Facultad de Ciencias

Instituto de Matemática

Intervención basada en el aprendizaje cooperativo, para mejorar las tasas de aprobación en la asignatura MTM124 Algoritmos y programación

Memoria para optar al Título Profesional de Profesor de Educación Media en Matemática, Mención Computación

Paula Palma Alvarado

Loreto Zambrano Fuentes

Profesor Guía: Oscar Caneo Salinas

Chile, Valparaíso 2014

Agradecimientos

En primer lugar agradezco a mi familia, mis padres y mi hermano, por su amor y por su apoyo incondicional en cada etapa de mi vida.

También agradezco a nuestro profesor guía Oscar Caneo, por su paciencia, por su ayuda, por sus consejos y, por sobre todo, por creer en nuestro trabajo.

A mi amiga y compañera Loreto, por la confianza, por su ayuda, por su compromiso con nuestro trabajo, y por cada momento compartido. Y a su familia, por su apoyo y su cariño.

A mis amigos, por creer en mí y por estar siempre presentes, en especial a María Olga, "mi manis", por ser uno de mis pilares desde que nos conocimos, y a Yeimy, con quien compartimos el sueño de ser profesoras.

Y, por último, a todos los profesores que han participado en mi formación.

A todos ustedes, ¡muchas gracias!

Paula Palma Alvarado

Agradecimientos

Agradezco a mis padres Ivonne y Hugo por su apoyo incondicional en este proceso, por su cariño y por ayudarme a cumplir con esta meta, para ellos nuestro trabajo.

Agradezco también a mi hermano Hugo, primos, tíos, mi lela, mi pololo por siempre apoyarme en este largo camino recorrido, en los momentos que pensé que todo era oscuro sus palabras me ayudaron a salir adelante.

A mis amigas Marcela y Karen por acompañarme y apoyarme en mis años de estudio, para mí son las mejores. Gracias Fernanda por ser incondicional cada vez que pedíamos ayuda, gracias por tu entrega y cariño. Y a una gran amiga y compañera de trabajo, gracias a ti todo esto, Paula Palma para ti mi agradecimiento y admiración, también tu familia, gran compañía en los momentos de trabajo, gracias tía Mari por su apoyo infinito.

Profesor Oscar mis más sinceros agradecimientos por acompañarnos estos dos años, por apoyarnos, por su infinita paciencia, por entregarnos su confianza, gracias a usted este trabajo.

Y a mis queridos profesores del colegio y la Universidad, gracias por sus enseñanzas y su dedicación, para ustedes mi admiración.

Gracias a todas las personas que durante los seis años confiaron en mí y me entregaban su apoyo y cariño.

Y mis angelitos, mi tata, tía Iris, Mama nena, tío Lalo, Rodolfo, en su memoria va todo esto.

Loreto Zambrano Fuentes

Índice de Contenidos

Índice de Tablas

Índice de Imágenes

Índice de Gráficos

Resumen

Las asignaturas relacionadas al área de programación en la Carrera de Matemática de la Universidad de Valparaíso presentan una alta tasa de reprobación durante los últimos años, esto motivó investigar acerca de las posibles causas de estos resultados. Además, se buscaron metodologías de trabajo que permiten optimizar los aprendizajes de los estudiantes, y en base a ellas se diseñó una intervención que buscaba mejorar los resultados obtenidos en este tipo de asignatura.

La intervención fue aplicada en las sesiones de ayudantía de la sección 2 de la asignatura Algoritmos y programación (MTM 124) de la Carrera de Matemática, durante el segundo semestre del año 2012, y consta de dos partes:

- En la primera se trabajó la familiarización de los estudiantes con el entorno de programación del software científico-matemático Scilab.
- En la segunda parte se trabajó el proceso completo de programación, es decir: análisis del problema, diseño del algoritmo, codificación del algoritmo y prueba y depuración.

En la intervención se realizaron 10 sesiones, las cuales se basaron en la aplicación del aprendizaje cooperativo, para ello se formaron equipos de trabajo de 3 estudiantes cada uno, y se realizaron actividades que debían desarrollar con su equipo.

Al finalizar la intervención la tasa de aprobación en la asignatura no tuvo un aumento significativo, pero se logra distinguir que aquellos estudiantes con mayor asistencia a las sesiones de ayudantía lograron aprobar la asignatura, por lo que se propone repetir este tipo de intervención, manteniendo como base el aprendizaje cooperativo, pero cambiando el sistema de asistencia libre por asistencia obligatoria.

