o mayor medida esta capacidad intelectual.

El proceso más simple y básico del aprendizaje es la imitación, ya que es la repetición de un proceso observado, es así como los niños aprenden para subsistir y desarrollarse en una comunidad.

La capacidad de aprender no es exclusiva del ser humano, aunque gracias al desarrollo de esta hemos logrado alcanzar independencia del entorno ecológico, al punto de cambiarlo para adaptarlo a nuestras necesidades.

Algunos elementos que facilitan o complican la tarea de aprender son:

- La motivación: esta se puede ver disminuida o aumentada de acuerdo a elementos tanto internos como externos del individuo, pero se puede definir como el interés que muestra el alumno por aprender.
- La maduración psicológica: Esta es importante ya que dependiendo de la edad del alumno puede aprender de una forma más fácil y así saber que temas tratar con él.
- Material: esto puede ayudar al aprendizaje y cuando falta puede atrasar al alumno.

- La actitud dinámica y activa: siempre hay que tomar en cuenta que mientras el alumno tenga el interés por aprender es mucho más sencillo enseñar con dinámicas, es decir, juegos, competencias, etc.
- Estado de fatiga: es importante que el alumno esté en las condiciones óptimas para aprender, para que pueda poner atención en la clase debe estar descansado y haber dormido bien.
- Capacidad intelectual: El desarrollo de esta varía dependiendo de cada persona, así que se tiene que explicar paso a paso el tema para que todos los alumnos entiendan.
- Distribución del tiempo para aprender: es importante tener en consideración la distribución del tiempo ya que esto nos ayuda a que el alumno tenga una mente dispuesta a aprender.

2.2.1 Tipos de Aprendizaje

El ser humano es capaz de múltiples tipos de aprendizaje, estos dependiendo de las capacidades y experiencias adquiridas de cada uno, por esta razón la clasificación de los tipos de aprendizaje, al igual que los métodos de enseñanza, es un proceso subjetivo.

Por razones de estandarización hemos decidido utilizar en esta tesis la clasificación pedagógica de dichos tipos de aprendizaje.

2.2.1.1 Aprendizaje Memorístico o Repetitivo

Se origina cuando el alumno memoriza la nueva información sin comprender, sin relacionarla con información anterior y de manera impersonal, esto produce que la nueva información sea carente de significado, de tal modo si el alumno olvida una sola palabra, al no ser capaz de explicar con las propias, no puede seguir recordando la información, generalmente escrita.

Aunque en todos los tipos de aprendizajes se requiere de la memoria para almacenar la información, al ser este un tipo de aprendizaje basado solamente en la repetición mecánica, la información adquirida se incorpora solo a la memoria de corto

plazo, ya que su falta de significado no le permite tener relación con ninguna otra información que le pueda ayudar a recordar, por lo que después de un periodo de tiempo se olvidará.

Algunos ejemplos de esto es la forma en la que se aprenden las tablas de multiplicar o cuando se estudia para un examen por medio de una guía.

2.2.1.2 Aprendizaje Receptivo

En este aprendizaje al alumno se le presenta el tema en su forma final, necesita comprender el contenido para poder reproducirlo, pero no es necesario que comprenda el significado del contenido. Al igual que el aprendizaje memorístico o repetitivo el alumno permanece como un ser pasivo ya que solo recibe la información que debe reproducir en cierto momento.

Un ejemplo de este aprendizaje, es cuando un alumno aprende una fórmula matemática y sabe cómo aplicarla, pero no sabe de dónde viene.

2.2.1.3 Aprendizaje por Descubrimiento

En este aprendizaje se fomenta que el alumno actúe de manera activa adquiriendo nuevos conocimientos por sí mismo, por lo que el profesor no presenta la información en su forma final, sino que promueve que el alumno mediante la observación, la comparación, el análisis de semejanzas y diferencias y la experimentación entre lo previamente aprendido y el mundo que lo rodea, genere esta nueva información. Así mismo este método estimula la autoestima y seguridad del alumno.

Un ejemplo de esto es cuando se enseñan los colores primarios, y después de jugar con las combinaciones pueden descubrir los colores secundarios.

2.2.1.4 Aprendizaje por Observación

Como su nombre lo dice este aprendizaje se adquiere por observación e imitación, es la forma más fácil en la que los seres humanos aprendemos. Para que se produzca un

aprendizaje depende de factores cognitivos, la atención, la memoria, la posibilidad de reproducir la conducta y la motivación, así mismo este tipo de aprendizaje acelera estos mecanismos cognitivos y modelos de interacción social, sólo si el modelo resulta atractivo, funcional o presenta incentivos.

2.2.1.5 Aprendizaje Significativo

En este aprendizaje se busca que el alumno actué de forma activa en su propio aprendizaje, siendo el profesor quien presenta el problema y ayuda a los alumnos a interrogar e interrogarse. Los alumnos aprenden mejor cuando el problema lo perciben como relacionado a ellos mismos o a una situación que consideran importante, cuando el problema lo consideran ajeno o sin importancia, el aprendizaje se ubica en la memoria a corto plazo y eventualmente lo olvidarán.

En este aprendizaje también influye mucho que el proceso educativo se desarrolle en un lugar adecuado, con material didáctico y métodos de enseñanza participativos. Así mismo el profesor debe estar consciente que este tipo de aprendizaje involucra aspectos afectivos, por lo que se la enseñanza se debe dar en un ambiente con relaciones interpersonales basadas en la confianza, la tolerancia y el respeto. Para generar un compromiso del alumno con su propio proceso de aprendizaje.

2.2.1.6 Aprendizaje Creativo

En este aprendizaje se busca que por medio de la creatividad el alumno mejore su capacidad para resolver problemas, por medio del mismo proceso creativo, de las relaciones sociales y de la empatía.

Para poder formar alumnos originales, flexibles, con visión futura, iniciativa y confianza es importante desarrollar por medio del proceso educativo la creatividad, esto también fortalece las herramientas para la innovación.

La creatividad favorece las capacidades del alumno alcanzando una mejor utilización de recursos individuales y grupales dentro del proceso enseñanza-aprendizaje. Para que

esto se complete es importante que exista una atmósfera creativa que favorezca el pensamiento reflexivo del alumno.

Por naturaleza el ser humano es creativo. Esta creatividad es el punto base para la adaptación al cambio y nos lleva al progreso humano. Para que una persona pueda manifestar su propia creatividad es necesario tener la capacidad de motivación y la comunicación.

Para poder educar creativamente el profesor debe considerar los siguientes puntos:

- Los alumnos deben de aprender a tolerar la ambigüedad, esto se puede lograr estimulándolos desde el principio de la clase a reflexionar la situación de la problemática.
- El profesor debe crear un ambiente de incertidumbre dentro de la clase que provoque que el alumno explore fuera del conocimiento adquirido para complementarlo con sus propias experiencias.
- El profesor debe enseñar a los alumnos que los obstáculos que puedan encontrar al tratar de solucionar el problema lo deben convertir en oportunidades, y no en amenazas.
- La confianza en los alumnos y en sus propias convicciones es muy importante por lo que el profesor debe dar indicadores a los alumnos de que están haciendo un buen trabajo, que no siempre son buenas notas y pasar el curso.
- Durante la clase el profesor debe comenzar a analizar qué recursos pedagógicos utilizará para la siguiente clase, basado en el desempeño de los alumnos.
- El profesor debe creer en el potencial de los alumnos, el que no puedan resolver un problema el día de hoy no quiere decir que sin el apoyo sean incapaces de resolverlo mañana.
- Enseñar a los alumnos a reciclar los errores como fuente de aprendizaje, para que el alumno pierda el temor a cometerlos.

- Evitar que el alumno no exprese sus ideas por miedo a romper con la imagen de buen estudiante, contradecir el método o al maestro, etc.
- Responsabilizar a los alumnos de su propio aprendizaje, a medida que el profesor motiva a los alumnos con las estrategias de enseñanza.
- Enseñar al alumno a ser imaginativo y a que cuestione las "verdades" que dice el maestro y los libros.

Aunque este método puede sonar tardado para el profesor (principalmente en las primeras sesiones) es más útil que el alumno obtenga pocos conocimientos a fondo, con los que pueda discutir el significado de los mismos, envés de que obtenga una gran cantidad de conocimientos pero que, al ser adquiridos de manera superficial, estos no tengan una duración significativa en la memoria.

2.2.1.7 Aprendizaje Auditivo

Aunque la mayoría de la gente es principalmente visual y la forma en que se relacionan con el mundo alrededor y la forma en la que encuentran y absorben conocimientos, es cierto que existe un pequeño porcentaje de personas que el aprendizaje auditivo es su método primario.

Las personas auditivas mejoran su aprendizaje cuando reciben explicaciones orales y a su vez pueden debatir esa nueva explicación con alguien más. Aunque el sistema auditivo no permite relacionar o elaborar conceptos abstractos con la misma facilidad y rapidez que el sistema visual, el sistema auditivo es fundamental para el aprendizaje de los idiomas y de la música.

Los alumnos con aprendizaje auditivo se pueden reconocer, por ser más lentos leyendo, son descuidados con sus apuntes, o no toman ninguno y son buenos para las respuestas en voz alta, como los exámenes orales.

2.2.1.8 Aprendizaje Visual

Este aprendizaje se basa en el uso de imágenes o material visual, este ayuda a los alumnos tanto para representar información como para trabajar con ideas y conceptos que los ayudan a adquirir nuevo conocimiento. Este aprendizaje enseña a los alumnos a:

- Filtrar el pensamiento: los estudiantes puedes ver la conexión de las ideas y organizar o agrupar la información. Los nuevos conceptos se vuelven más profundos y provoca que sean fácilmente comprendidos.
- Refuerza la comprensión: El reproducir lo que han aprendido en sus propias palabras, ayuda a los alumnos a interiorizar la nueva información y les da posesión de sus propias ideas.
- Integrar un nuevo conocimiento: Mediante la revisión de material visual de información previa los alumnos son capaces de integrar la nueva información ajustándola con la previa.
- Identificar conceptos erróneos: Así como un mapa conceptual muestra lo que los alumnos han aprendido, los enlaces mal dirigidos muestran los que aún les falta asimilar.

2.2.1.9 Aprendizaje Kinestésico

En este aprendizaje el alumno procesa la información asociándola con movimiento o sensaciones del cuerpo, por lo que prefiere la experiencia física más que leer o escuchar las instrucciones.

Este aprendizaje se utiliza de forma natural al aprender un deporte, pero se puede ocupar para muchas otras actividades, por ejemplo, las personas que saben escribir a máquina no necesitan ver el teclado, porque sus dedos saben lo que tienen que hacer, aún así, puede ser que al preguntarles la localización de una letra la desconozcan.

Ya que el aprendizaje kinestésico está basado en experimentar este es mucho más lento que el aprendizaje auditivo o visual.

2.2.2 Teorías del Aprendizaje

Las personas mayoritariamente aprendemos a través de la experiencia, sin necesidad de preocuparnos por el proceso de aprendizaje. Así los padres les enseñan a los hijos y los artesanos a sus aprendices. Esta enseñanza se realiza de forma empírica, es decir, diciendo y mostrando cómo hacer las cosas.

Cuando se crearon las escuelas se formalizó el aprendizaje y así la enseñanza dejó de ser una tarea simple, tanto por los contenidos (lectura, escritura, matemáticas, geografía, historia, etc.) como por el número de estudiantes.

Los profesores rápidamente se dieron cuenta que el aprendizaje escolar resultaba ineficiente y no se obtenían resultados apreciables, provocando que a los estudiantes no les agradara la escuela y resistiéndose al aprendizaje.

Actualmente la idea de aprender por medio de información que recibimos del exterior, la procesamos e interpretamos obteniendo nuevos conocimientos nos puede parecer lógica y común, pero esta idea no siempre ha existido, surgiendo así las escuelas psicológicas dando pie a múltiples teorías del aprendizaje, siendo estas la forma que el profesor elabora un plan de estudios hace una selección de materias y escoge las técnicas con las que les enseñará a los alumnos.

Por consiguiente, una teoría de aprendizaje es una guía para el profesor del proceso que llevará para enseñar a los alumnos, dándole a la enseñanza una finalidad y un plan a largo plazo.

Muchos profesores emplean un conjunto de métodos de enseñanza, pero sin una orientación teórica, desgraciadamente esta forma desorganizada de enseñanza causa muchas de las críticas de la educación, utilizando las teorías de aprendizaje el profesor tiene una mayor posibilidad de causar resultados más eficientes dentro del aula.

Vamos a considerar las principales teorías de aprendizaje, estas son:

- Teoría Conductista.
- Teoría Cognitiva.
- Teoría Sociocultural.
- Teoría Constructivista.

2.2.2.1 Teoría Conductista

Aunque la teoría conductista actualmente no encaja dentro de los nuevos paradigmas educativos, por considerar el aprendizaje como algo deshumano, mecánico y haciendo una simplificación excesiva de los conceptos, esta es la de mayor tradición y la que se ha mantenido durante más años. Incluso muchos programas actuales se basan en la descomposición de información para crear unidades, en el diseño de actividades que requieren una respuesta, y en la planificación de refuerzos.

Dentro de la teoría conductual el aprendizaje se define como la adquisición de conocimientos y/o habilidades a través de la experiencia provocando un cambio permanente en el comportamiento, estos cambios no incluyen a los cambios provocados por maduración.

A continuación, mencionaremos a los principales representantes de esta teoría y hablaremos de su contribución.

Iván Petróvich Pávlov (1849-1936)

Imagen 2–1. Iván Petróvich Pávlov.

A través de su trabajo con perros desarrolló la teoría del reflejo condicionado o condicionamiento clásico, que consiste en dar un estímulo natural a un individuo que lo hará realizar una acción natural y asociarla a un estímulo externo, de esta forma el simple estímulo externo generará la acción natural. Está teoría la desarrolló haciendo sonar una campana (estímulo externo) justo antes de dar alimento (estimulo natural) a un perro provocando que comenzara a salivar (acción natural), así el perro después de un tiempo

comenzaba a salivar en cuanto oía el sonido de la campana sin necesidad de presentar el alimento.

Este condicionamiento se enfoca en el aprendizaje de una respuesta involuntaria a algún estimulo que antes no tenía ningún efecto. Un ejemplo claro de cómo el condicionamiento clásico es utilizado en el método conductista son los castigos en clase.

Edward Lee Thorndike (1874 – 1949)

Imagen 2–2. Edward Lee Thorndike.

Mediante su trabajo con perros, gatos y pollos desarrolló la teoría de estímulo-respuesta, la cual nos dice que un individuo aprende como el resultado de asociar estímulos y respuestas, estas asociaciones se fortalecen o debilitan dependiendo del tipo de estimulo y la frecuencia del mismo. Así se aprende por prueba y error, incitando a dar las respuestas correctas dando estímulos positivos (gratificaciones). Un ejemplo claro de cómo las conexiones estimulo-respuesta es utilizado es el método conductista son las recompensas que el profesor puede dar a los alumnos por un buen desempeño.

John Broadus Watson (1878 – 1958)

Imagen 2–3. John Broadus Watson.

Por medio de su teoría de estímulo-respuesta desarrolló la teoría conductista, siendo así conocido como el padre del Conductismo.

Su fundamento teórico está basado en la conducta como la única medida posible para el estudio científico, sin considerar los procesos mentales del individuo al no ser observables.

En su teoría (condicionamiento clásico) señala que si a un estímulo neutro que no genera una respuesta se le une otro

estimulo neutro que genera una respuesta, el primer estimulo neutro comienza a generar una respuesta condicionada en el individuo.

Esto lo comprobó con un bebe (pequeño Albert), al presentarle un ratón (estimulo neutro) este no le tenía miedo, pero al asociar al ratón con un sonido fuerte (estimulo neutro) que le genera miedo, eventualmente al presentarle un ratón o algo peludo le generara miedo (estimulo condicionado).

Burrhus Frederic Skinner (1904 – 1990)

Imagen 2–4. Burrhus Frederic Skinner.

Influido por las teorías de Pavlov y Watson, Skinner desarrolló el sistema de condicionamiento operante a partir de los esquemas de refuerzo experimentando con animales en "la caja de Skinner", muy parecido a la propuesta de Watson, pero incorporando la observación de la psicología interna, como los sentimientos.

El condicionamiento operante consiste en dar un refuerzo al individuo cada vez que lleve a cabo un comportamiento deseado aumentando la probabilidad de repetir dicho comportamiento en el futuro, el único problema es que cuando se deja de dar el refuerzo el comportamiento aprendido se pierde con el tiempo, para solucionar este problema creo los "esquemas de refuerzo" dando el refuerzo en intervalo de tiempos fijos y variables.

Albert Bandura (1925 -)

Imagen 2–5. Albert Bandura.

El centro de su estudio se enfocó en los procesos de aprendizaje que hay en la interacción del sujeto y su entorno, creando así la Teoría del Aprendizaje Social. Los psicólogos conductistas explicaban la adquisición de nuevas conductas y conocimiento con una aproximación gradual por medio de refuerzos negativos o positivos en un ambiente controlado, mientras que Bandura explicó que un individuo puede aprender

cosas nuevas y desarrollar nuevas conductas mediante la observación de los individuos que lo rodean sin la necesidad de un refuerzo directo ya sea positivo o negativo.

Podemos resumir la Teoría Conductista de la siguiente forma:

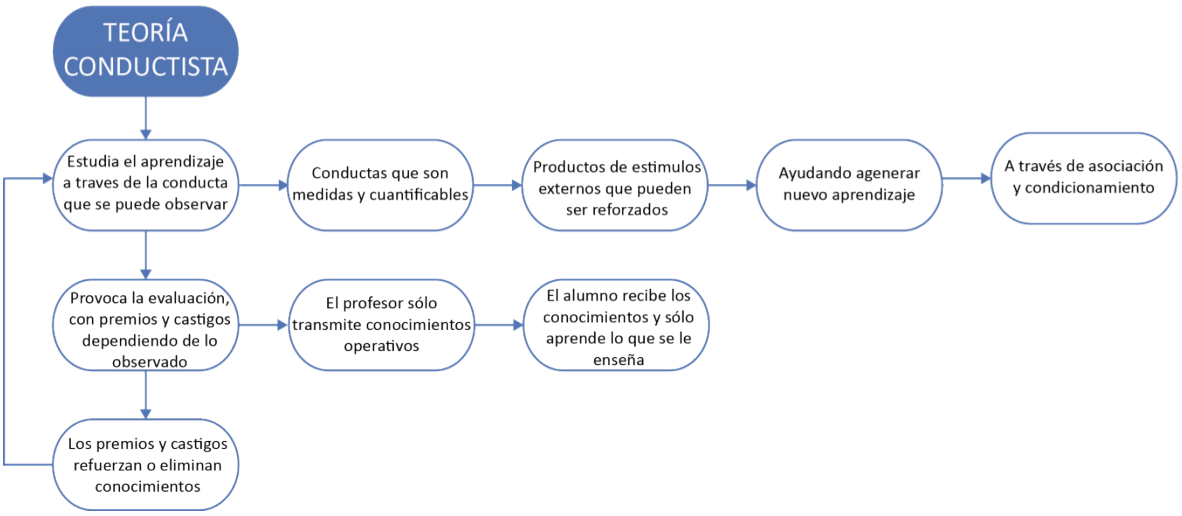


Imagen 2–6. Teoría Conductista. Fuente: Elaboración propia (2017).

2.2.2.2 Teoría Cognitiva

Tiene mucha influencia de la Teoría Conductista, utilizando varios de sus conceptos pero reformulándolos, dentro sus principales intereses se encuentra la memoria y la percepción. Esta teoría está basada en la conducta observable que nos permiten entender los procesos mentales del alumno donde también se involucran las relaciones que se establecen con los demás individuos a su alrededor.

Aunque se considera el refuerzo como una parte importante del aprendizaje, este se acentúa como elemento retro alimentador funcionando como motivación para la corrección de respuestas.

Los pasos que las teorías cognitivas manejan para el aprendizaje de los alumnos de nueva información son:

- Que la información sea correctamente recibida
- Organizada y almacenada
- Vinculada a información previa

A continuación, mencionaremos a los principales representantes de esta teoría y hablaremos su de contribución.

Jean Piaget (1896 – 1980)

Imagen 2–7. Jean Piaget.

Piaget definió a la inteligencia como el proceso de adaptación de un conjunto de operaciones lógicas con tendencia a organizar el conocimiento en estructuras y esquemas considerando los cambios del entorno.

Así mismo la inteligencia según Piaget se divide en cuatro etapas, las cuales son progresivas y cada una se caracteriza por una estructura de conocimiento. Estas etapas son:

- Motora Sensorial (0 – 2 años): A partir de la experiencia existe un control motor y aprendizaje físico sobre los objetos.
- Pre Operacional (2 – 7 años): Se desarrollan las habilidades de comunicación y el lenguaje.
- Concreta Operacional (7 – 12 años): Desarrollo, entendimiento y uso de conceptos abstractos.
- Operacional formal (12 – 15 años): Desarrollo sistemático del razonamiento lógico.

Jerome Seymour Brunner (1915 – 2016)

Imagen 2–8. Jerome Seymour Brunner.