Introducción

A través de los años se ha constatado, en el ámbito de la educación superior, que la mayoría de los alumnos no llegan a desarrollar óptimamente ciertas competencias que tienen que ver con el área de la programación. Una de estas es la creación de programas, para lo cual se requiere no sólo contar con el tiempo adecuado para comprender cada etapa del proceso, sino también, con la práctica necesaria para abordar su ejecución.

En este sentido, en la Universidad de Valparaíso, específicamente en las asignaturas relacionadas con el área de programación en la Carrera de Matemática, es común que el docente aborde una gran cantidad de contenidos en un tiempo reducido, ya que su programa exige desarrollar una parte teórica y otra práctica, lo que conlleva a que se produzcan resultados insuficientes, traducidos a una alta reprobación.

Algoritmos y programación es una de esas asignaturas donde no sólo se requiere aprender el aspecto teórico, sino también ampliamente el práctico, trabajo indispensable para madurar el conocimiento y asimilarlo a la profesión, entendiendo que será una herramienta clave para desarrollar el pensamiento lógico.

Fue precisamente la alta reprobación en las asignaturas del área de Programación la que motivó nuestro trabajo de tesis, enfocándolo hacia la priorización de la práctica de sus contenidos. En síntesis, nuestra propuesta fue trabajar en un tiempo extra a la clase formal (en las sesiones de ayudantía) enfocándonos en utilizar de manera cooperativa el software utilizado en dicha asignatura.

Es así como se llevó a cabo una intervención pedagógica en la asignatura Algoritmos y programación, que duró diez sesiones de ayudantías, en las cuáles se realizaron actividades y trabajos prácticos de manera grupal, para que de este modo, el aprendizaje no fuera individual y unilateral (profesor-alumno), sino grupal y entendido como un proceso de cooperación (alumno/grupo-alumno).

MARCO TEÓRICO

Capítulo 1: Introducción

1.1 Carrera de Matemática de la Universidad de Valparaíso

La Facultad de Ciencias de la Universidad de Valparaíso, en su afán de cultivar y desarrollar las Ciencias Básicas, crea en el año 2001 la Carrera de Matemática, con dos opciones de titulación profesional: Licenciatura en Matemática y Pedagogía en Matemática con la mención Didáctica de la matemática y la mención Computación (Universidad de Valparaíso, Departamento de Matemática, s.f.).

La Carrera de Matemática tiene como objetivo formar profesionales en el área de la matemática. La especialización de Pedagogía se orienta a la formación de profesores de matemática para el nivel de educación media, con sólidos conocimientos tanto en la disciplina como en la pedagogía, y que sean capaces de apoyarse en herramientas tecnológicas para enseñar y transmitir esta ciencia (Universidad de Valparaíso, Departamento de Matemática, s.f.).

Las menciones de la especialización de Pedagogía entregan a los estudiantes una formación complementaria, la mención Didáctica de la Matemática prioriza potenciar los conocimientos y saberes fundamentales de la matemática escolar, tanto desde el punto de vista de la matemática misma como de su didáctica, y la mención Computación privilegia el uso y la integración en la práctica docente de las TIC (Tecnologías de la Información y Comunicación) para la enseñanza de la matemática (Universidad de Valparaíso, Departamento de Matemática, s.f.).

La especialización de Licenciatura entregará herramientas para formar profesionales con conocimientos fundamentales y específicos de las matemáticas de pregrado, para que puedan así profundizar esta ciencia en programas de posgrado en matemática (Universidad de Valparaíso, Departamento de Matemática). La carrera tiene como fin formar personas con fuertes conocimientos matemáticos tanto para la pedagogía como para la licenciatura, las que luego deben desenvolverse adecuadamente dentro del rubro en el que trabajará cumpliendo con el perfil

de egreso que la carrera presenta, el cual detalla los saberes que los titulados de la carrera deben manejar.

La Carrera de Matemática consta de un Plan Común formado por 18 asignaturas, las cuales se imparten en 4 semestres. Cursadas estas asignaturas, los estudiantes deben elegir una especialización, ya sea la de Pedagogía, que está compuesta de 24 asignaturas distribuidas en 6 semestres, o la de Licenciatura, compuesta por 15 asignaturas distribuidas en 4 semestres (anexo 1.1, pág. 85).

Al término de la Carrera, en la especialización de Pedagogía se obtiene el Grado de Licenciado en Educación y el Título de Profesor de Educación Media en Matemática con mención en Didáctica de la matemática o con mención en Computación, y en el caso de la especialización de Licenciatura se adquiere el Grado de Licenciado en Matemática. (Universidad de Valparaíso, Departamento de Matemática)

La malla curricular señalada anteriormente (anexo 1.1, pág. 85) estuvo vigente durante 10 años, luego de lo cual se realiza un cambio en ésta buscando que los estudiantes que ingresen a partir del año 2012 logren un óptimo desarrollo de sus habilidades, adecuándose a la realidad actual de su campo laboral.