Para Jerome Bruner el elemento principal durante el proceso de aprendizaje es la participación activa del alumno, de la forma que procesa y le da sentido a la nueva información.

De acuerdo a la Teoría Cognitiva de Bruner, el alumno al recibir información externa la clasifica en conjuntos de objetos semejantes con el fin de asociarla a eventos de su realidad, al proceso se le dio el nombre de categorización. Esta categorización le permite al alumno la formación de nuevos conceptos y la habilidad de hacer predicciones para poder tomar decisiones. Esta categorización no permanecerá siempre estable y cerrada, al contrario, se irá modificando y aumentando de acuerdo a las experiencias que el alumno acumule, mediante tres etapas que pueden ser utilizadas simultáneamente, estas son:

- Sistema enactivo: El conocimiento se adquiere a través de las acciones físicas, generalmente este sistema se utiliza en los primeros años de vida. La interacción con el entorno sirve para aprender, basándonos en la imitación.
- Sistema icónico: La adquisición de conocimiento se ayuda de elementos visuales como fotografías, dibujos e imágenes que sirven para aportar información más allá de ellos, siendo este sistema la transición de lo concreto a lo abstracto.
- Sistema simbólico: Se obtiene el conocimiento por medio del lenguaje hablado o escrito, en este sistema el nivel de desarrollo intelectual debe ser mayor ya que se tiene que tener capacidad de abstracción.

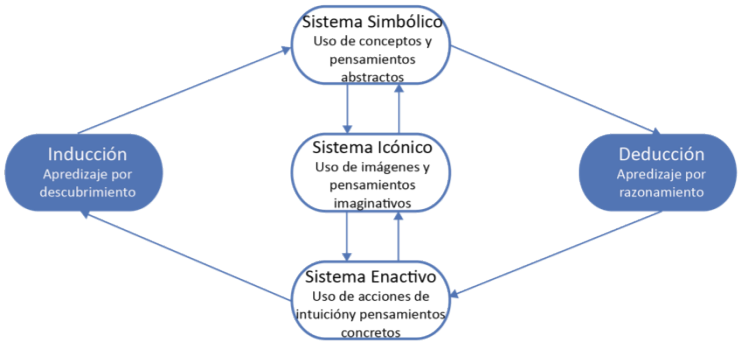


Imagen 2–9. Categorización de Brunner. Fuente: Elaboración propia (2017).

David Paul Ausubel (1918 – 2008)

Imagen 2–10. David Paul Ausubel.

El enseñar para Ausubel era el ayudar al alumno a perfeccionar el conocimiento que ya tiene y aumentarlo sin imponer un temario a memorizar, siendo la educación una transmisión de información bilateral, por lo que la primera etapa es averiguar qué es lo que sabe el alumno para conocer la lógica de su modo de pensar.

El conocimiento verdadero solo puede nacer cuando los nuevos contenidos tienen un significado a la luz de los conocimientos que ya se tienen.

Esta es la idea con la que Ausubel trabajo para el aprendizaje significativo, es decir todo nuevo conocimiento se debe conectar con uno anterior, así el conocimiento anterior con el nuevo crea un nuevo significado y lo integra a la estructura cognitiva que el alumno ya posee.

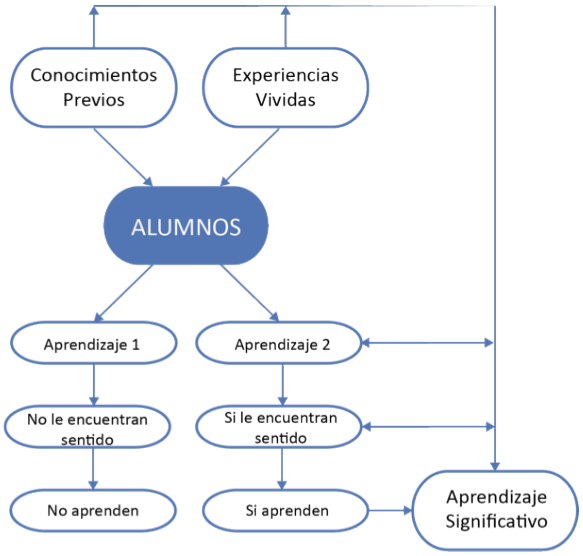


Imagen 2–11. Aprendizaje significativo segun Ausubel. Fuente: Elaboración propia (2017).

Joseph Donald Novak (1932 -)

Imagen 2–12. Joseph Donal Novak.

Las investigaciones de Novak se centran en la representación del conocimiento en el aprendizaje humano.

Según la teoría de Novak la clave para generar un aprendizaje significativo en los alumnos es la reasignación de los conceptos nuevos y su ordenamiento en mapas conceptuales en base a los pensamientos, sentimientos y actos de los alumnos.

Imagen 2–13. Teoria de Novak.
Fuente: Elaboración propia (2017).

Resumiendo, la Teoría Cognitiva:

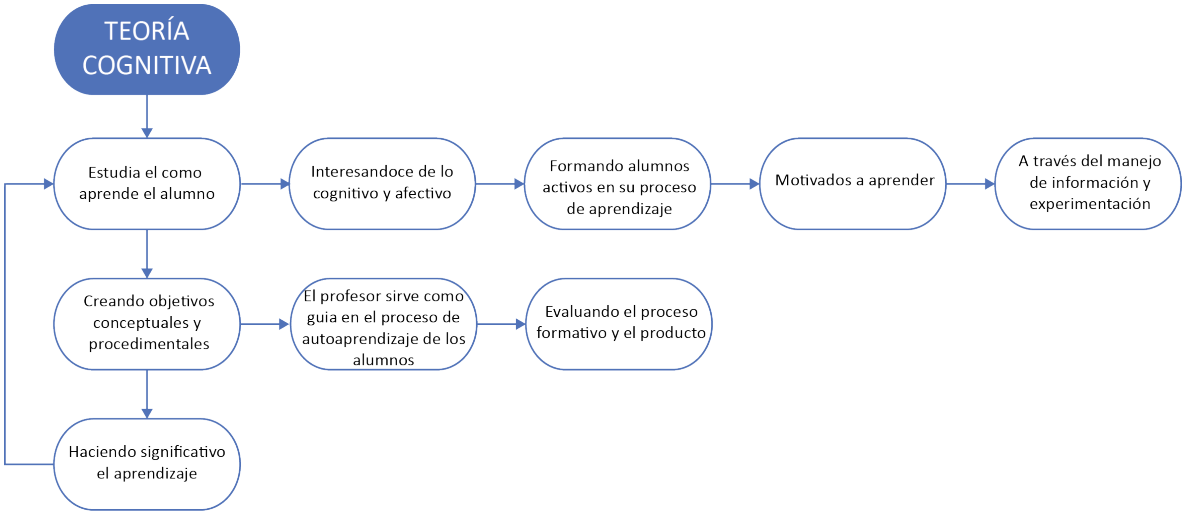


Imagen 2–14. Teoría Cognitiva. Fuente: Elaboración propia (2017).

2.2.2.3 Teoría Sociocultural

La Teoría Sociocultural surge a partir de la teoría cognitiva, pero se basa en el contexto sociocultural de cada alumno dando vital importancia a la interacción tanto de los profesores como de los compañeros para poder desarrollar el nuevo conocimiento que se adquiere de una forma flexible y abierta de acuerdo a su interacción con el medio.

A continuación, mencionaremos a los principales representantes de esta teoría y hablaremos de su contribución.

Lev Semiónovich Vigotsky (1896 – 1934)

Imagen 2–15. Lev Seminóvich Vigotsky

La teoría sociocultural de Vigotsky describe al aprendizaje como un proceso colaborativo, sostiene que los alumnos desarrollan los nuevos conocimientos/habilidades o mejorando los que ya tiene mediante la interacción social, permitiéndole interiorizar las estructuras de pensamiento de la sociedad que lo rodea y apropiándose de ellas.

El papel de dirección y apoyo lo llevan los profesores o los compañeros más avanzados ayudando al alumno a cruzar la brecha entre lo que son capaces de hacer y lo que aún no pueden hace por si solos.

La teoría Sociocultural de Vigotsky hace énfasis en el potencial del alumno asegurando que este es de un valor incalculable, de la misma forma asegura que el desarrollo de un alumno no puede ser una norma para los demás por las variantes culturales que existen en cada uno de los casos.

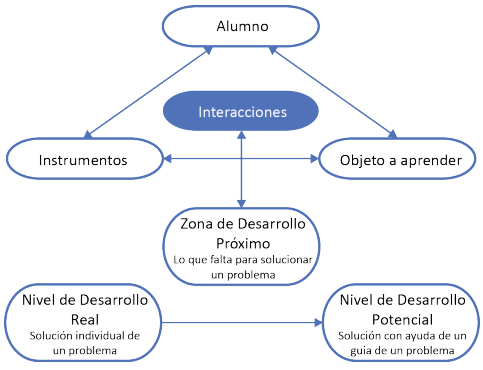


Imagen 2–16. Teoría Sociocultural de Vigotsky. Fuente: Elaboración propia (2017).

2.3.2.3.2 Reuven Feuerstein (1921 – 2014)

Imagen 2–17. Reuven Feuerstein.

Feuerstein desarrolló un sistema de evaluación que propone evaluar las capacidades y cambios que el alumno demuestra durante el proceso de evaluación, denominado Programa de evaluación Dinámica de la Propensión al Aprendizaje (EDPA) más que predecir un desempeño futuro midiendo las características estables del sujeto, se miden los cambios que puede haber en estas características.

Postulando de esta forma que el alumno cuenta con una inteligencia dinámica, flexible y receptora a la intervención de otra persona, que puede ser un profesor o algún compañero más avanzado. De esta forma el alumno puede producir nueva información.

Analizando la Teoría Sociocultural, podemos resumir lo siguiente:

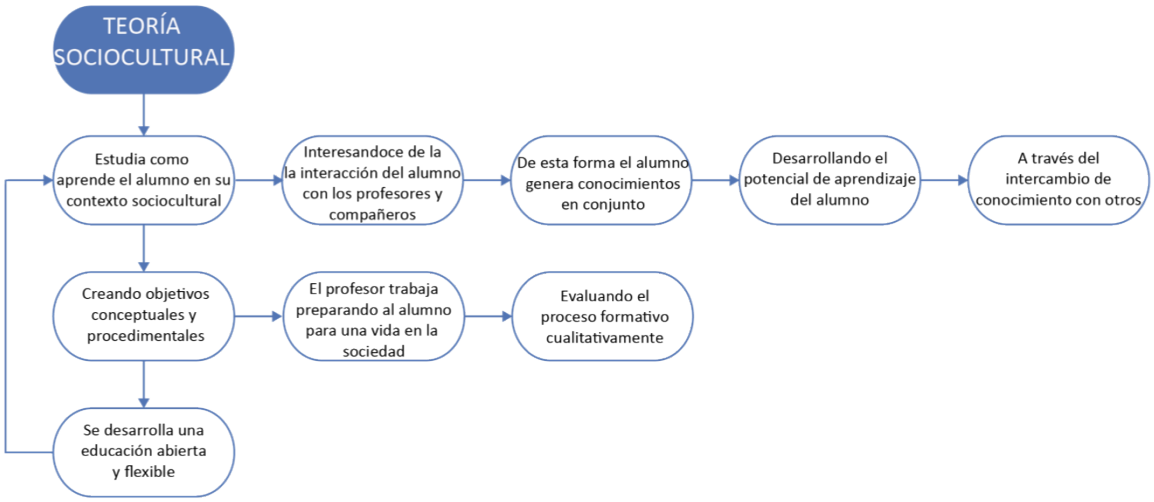


Imagen 2–18. Teoría Sociocultural. Fuente: Elaboración propia (2017).

2.2.2.4 Teoría Constructivista

La Teoría Constructivista toma elementos de las anteriores, postulando que sólo se puede generar conocimiento nuevo a partir del conocimiento ya existente.

Volviéndose el alumno la persona responsable de construir nuevo conocimiento a partir de la información que recibe de los profesores y de su propia experiencia, trasladando y aplicando su conocimiento a la práctica en un escenario real.

En esta teoría el "conflicto cognitivo" es lo que empuja al alumno a aprender, entendiéndolo como el desequilibrio de las estructuras mentales ya establecidas por el alumno cuando se enfrenta a algo nuevo que no se puede explicar con los conocimientos previos, de esta forma el alumno debe modificar con el conocimiento nuevo la estructura mental previa.

Este proceso de reconstrucción de aprendizaje se da en tres etapas:

1. Equilibrio Inicial: El alumno cuenta con una estructura mental consolidada, la cual le permite explicar el mundo.
2. Desequilibrio: El alumno se enfrenta a un "conflicto cognitivo" adquiriendo nuevas experiencias.
3. Reequilibrio: El alumno adapta los nuevos conocimientos, modificando o sustituyendo su estructura mental pasando a ser el nuevo "equilibrio inicial"

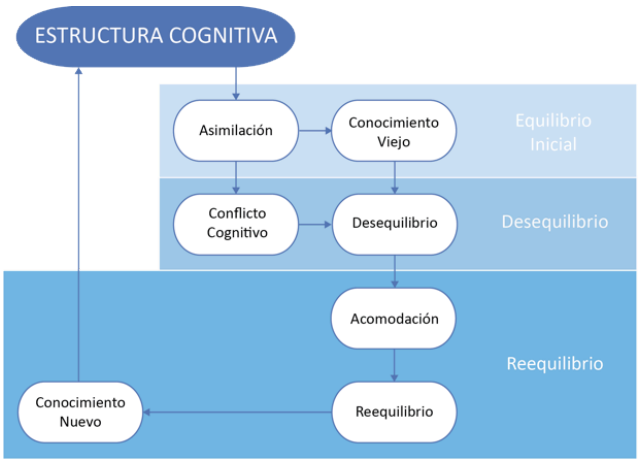


Imagen 2–19. Estructura Cognitiva.
Fuente: Elaboración propia (2017).

Resumiendo, la Teoría Constructivista de la siguiente forma:

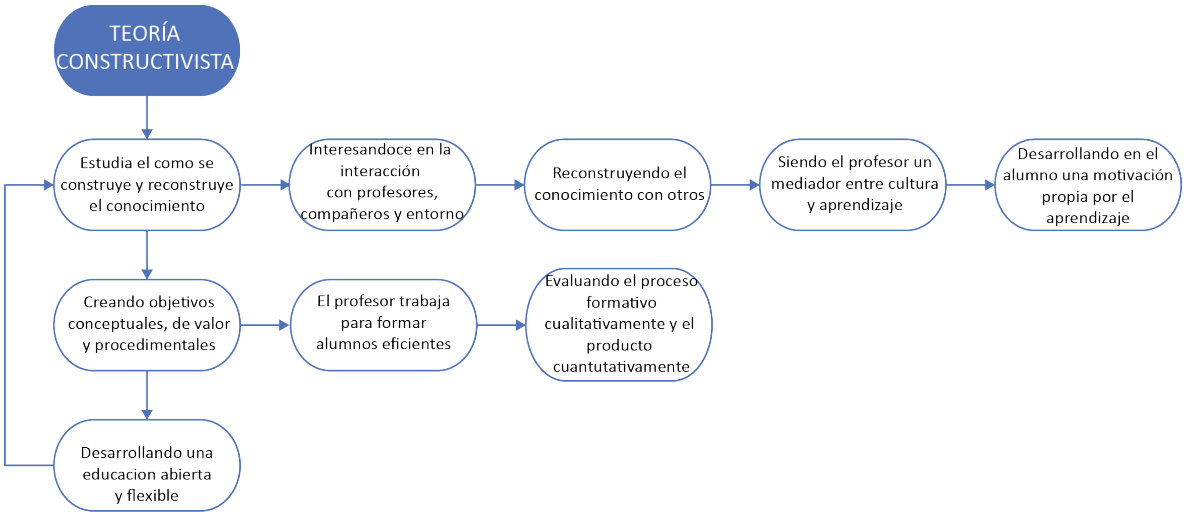


Imagen 2–20. Teoría Constructivista. Fuente: Elaboración propia. (2017).

Como hemos podido ver, a través del tiempo han cambiado los paradigmas de cómo enseñar, así como las necesidades de los alumnos al aprender. Si bien es cierto que en el pasado bastaba con la memorización de nuevos conceptos, actualmente los mismos alumnos piden más que solo repetición, así como los mismos conocimientos requieren de nuevas técnicas para ser impartidos dado a sus características particulares, dándole así al profesor un nuevo papel en el método de enseñanza para que este interés innato por aprender no se pierda. En el siguiente capítulo diseñamos la propuesta de un método de enseñanza para la impartición de los principios de programación tomando como referencia los métodos que acabamos de exponer en este capítulo, especialmente el constructivismo.

CAPÍTULO III: PROPUESTA METODOLÓGICA

En el capítulo I se abordó la problemática de la enseñanza-aprendizaje de la programación a la que se enfrentan profesores y alumnos de la carrera de Ingeniería en Computación de la FES Aragón. Este análisis originó una propuesta metodológica la cual está basada en nuestra visión, experiencia y así como en las diferentes teorías pedagógicas descritas en el capítulo II.

La metodología que desarrollamos tiene como objetivo proponer una serie de métodos que ayuden al profesor enseñar a los alumnos los principios básicos de programación, los cuales son: el análisis, el diseño, la codificación y las pruebas.

Esta metodología representa un cambio de paradigma, diferente de lo que hasta ahora ha sido la enseñanza de la programación. Dado que con esta metodología el alumno pasa de un estado pasivo a un estado activo y se convierte en un elemento dinámico de su propio aprendizaje.

Al principio habíamos planteado que la tesis abarcara sólo la clase teórica, sin embargo, conforme se fue desarrollando el trabajo, fue surgiendo la necesidad de ir diseñando diferentes tipos de elementos metodológicos para lograr una mayor consolidación de lo enseñado.

Se plantearon cinco elementos metodológicos, los cuales en conjunto, conforman la metodología. Estos elementos son:

- Clase Teórica.
- Clase de Comparación.
- Centro de Cómputo.
- Examen.
- Revisión de examen.

Tomando en cuenta estas clases, se elaboró un cronograma de cómo se podrían

impartir estas clases, el cual se encuentra en el Anexo 3. Hay que tomar en cuenta que este cronograma no abarca todo un semestre, sólo la sección referente a los principios básicos de programación.

3.1 Clase Teórica.

Este elemento es el principal de la metodología, en el cual planteamos nuestra visión del cómo se deberían de impartir los conceptos básicos de programación, de tal modo que al alumno se le enseñe la teoría, las estructuras de control y los elementos del ciclo de programación, además de fomentar en el alumno el pensamiento lógico y trabajo en equipo.

Una característica clave de este método, yace en el hecho de que el profesor no funge más como figura principal, sino más bien como guía en el proceso de aprendizaje del alumno.

Las etapas que conforman la clase teórica son:

- Planteamiento del problema.
- Propuesta de solución.
- Revisión de propuestas.
- Presentación del tema.
- Aplicación del tema a la propuesta.
- Consolidación.

A continuación, explicamos cada una de estas etapas. En el Anexo 4 se encuentra un ejemplo de una Clase Teórica.

3.1.1 Planteamiento del problema.

Como se llegó a expresar en el capítulo I, uno de los mayores problemas a los que se enfrentan los alumnos en el momento de programar, es precisamente el hecho de que no han comprendido el problema, he aquí la necesidad de abordar esta situación.

El planteamiento del problema es un punto muy importante, el cual no debe ser subestimado por el profesor. Es desde aquí donde se le enseña al alumno a comprenderlo, para que pueda diseñar una solución primero deberá de entenderlo, en otras palabras, en el planteamiento del problema deberá responder la pregunta ¿qué se quiere?

El planteamiento del problema se deberá de cumplir con las siguientes características:

- Debe ser expresado en un lenguaje no técnico y fácil de entender, dado que se busca que los alumnos se sientan cómodos y no excluidos por usar un lenguaje técnico.
- Orientado a una situación en la que los alumnos puedan sentirse identificados, con la finalidad de que estos sientan interés y/o se sientan familiarizados con el problema y de este modo captar su atención desde el principio.

3.1.2 Propuesta de solución

En esta etapa se plantea que sea el propio alumno y no el profesor quien proponga la solución al problema, por lo que de forma indirecta el alumno comenzará un proceso de análisis y diseño. De esta manera se va involucrando al alumno dentro de las primeras fases de desarrollo, además de ir fomentando estas habilidades.

El diseño de un algoritmo no va más allá de una secuencia de pasos, por lo que para el alumno no debe ser una actividad complicada, salvo que la situación lo requiera, la intervención del profesor no debe ir más allá de una asesoría para redirigir u orientar el diseño. En consecuencia, el profesor no diseñaría ningún algoritmo a lo largo del curso, inclusive si el tema pudiera parecer muy simple, esta actividad recae totalmente en el alumno desde la primera hasta la última clase.

El propósito de que el profesor no realice un algoritmo es con el fin de que los alumnos no traten de imitarlo cuando diseñen sus propios algoritmos.

Podemos esperar que los primeros algoritmos sean sencillos, pero hay que tomar en cuenta que todo algoritmo sin importar el nivel de detalle es aceptable.

Un ejemplo de esto, está planteado en la siguiente tabla, donde se muestran dos algoritmos enfocados a resolver una suma de números enteros pero con diferente nivel de detalle:

Nivel básico de algoritmo	Nivel detallado de algoritmo
1. Dame el primer número. 2. Dame el segundo número. 3. Suma el primer número con el segundo.	1. a= Dame un número entero. 2. Valida que el valor recibido sea un número entero 3. b= Dame un número entero. 4. Valida que el valor recibido sea un número entero 5. c = a + b 6. Muestra c

Tabla 3-1. Ejemplo de diseño de algoritmo. Fuente: Elaboración propia (2017).

El algoritmo de la izquierda sería el esperado por un alumno y el de la derecha por el profesor, el algoritmo de la derecha refleja una mayor experiencia, no obstante, ambos son válidos.

Al no haber mostrado al alumno cómo hacer un algoritmo, al buscar la solución al problema este lo estará diseñando de forma inconsciente.

Conforme se avance en el curso se espera que los alumnos pasen de algoritmos sencillos a más elaborados y empecen a diseñar pseudocódigos. Esto reflejará un progreso en el pensamiento de los alumnos, dado que ya estarán diseñando en función del paradigma.

Expuesto lo anterior, en esta etapa del método, dependiendo de la dificultad del problema, el profesor solicitará a los alumnos que trabajen en equipos o de forma individual, para desarrollar una solución. Esta deberá de ser expresada en una secuencia de pasos, es decir, en un algoritmo.

Las razones por la que se introduce el trabajo en equipo dentro de la metodología son:

- Para que los alumnos que al principio no sientan la confianza de preguntar al profesor sus inquietudes, tengan un espacio para resolver dudas a través de sus compañeros.
- Que tengan una retroalimentación o mejor compresión de los temas. Con forme se avanza en el curso, los alumnos irán incorporando a sus diseños los temas aprendidos, si por alguna circunstancia algún alumno no ha dominado por completo un tema, el trabajo en equipo le será de ayuda, dado que puede aprender mediante sus compañeros de equipo la forma en que se implementan.

En el caso de que se trabaje en equipos, estos deberán de estar conformados por tres personas. El motivo de plantear un máximo de tres personas, es porque resulta más fácil la interacción de los miembros, mientras que en equipos más grandes, los miembros requieren más tiempo para ponerse de acuerdo en una solución, además de que comúnmente no todos los miembros trabajan.

Uno de los cambios más notables con respecto a una clase tradicional de programación, yace en el hecho de que primero el alumno resuelve el problema y después se le enseña el tema o estructura de programación.

3.1.3 Revisión de propuestas

Teniendo elaboradas las propuestas, el profesor seleccionará algunas para ser presentadas por los alumnos hacia el resto del grupo, de este modo los demás alumnos

comenzarán un análisis y retroalimentación al ir comparando con sus respectivas propuestas.

En esta etapa el profesor funge con el papel de moderador del análisis, sólo en caso de que los alumnos no lleguen a una solución acorde al problema, el profesor toma el papel de guía, cuestionándolos, y así guiarlos a una solución más adecuada.

Habiendo sido expuestas las propuestas, los alumnos adecuarán o implementarán algunas características a sus respectivas propuestas, si así lo consideran pertinente.

3.1.4 Presentación del tema

En esta etapa el profesor presenta el tema a través del lenguaje de programación. La enseñanza del tema debe ser muy concreta, no se pretende que el profesor desarrolle un programa completo para enseñar el tema, sino que simplemente enseñe la estructura en cuestión. Este enfoque se debe a que en la siguiente etapa será el propio alumno el que implemente el tema a su propuesta de solución.

La enseñanza del tema no es una actividad trivial como puede parecer, sino que hay que cumplir ciertos criterios al momento de enseñar, los cuales se mencionan a continuación.

Explicación del tema

En este punto se da una explicación del tema, ¿qué es?, ¿Para qué sirve?, ¿Por qué es importante el tema? El propósito de responder estas preguntas es para proporcionarle un referente al alumno para que este tenga un contexto sobre el nuevo conocimiento. Esto en base al planteamiento de la Dra. Rocío Quesada sobre los conocimientos previos que se mencionó en el capítulo I.

Presentación de la estructura

En este punto se responde a la pregunta ¿Qué es? El profesor presenta la o las estructuras pertenecientes al tema, así como de sus posibles variaciones, por ejemplo,

suponga que se está presentando la estructura de bifurcación *if*, el profesor deberá mostrar todas las variantes de esta estructura.

Recomendamos el uso de colores constantes y no aleatorios para las diferentes partes del código, con el fin de aumentar la legibilidad de este y facilitar el copiado del pizarrón al cuaderno, así el alumno tendrá una mejor asimilación de la nueva estructura enseñada.

Interpretación de la estructura

Presentada la estructura, el profesor deberá de dar una explicación detallada de cómo se interpreta la estructura, esto con el fin de que el alumno comprenda el funcionamiento, respondiendo a la pregunta ¿Cómo funciona?

Implementación

A diferencia de los puntos anteriores en los que se muestra al alumno de forma concreta la estructura, en este punto se muestra la implementación con un ejemplo sencillo, recordando que no se requiere de la elaboración de un programa completo sino de una parte orientada a la estructura.

Como se llegó a mencionar anteriormente, el profesor en ningún momento elabora algoritmos a lo largo del curso, tampoco llega a implementar un programa completo, reiterando que estas actividades recaen directamente en el alumno.

Prueba de escritorio

Teniendo la estructura implementada, el profesor ejecutará una prueba de escritorio con el fin de mostrar a los alumnos cómo se comporta la estructura. Es muy importante que el profesor la realice cada vez que enseñe un nuevo tema. Otra ventaja es que muestra al alumno cómo se comportan las variables al interactuar con las estructuras de control.

3.1.5 Aplicación del tema a la propuesta

Habiendo expuesto el tema, el profesor solicitará a los alumnos que implementen su solución al lenguaje de programación. Esto se hace con la finalidad de que los alumnos empiecen un proceso de asociación entre la nueva estructura de programación y su algoritmo. Esta asociación es muy importante porque en este punto los alumnos empiezan a entender cuando se implementan las diferentes estructuras o sentencias a un problema dado.

La siguiente tabla ejemplifica las asociaciones que se irán dando conforme a diferentes temas.

Acción	Estructura o sentencia
Comunicarme	Sentencia que imprime texto en la pantalla
Obtener información	Sentencia que lee información del teclado
Toma de decisiones	Estructuras de bifurcación
Repetición de una tarea	Estructuras de repetición

Tabla 3-2. Asociaciones de acciones con estructuras. Fuente: Elaboración propia (2017).

Al terminar, el profesor solicitará a algunos equipos que escriban su código en el pizarrón, para que con todo el grupo se revise la sintaxis y se resuelvan dudas. En el caso de la corrección de errores de sintaxis, el profesor deberá de hacer énfasis en el error, explicando la corrección de este sin importar si con anterioridad ya había corregido el mismo tipo de error sintáctico.

3.1.6 Consolidación

Finalmente el profesor dará un ejercicio en donde se utilicen las estructuras previamente enseñadas para que los alumnos de manera individual diseñen una solución a este. Posteriormente se revisarán algunas propuestas con todo el grupo y se escogerá solo una, por último los alumnos codificaran este algoritmo como tarea.

Las tareas que se dejen en la Clase Teórica serán necesarias para poder trabajar en el centro de cómputo, estas tendrán que llevar un nombre específico y todos los archivos deben de encontrarse en una sola carpeta.

3.2 Clase en el Centro de Cómputo

En el capítulo I se menciona que una de las demandas de los alumnos es la impartición de las clases de programación en el Centro de Cómputo, sin embargo, si se llegara a dar un curso completo de programación en este, aunque pudiera existir una mejora, esto no garantizaría un mayor nivel con respecto a una clase tradicional.

Uno de los grandes problemas para el alumno es no saber codificar correctamente, más aún no saber corregir los errores de sintaxis cuando se le presentan durante la compilación, por lo que en esta metodología, aunque no consideramos que cada clase de programación se dé en el Centro de Cómputo, no negamos la importancia de que los alumnos corrijan estos errores frente a un equipo de cómputo, pero lo planteamos con un enfoque diferente.

Las clases en el Centro de Cómputo son únicamente para reforzar la codificación de los alumnos, en especial la sintaxis, y llevar a cabo la última fase dentro del ciclo de desarrollo, la fase de pruebas. Para esto el papel del profesor será solamente de asesor cuando surjan dudas. Insistimos en hacer énfasis que la clase en el Centro de Cómputo no es para tener una clase teórica. Para dar esta clase se deben cumplir las siguientes características:

- El grupo se dividirá en tres, considerando grupos grandes. Atendiendo a los dos primeros grupos en una sesión y al último grupo en la siguiente, esto es para poder dar una mejor atención a los alumnos.
- La duración de la Clase en el Centro de Cómputo será de la mitad, es decir, si la clase dura 120 minutos (martes y jueves) las sesiones del Centro de Cómputo serán

de 60 minutos por grupo, y si la clase dura 90 minutos (lunes, miércoles y viernes) las sesiones en el Centro de Cómputo serán de 45 minutos por grupo.

- Los alumnos deben de presentarse con las tareas asignadas en las clases teóricas, cabe resaltar que no es necesario que los programas sean totalmente funcionales. Al trabajar con las tareas en lugar de pedirles que diseñen un código nuevo tiene dos propósitos: primero, evitar que los alumnos consuman tiempo en el análisis y diseño, enfocándose solamente en la codificación y pruebas, siendo este el propósito de la clase; y segundo, que al corregir el código en el que invirtieron tiempo y esfuerzo, habrá una mayor consolidación.

  El desarrollo de la Clase de Centro de Cómputo será el siguiente:

  1. Entrada al Centro de Cómputo: Los alumnos trabajarán de manera individual, utilizando cada quien un equipo de cómputo.
  2. Revisión de tareas: Los alumnos pasarán la primera tarea al compilador y ejecutarán, en caso de que su tarea tenga algún problema y no puedan encontrar el error, le notificará al profesor.
  3. Corrección de tareas: El profesor le pedirá a alumnos que no hayan tenido problemas de compilación que ayuden a los compañeros que sí. En caso de que ninguno consiga encontrar el error, el profesor cambiará su papel de moderador por el papel de guía para ayudarlos a encontrarlo. Una vez que ningún alumno tenga errores de compilación se repetirán estos pasos con cada una de las tareas.
  4. Consolidación: El profesor les proporcionará un código con errores de sintaxis para que lo corrijan. Con el fin de que los alumnos sepan detectar más fácilmente estos.

Un ejemplo de cómo implementar la clase de cómputo se encuentra en el Anexo 5, considerando la sesión de 45 minutos, en caso de ser de 60 minutos los tiempos se pueden extender en los puntos que el profesor considere necesarios.

3.3 Clase de comparación

Uno de los problemas que se ha observado en los alumnos, está relacionado con el hecho de que no saben distinguir cuándo utilizar las diferentes estructuras de bifurcación o de repetición, es decir, no saben cuándo utilizar una estructura de múltiples *if* de una *if-esleif*, o una estructura *for* de una *while* o de una *do-while*. Por tal motivo surge la necesidad de tener clases de comparación, las cuales estarán dirigidas para mostrar a los alumnos cuando utilizar cada una de ellas.

Durante el curso sólo habrá dos clases de este tipo, una al terminar las estructuras de bifurcación y otra al terminar las estructuras de repetición.

El desarrollo de estas clases es muy parecido a la clase teórica y es el siguiente:

1. Planteamiento del problema. El profesor deberá de plantear un problema muy sencillo enfocado a estos temas, con el fin de que los alumnos se centren principalmente en la implementación de la estructura en cuestión.
2. Propuesta de solución: En equipos de tres los alumnos crearán un pseudocódigo con la estructura de su elección enfocada a resolver este problema.
3. Revisión de propuestas: El profesor seleccionará un equipo por cada estructura para que expongan el pseudocódigo en el pizarrón. Una vez que todas las estructuras estén en el pizarrón, el profesor preguntará a los alumnos si encuentran errores en el uso de las estructuras de bifurcación o repetición según sea el caso.
4. Consolidación: El profesor les cuestionará a los alumnos cuál es la estructura más recomendable para resolver el problema planteado y al ver que no existe sólo una forma de solucionar el problema, comenzarán un debate de cuál estructura es mejor, cada uno defendiendo su propuesta. El rol del profesor será de moderador y los cuestionará para llevarlos a la conclusión de que no importa qué tipo de estructura elijan, cualquiera sirve, y dependiendo de la naturaleza del problema será el tipo de estructura a utilizar.

En el Anexo 6 se encuentra un ejemplo de la Clase de Comparación, considerando la clase de 90 minutos, en caso de ser de 120 minuto los 30 minutos restantes se distribuirán a consideración del profesor.

3.4 Examen

El objetivo del examen más que un medio para evaluar al alumno, es una herramienta para que el profesor determine qué temas se deberán de reforzar, y así poder reorientar sus clases para repasar dichos temas.

Proponemos exámenes sencillos de opción múltiple, que se impartirán en la segunda clase del Centro de Cómputo, por lo que de la misma manera será de la mitad de tiempo de una clase, es decir, 45 minutos (lunes, miércoles y viernes) ó 60 minutos (martes y jueves).

3.5 Revisión de examen

Esta clase tiene por objetivo repasar los temas en que los alumnos aún tengan dudas, por medio de la revisión de examen.

Los exámenes deben de estar calificados, el profesor los entregará a los alumnos y la revisión comenzará con todo el grupo.

El profesor tendrá el papel de moderador, incitando a los alumnos que aún tienen dudas a preguntar, y a los otros a explicar sus respuestas.

La revisión se llevará acabo pregunta por pregunta, pidiendo que los alumnos den la respuesta correcta y la explicación, así como el por qué las otras opciones no los son. Si al terminar la explicación continúan existiendo dudas, el profesor retomará su papel de guía para ayudar a resolver estas y reforzar el tema donde el profesor detectó un porcentaje significativo de errores en el examen.

CONCLUSIONES

Si bien en el presente trabajo se propuso una metodología para enseñar los principios de programación, aplicada a la carrera de Ingeniería en Computación de la FES Aragón de la UNAM, ésta no se implementó porque se trata de una investigación teórica, que puede servir como antecedente para futuras investigaciones. La segunda fase de la investigación será la evaluación de esta propuesta metodológica en coordinación con los profesores de la materia.

Entre las contribuciones de esta metodología, podemos mencionar las siguientes:

La metodología cuenta con cuatro tipos de clase: clase de presentación de estructuras, clase de comparación entre estructuras similares, clase de codificación y clase de revisión de examen, lo que permite elaborar un cronograma de clases que abarca los temas básicos de programación.

Esta propuesta no sólo cambia el método de enseñanza de los maestros, sino que también da un modelo nuevo de aprendizaje a los alumnos, que podrán ajustar a sus características particulares, según la manera de aprender de cada uno.

Con este método se pretende que los alumnos entiendan y realicen cada una de las fases de desarrollo de un programa, que son el análisis, el diseño, la codificación y las pruebas. De este modo, el alumno asume un papel más activo en su propio aprendizaje. Esta manera de proceder contrasta con algunas clases tradicionales en las cuales el alumno sólo se limita a copiar.

Además, esta metodología fomenta el trabajo en equipo, habilidad que le será de gran utilidad al alumno, no sólo en su formación académica sino también en su desempeño profesional y personal.

Así mismo, la metodología planteada en esta tesis propone que los profesores asuman un nuevo papel, el de mediadores, de tal modo que se conviertan en guías para los

alumnos y se olviden de la figura autoritaria que todavía algunos relacionan con el quehacer docente.

De los problemas expresados por los alumnos en el capítulo I, retomamos algunos que son recurrentes. Por un lado, los alumnos piden que los profesores les enseñen las bases de la programación, ya que ellos mismos aceptan que no comprenden cómo empezar a programar, es decir, no saben qué deben escribir o cómo abordar el problema, situación que refleja la ausencia de habilidades analíticas y de diseño por parte de los alumnos.

Por otro lado, los alumnos no saben cómo convertir su solución en código, no entienden el funcionamiento de las sentencias o de las estructuras y, mucho menos, la totalidad de un código para solucionar un problema determinado, lo que refleja un desconocimiento del lenguaje de programación y una falta de cohesión entre el lenguaje de programación y el diseño de la solución.

Los alumnos encuestados también piden que se impartan las clases de programación en un centro de cómputo, dado que ellos consideran que de esta forma aprenderán a programar. Sin embargo, es necesario que los alumnos comprendan que el trabajo de análisis y diseño se hace previamente en el aula, sin ayuda del equipo de cómputo, y que más bien la visita a este centro es el siguiente paso.

Por último, los alumnos comentaron que los profesores dan por hecho que ellos ya saben programar y que por eso les asignan tareas de programación, sin primero enseñarles las bases. Desafortunadamente, algunos profesores se enfocan en enseñar la codificación, debido a que no cuentan con un método de enseñanza que incluya el proceso de análisis y diseño.

Dicho lo anterior, es posible afirmar que la metodología propuesta en esta tesis es una herramienta que empodera a los alumnos respecto de su propio aprendizaje, pues son

ellos quienes ejecutan cada una de las fases del ciclo de desarrollo de un programa e identifican y corrigen las fases en las cuales tienen deficiencias.

Al realizar este trabajo nos enfrentamos a diversas limitaciones. Por un lado, el diseño de la encuesta fue nuestro primer obstáculo, debido a nuestro escaso conocimiento en la elaboración de este tipo de instrumentos. Si bien diseñamos una encuesta que nos permitió obtener la información que buscábamos, también es cierto que aplicamos otras encuestas que no aportaron resultados significativos para la investigación, por lo que decidimos omitirlas.

Por otro lado, con el propósito de encuestar a toda una generación y evitar que los participantes se repitieran, aplicamos la encuesta a los tres grupos de una misma materia. Desafortunadamente, en la realidad, sólo estuvieron presentes dos tercios de la generación, lo cual redujo significativamente nuestra muestra representativa. Además, es importante señalar que existe la posibilidad de que los profesores de la materia de Programación cambien de generación en generación, por lo que la encuesta se tendría que aplicar en diferentes periodos, con la finalidad de tener un estudio más completo.

Por último, decidimos hacer una entrevista a los profesores de programación, dado que al haber un universo muy reducido de ellos no era viable la realización de una encuesta. La aplicación de este instrumento nos permitió acercarnos a la situación particular a la que se enfrentan y conocer su opinión del origen-causa de esta problemática.

Cabe destacar que los diferentes métodos pedagógicos son planteamientos generales de enseñanza, por lo que gran parte del esfuerzo de esta tesis fue adaptar estos métodos a la materia de nuestro interés, es decir, a la enseñanza de los principios de la programación, entendiendo ésta como una actividad abstracta cuyo proceso de desarrollo se conforma de diferentes etapas, que ya se mencionaron.

Finalmente, consideramos que al implementar este método de enseñanza es posible lograr que tanto los alumnos como los profesores consigan aprender y enseñar los principios de programación.

ANEXO 1

Universidad Nacional Autónoma de México
Facultad de Estudios Superiores Aragón

Encuesta sobre el proceso de enseñanza-aprendizaje en la FES Aragón en la carrera de Ingeniería en Computación de la materia Programación Orientada a Objetos.

Conteste la siguiente encuesta de forma objetiva y en el caso de las preguntas abiertas describa de forma explicita lo que se le pide.

1. ¿Sabes programar?
   O Sí **O No**

2. ¿Dónde aprendiste a programar?
   O Escuela **O Trabajo** **O Cursos**
   O Libros **O Otro:** _____

3. En caso de haber contestado **Sí** en la pregunta 1, especifique el paradigma que manejas:
   O Estructurado **O Orientado a Objetos**

4. En caso de haber contestado **Estructurado** en la pregunta anterior, define los siguientes conceptos:

   Estructura de repetición (ciclos): _____
   _____
   _____
   _____
   Estructura de selección (if, if-else): _____
   _____
   _____
   _____
   Función: _____
   _____
   _____
   _____

5. En caso de haber contestado **Orientada a Objetos** en la pregunta 3, define los siguientes conceptos:

   Clase: _____
   _____
   _____
   _____
   Objeto: _____
   _____
   _____
   _____
   Método: _____
   _____
   _____
   _____

6. ¿Cuándo tenías que hacer un programa hacías algoritmos y/o seudocódigos?
   O Sí **O No**

Universidad Nacional Autónoma de México
Facultad de Estudios Superiores Aragón

7. ¿Sabes desarrollar algoritmos y seudocódigos?
O Sí **O No**

8. ¿Sabes cómo analizar los problemas para posteriormente codificar su solución?
O Sí **O No**

9. ¿Se te enseñó a codificar (usar un lenguaje)?
O Sí **O No**

10. ¿Se te enseñó a programar (hacer algoritmos, seudocódigos, a usar el paradigma en cuestión)?
O Sí **O No**

11. ¿La clase se llevó a cabo en algún centro de cómputo?
O Sí **O No**

12. De haber contestado **Sí** en la pregunta anterior ¿Con qué frecuencia?
O Siempre **O 1 a 2 veces por semana** **O 1 a 2 veces por mes** **O Casi nunca**

13. Tenias en casa el siguiente equipo para estudiar programación:
O Computadora **O Compilador** **O No**

14. ¿Tu profesor utilizaba algún medio didáctico, por ejemplo, proyector?
O Sí **O No**

15. En caso de haber contestado **Sí** en la pregunta anterior, indica de que tipo

_____
_____

16. En caso de haber contestado **Sí** en la pregunta 14, ¿Se te facilitó aprender con esto?
O Sí **O No**

17. En caso de haber contestado **No** en la pregunta anterior, ¿Por qué no?

_____
_____
_____

18. ¿De qué forma obtenías el código de los programas que veías en clase para posteriormente comprobar su funcionamiento?
O Los copiaba del pizarrón **O El maestro daba copias**
O Por algún medio Web **O Otro _____**

19. ¿Durante el curso se plantearon problemas reales en donde se les daba solución mediante un programa informático?
O Sí **O No**

20. ¿Tu profesor te generaba el interés de aprender más temas de programación?
O Sí **O No**

Universidad Nacional Autónoma de México
Facultad de Estudios Superiores Aragón

21. ¿El curso se desarrolló en un ambiente óptimo para ti?
O Sí **O No**

¿Por qué? _____
_____
_____

22. ¿Entendías el código de los programas que tu profesor te daba?
O Sí **O No**

23. En caso de haber contestado **No** en la pregunta anterior, ¿por qué?
O Me resultaba confuso el código **O No entendía el porque del código** **O Ambas**

24. Cuando estabas en alguna clase de Programación y no te quedaba claro algún tema, ¿preguntabas al profesor?
O Sí **O No** **O A veces**

25. En caso de haber contestado **No** o A veces en la pregunta anterior, ¿por qué no preguntabas?
O Vergüenza **O No te daba confianza el profesor**
O Falta de Interés **O Otro** _____

26. ¿Qué se te dificulta más?
O Programar **O Codificar** **O Ambos** **O Ninguno**

27. Durante el semestre se te iban dejando programas ¿Qué hacías?
O Siempre los hice **O Hacía el intento** **O Nunca los hacía**
O Hice la mayoría **O Al principio los hacía** **O Los conseguía**

28. ¿Buscas información fuera del salón de clases?
O Sí **O No**

29. En caso de haber contestado **Sí** en la pregunta anterior ¿En donde buscas la información?
O Libros **O Internet** **O Maestros**
O Cursos **O Otro**_____

30. ¿Cuáles son los problemas que enfrentas o enfrentaste al aprender programación?

_____
_____
_____

31. ¿Cómo te gustaría que te enseñaran programación?

_____
_____
_____

ANEXO 2

Universidad Nacional Autónoma de México
Facultad de Estudios Superiores Aragón

Encuesta para profesores sobre el proceso de enseñanza-aprendizaje en la FES Aragón en la carrera de Ingeniería en Computación de la materia Programación Orientada a Objetos.

1. Al inicio del curso ¿usted realiza un muestreo para ver si sus alumnos vienen con suficientes bases?, es decir, si los alumnos manejan las bases de la Programación Estructurada.

2. ¿Considera que los alumnos tienen las suficientes bases de programación para cursar la materia POO?

3. En caso de que un número considerable de alumnos no cuente con los conocimientos suficientes, usted ¿qué hace?

4. ¿Tiene identificados los temas que se les dificultan a sus alumnos? ¿Cuáles son? ¿Por qué cree que se les dificulten?

5. ¿Considera suficiente el tiempo del semestre para cubrir el temario?

6. ¿Con cuántos alumnos le sería más sencillo enseñar?

7. ¿Cuándo usted enseña un nuevo tema explica el objetivo de este?

8. ¿Les enseñaba como analizar un problema de tal forma que ellos puedan implementar la solución mediante la POO?

9. Cuando daba un código para explicar un tema ¿sus códigos eran sencillos o elaborados?

10. Cuando les pone un código a sus alumnos, ¿cómo da la explicación de este, de forma general o explica detalladamente línea por línea?

11. Para un alumno es difícil ya sea leer o copiar el código del pizarrón, ¿usted de alguna forma le facilitaba al alumno este?

12. ¿Usted qué prefería, dejar muchos ejercicios sencillos o dejar pocos, pero más elaborados?

13. ¿Lleva a sus alumnos al CC? ¿Con qué frecuencia?, en caso de que no ¿por qué no?

14. ¿Considera que los alumnos aprenderían mejor la materia si esta se llevara a cabo en un CC?

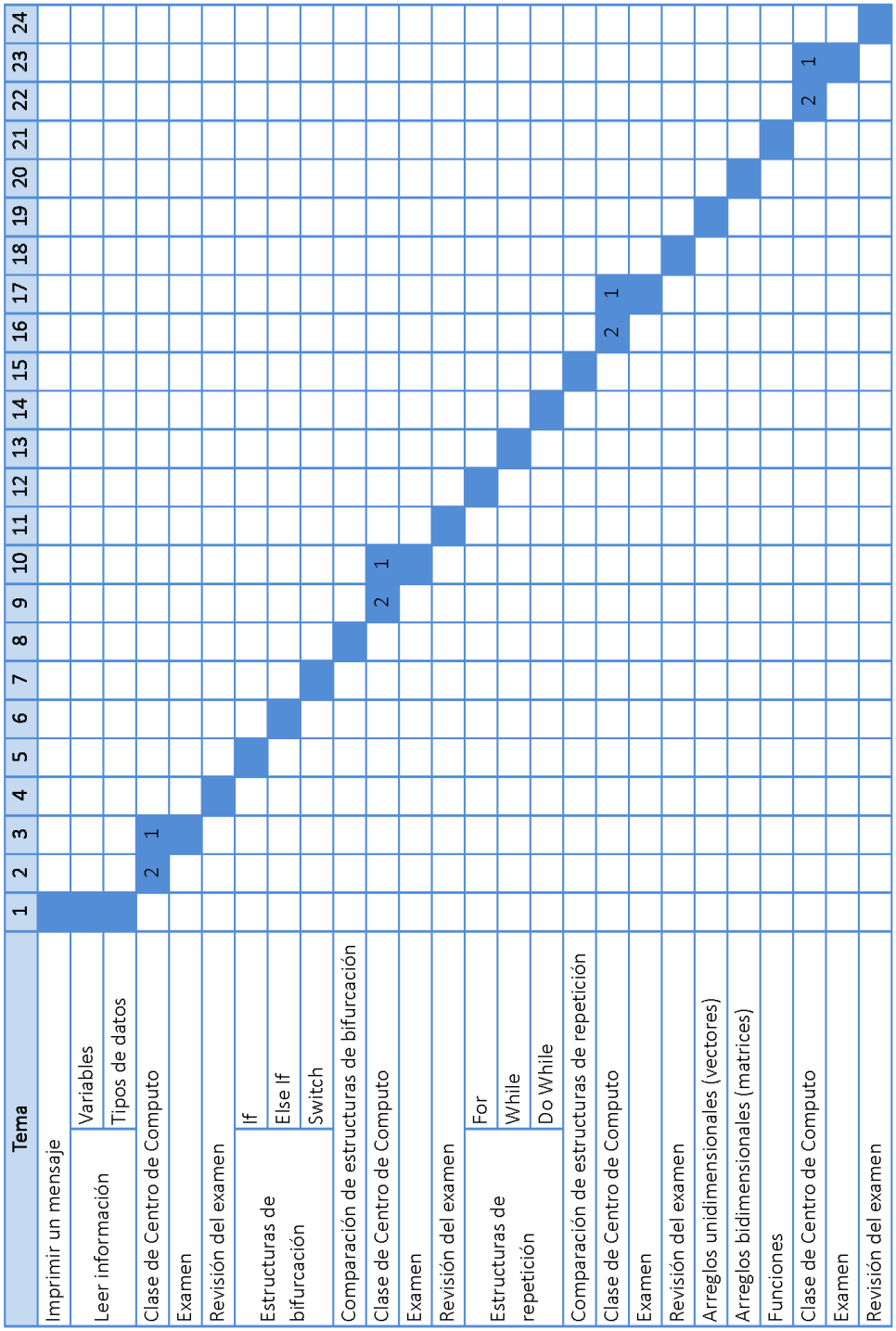
15. ¿Ha trabajado en la industria programando con java?

16. A su criterio ¿cuál es la problemática de la enseñanza-aprendizaje de la programación?

17. A su criterio ¿cuál sería una posible solución a la problemática de la enseñanza-aprendizaje de la programación?

ANEXO 3

Cronográma de clases



Tema	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Imprimir un mensaje																								
Leer información – Variables																								
Leer información – Tipos de datos																								
Clase de Centro de Computo		2	1																			2	1	
Examen																								
Revisión del examen																								
Estructuras de bifurcación – If																								
Estructuras de bifurcación – Else If																								
Estructuras de bifurcación – Switch																								
Comparación de estructuras de bifurcación																								
Clase de Centro de Computo									2	1						2	1							
Examen																								
Revisión del examen																								
Estructuras de repetición – For																								
Estructuras de repetición – While																								
Estructuras de repetición – Do While																								
Comparación de estructuras de repetición																								
Clase de Centro de Computo																								
Examen																								
Revisión del examen																								
Arreglos unidimensionales (vectores)																								
Arreglos bidimensionales (matrices)																								
Funciones																								
Clase de Centro de Computo																								
Examen																								
Revisión del examen																								

ANEXO 4

Planer Clase Teórica

Clase:	1	Tema:	Imprimir un mensaje	Objetivo:	Enseñar al alumno la manera de mostrar información en pantalla.
Tiempo:	30 minutos				

Objetivo Especifico	Ejemplo	Dinámica	Tiempo
Planteamiento del problema			
Dar la introducción al tema usando un problema muy sencillo.	Profesor: "El Sábado por la noche un amigo va a ir a una fiesta e irá la chica que le gusta, el problema es que aunque le ha querido hablar, es muy tímido y no sabe como hacerlo"	TC	1 min
Propuesta de solución			
El profesor pide una solución planteada de la misma forma sencilla en la que se planteo el problema.	Profesor: "¿Ustedes que harían? Necesito que anoten paso a paso que debería hacer mi amigo"	III	4 min
Revisión de propuestas			
Análisis de las propuestas de solución, buscando el mejor algoritmo para el problema.	El profesor pasara a tres equipos a escribir su solución en el pizarrón, para que todo el grupo escoja el mejor algoritmo.	TC	3 min
Presentación del tema			
Se presenta y explica la estructura, como se escribe y como se utiliza.	Presentación del tema, utilizando código de colores para cada estructura, y explicando para que sirve cada línea de código.	TC	13 min
Aplicación del tema			
El profesor pide a los alumnos que implementen el algoritmo que eligieron a la estructura que se les acaba de mostrar.	Profesor: "¿Creen que los pasos de cómo mi amigo le debe de hablar a la chica que le gusta se puedan aplicar a la codificación de cómo imprimir un mensaje? Necesito que en quipos de 2 personas, con quienes no hallan trabajado anteriormente, conviertan esos pasos a la codificación que les acabo de enseñar"	II	4 min
Revisión de la codificación.	Profesor: "Necesito que un equipo voluntario pase al pizarrón a anote la codificación que acaban de realizar"	TC	1 min
	Profesor: "Ahora, alguien puede ver algún error en la sintaxis de la codificación"	TC	2 min
Consolidación del tema			
El profesor dará un ejercicio sencillo.	Profesor: "Ahora que ya saben como se muestra información en la pantalla, necesito que escriban, individualmente, la codificación de cómo le pedirían su teléfono a la chica, recuerden utilizar todos los pasos que hemos hecho, análisis, diseño y codificación."	I	3 min

I Trabajo Individual II Trabajo en parejas III Trabajo en tríos TC Toda la clase

ANEXO 5

Planer Clase Centro de Cómputo Página 1 de 2

Clase:	2 y 3
Tiempo:	45 minutos

Tema:	Codificación de cómo imprimir y leer datos.

Objetivo:	Enseñar a los alumnos los pasos a seguir cuando se trabaja en el centro de cómputo y repasar la codificación de cómo imprimir y leer datos, así como encontrar los errores más comunes de sintaxis.

Objetivo Especifico	Ejemplo	Dinámica	Tiempo
Ingreso al centro de cómputo			
Dar la instrucción de que equipos utilizar.	**Profesor:** "Necesito que se coloquen cada uno en un equipo de cómputo, usando los equipos de adelante y dejando un espacio libre a un lado de ustedes"...	TC	1 min
Enseñar que se tiene que hacer en cada clase de cómputo	... "Esta es su primer clase en el centro de cómputo, así que sólo esta vez les explicaré cómo vamos a trabajar para aprovechar al máximo los 45 minutos que tenemos. Como ya les había dicho en la clase anterior es necesario que traigan la tarea, ya que es el material que vamos a ocupar para esta clase. Una vez que hayan ocupado sus lugares, abrirán el compilador y la carpeta donde viene la tarea"	TC	1 min
Enseñar el compilador	El profesor enseñará a los alumnos como usar el compilador y las funciones de este.	TC	10 min
Resolver dudas	**Profesor:** "¿Alguien tiene alguna duda?"	TC	3 min
Revisión de la tarea de imprimir datos			
Los alumnos ejecutan el primer programa de tarea y buscan errores de sintaxis	Profesor: "Necesito que abran el archivo (Nombre de la primer tarea), lo pasen al compilador y lo ejecuten, si presenta algún error encuéntrenlo y traten de corregirlo. Tienen 5 minutos"	I	5 min
Corrección de la tarea de imprimir datos			
Los alumnos corregirán la codificación con la ayuda de un compañero	**Profesor:** "Levanten la mano a quiénes no les funciona el código" El profesor asignará a un alumno que no haya tenído problemas en sus códigos para ayudar al que si los tenga. "Quiero que los alumnos que van a ayudar a encontrar el error le expliquen a su compañero el por qué y no sólo lo corrijan. Tienen 3 minutos"	II	4 min
Verificar que todos tenga correcta la sintaxis.	**Profesor:** "¿Algún equipo aún no encuentra el error?"	TC	1 min
Revisión de la tarea de leer datos			
Los alumnos ejecutan el segundo programa de tarea y buscan errores de sintaxis	Repetir las instrucciones de la "Revisión de la tarea de imprimir datos" para la segunda tarea. Darles 3 minutos	I	3 min

Clase:	2 y 3
Tiempo:	45 minutos

Tema:	Codificación de cómo imprimir y leer datos.

Objetivo:	Enseñar a los alumnos los pasos a seguir cuando se trabaja en el centro de cómputo y repasar la codificación de cómo imprimir y leer datos, así como encontrar los errores más comunes de sintaxis.

Objetivo Especifico	Ejemplo	Dinámica	Tiempo
Corrección de la tarea de leer datos			
Los alumnos corregirán la codificación con la ayuda de un compañero	Repetir las instrucciones de la "Corrección de la tarea de imprimir datos"	II	4 min
Verificar que todos tenga correcta la sintaxis.	**Profesor:** "¿Algún equipo aún no encuentra el error?"	TC	1 min
Consolidación			
Los alumnos buscaran errores de sintaxis.	**Profesor:** "Ahora que ya han encontrado algunos de los errores que más comúnmente cometen, quiero que me ayuden a encontrar los míos. Les pasaré un programa que hice para leer e imprimir datos, pero tiene errores, así que de forma individual los buscarán y corregirán, las primeras 5 personas que logren que el programa compile les daré una recompensa. Tienen 10 minutos"	I	11 min
Salir del Centro de Cómputo	Los alumnos desalojarán el centro de cómputo.	TC	1 min

I Trabajo Individual II Trabajo en parejas III Trabajo en tríos TC Toda la clase

ANEXO 6

Planer Clase de Comparación

Clase:	15	Tiempo:	90 minutos
Tema:	Comparación de Estructuras de Repetición		

Objetivo:	Ayudar al alumno a entender cuándo usar cada una de las estructuras de repetición.

Objetivo Especifico	Ejemplo	Dinámica	Tiempo
Planteamiento del problema			
Plantear al alumno un problema sencillo donde se implemente una estructura de repetición.	Profesor: "La semana pasada me reuní con unos amigos, y organizamos un juego. Así se juega, primero formamos un circulo, y cada quien iba dando un número primo, el que acertara en el número iba ganando puntos, el que después de 10 vueltas tenía más puntos era el campeón. El problema fue que teníamos que comprobar si el número era primo o no, y eso nos quitó tiempo. Entonces recordé que mis alumnos terminaron de ver estructuras de repetición y que ellos nos podían ayudar a hacer un programa que nos permita jugar sin perder tiempo"	TC	2 min
	Profesor: "¿Alguien me puede explicar qué es un número primo y como obtenerlo?"	TC	10 min
Diseño de la solución			
Los alumnos diseñan el pseudocódigo.	Profesor: "Se van a organizar en equipos de tres, recuerden que no deben de haber trabajado antes con sus compañeros de equipo. Van a diseñar el pseudocódigo de este programa. Tienen 30 minutos"	III	30 min
Revisión de propuestas			
Obtener una solución por cada estructura de repetición.	Profesor: "¿Qué equipos diseñaron el pseudocódigo con la estructura *for*, que equipos con *while*, y que equipos con *do-while*?". Seleccionar a un equipo por estructura para escribir su solución en el pizarrón.	TC	10 min
Corrección de las estructuras	Profesor: "Alguien ve algún error en el uso de las estructuras de estos tres pseudocódigos?"	TC	8 min
Comprobar que las tres propuestas sirven para solucionar el problema.	Profesor: "En el mismo equipo quiero que hagan una prueba de escritorio con cada una de estas tres propuestas y me digan si alguna falla para saber si es un número primo. Tienen 15 minutos"	III	15 min
Consolidación del tema			
Hacer reflexionar al alumno de que el programa es posible de resolver con cualquier estructura de repetición.	Profesor: "Como ya vimos, es posible diseñar el programa con cualquier estructura ¿Cuál creen que es la más conveniente y por qué?" Los alumnos comenzarán a debatir defendiendo sus posturas, mientras el maestro funge como	TC	15 min

I Trabajo Individual II Trabajo en parejas III Trabajo en tríos TC Toda la clase

BIBLIOGRAFÍA

Beltrán, J. (1993). *Procesos, estrategias y técnicas de aprendizaje*. Madrid: Síntesis.

Bruer, J. T. (1995). *Escuelas para pensar*. Barcelona: Paidós.

Bruner, J. (1980). *Investigaciones sobre el desarrollo cognitivo*. Madrid: Pablo del Rio.

Caparrós, A. (1980). *Los paradigmas en Psicología. Sus alternativas y sus crisis*. Barcelona: Horsori.

Carretero, M. (2009). *Constructivismo y educación*. Buenos Aires (Argentina): Paidós.

Chávez G. (2011). *En México los programadores están en pañales*. Abril 25, 2015, de Excelsior: http://www.excelsior.com.mx/2011/11/23/dinero/787127#view-1

Concepto de cognitivo - Definición y Concepto. (s.f.) Diciembre 16, 2017, de Concepto.de: http://concepto.de/cognitivo/#ixzz4bAF4bpBI

Definición de Metodología. (Octubre,19,2014). Sitio web: http://conceptodefinicion.de/metodologia/

Díaz Barriga Arceo, F., & Hernández Rojas, G. (2010). *Estrategias docentes para un aprendizaje significativo*. México, D.F.: McGraw Hill.

Flórez Ochoa, R. (2000). *Hacia una pedagogía del conocimiento*. México: McGraw-Hil.

Gayla S. Keesee. (s.f.). *Teaching and Learning Resources / Learning Theories*. Enero 17, 2016, de Teachinglearningresources.pbworks.com: http://teachinglearningresources.pbworks.com/w/page/19919565/Learning%20Theories

Gómez, A. (s.f.) *Teorías del aprendizaje, ¿Cómo se adquieren los conceptos?*. Junio 18, 2014, de *monografias.com*: http://www.monografias.com/trabajos5/teap/teap.shtml

Hernandis, M. (2016). *Innovar se pone caro: La escasez de programadores recalienta salarios*. Diciembre 29, 2016, de El Mundo:http://www.elmundo.es/economia/2016/05/04/5729c50ee5fdea525d8b46bb.html

Kahneman, Daniel. (2014). Pensar rápido, pensar despacio. España: Debate.

Moore, T. (1995). *Introducción a la teoría de la educación*. Madrid: Alianza.

Overview of Learning Theories | GSI Teaching & Resource Center. (s.f.). Enero 17, 2016, de Gsi.berkeley.edu: http://gsi.berkeley.edu/gsi-guide-contents/learning-theory-research/learning-overview/

Peñaloza Romero, E. (2004). *Fundamentos de la programación C/C++* / 4 Ed. México D.F.: Alfaomega.

Pérez, J. & Gardey, A. (2008), Definición de Metodología. Sitio web: Definición de metodología — Definicion.de

Psicología y Mente. (s.f) Abril 28, 2016, de Psicologiaymente.net: https://psicologiaymente.net

Quesada, R. (2008). *Cómo plantear la enseñanza estratégica*. México D.F.: Limusa.

Rodrigo, M., Arnay, J., & Carretero, M. (1997). *La construcción del conocimiento escolar*. Barcelona: Paidós.

Suárez Díaz, R. (2005). *La educación*. México: Trillas.