Se producen cambios en las asignaturas, mejorando la organización de los contenidos, incorporando nuevos temas que permitan la mejor preparación de los estudiantes, y renombrando algunas asignaturas para que exista mayor coherencia entre el nombre y el contenido de la misma. Este proceso ocurre tanto en el plan común como en las especializaciones de Pedagogía y Licenciatura, además la mención Computación de Pedagogía pasa a ser mención en Informática educativa. Todo esto con el fin de que los futuros profesionales que egresen de la carrera estén mejor capacitados para enfrentar los desafíos que se le presentarán, ya que se observan debilidades en los egresados, tanto didácticas como pedagógicas, que se contraponen con su dominio en el área de la matemática. (Caamaño, Lewin, & Soto, 2012)

Dados los cambios realizados a la malla curricular el Plan común que tenía 18 asignaturas, actualmente tiene 19, la especialización de Pedagogía que estaba formada por 24 asignaturas actualmente consta de 28 asignaturas (ver anexos 1.1, pág. 85 y anexo 1.2, pág. 89).

Las asignaturas relacionadas con el área de computación del plan común de la antigua malla curricular eran: Computación (MLP 103), Programación y programación lógica (MLP 113), y Programación científica (MLP 213). Mientras que en la nueva malla curricular se encuentran: Computación (MTM 114), Algoritmos y programación (MTM 124), y Programación científica (MTM 224).

En la especialización de Pedagogía las asignaturas de la mención Computación eran: Didáctica y computación (MLP 307), Elementos de estructura y base de datos (MLP 317), Sistemas operativos (MLP 407), y Redes de computadores (MLP 417). Como se explicó anteriormente, esta mención cambió su nombre a Informática educativa, y sus asignaturas son: Didáctica y computación (MTME 315), Taller de software educativo (MTME 325), Informática educativa 1 (MTME 415), e Informática educativa 2 (MTME 425) (Universidad de Valparaíso, Departamento de Matemática, s.f.).

1.2 Área de Computación

El cambio de nombre de mención Computación a mención en Informática educativa se debe a la modificación de las asignaturas nombradas anteriormente, cuyo objetivo es "que profundicen en el uso didáctico de las TIC en el proceso de enseñanza y aprendizaje de la matemática, como también de herramientas metodológicas que permitan realizar una apropiada integración de la tecnología al aula" (Galleguillos, Pizarro, & Juyumaya, 2012).

En las asignaturas relacionadas al área de computación, tanto en la malla curricular antigua como en la nueva, se incluye la enseñanza de programación, en ellas se entrega a los estudiantes elementos básicos de la programación. Se espera que el estudiante pueda desenvolverse en un ambiente de programación y utilizar un software matemático para visualizar la matemática de una manera práctica, validando propiedades, descubriendo o reconstruyendo teoremas y realizando actividades que fomenten el razonamiento y la reflexión en esta ciencia, y las habilidades de resolución de problemas. Todo lo anterior permite al estudiante desarrollar su razonamiento científico (Universidad de Valparaíso, Carrera de Matemática, s.f.).

1.3 Perfil de egreso

La carrera prepara al estudiante para que el titulado domine los saberes específicos de matemática necesarios en su área:

- En Pedagogía se entregan herramientas para un buen desempeño profesional, tanto en lo referido al proceso de enseñanza y aprendizaje como en lo que respecta a la relación con colegas. Las menciones en Didáctica de la matemática y en Informática educativa entregan al estudiante una formación complementaria potenciando sus conocimientos, en la primera en los saberes de didáctica de la matemática y en la segunda, en el uso de tecnologías para integrarlas en sus prácticas docentes (Universidad de Valparaíso, Departamento de Matemática, s.f.).
- En Licenciatura en Matemática se pretende preparar un profesional con una óptima formación en matemática de pregrado que le permitirá luego continuar con estudios de magister en matemática o en áreas afines, además de preocuparse de la capacitación para desarrollar actividades iniciales de investigación en matemática y aplicación de éstas a otras áreas del conocimiento. (Universidad de Valparaíso, Departamento de Matemática, s.f.).

Para finalizar, hay que recalcar que la carrera de matemática en toda su formación, tanto en la especialización de Pedagogía como en la de Licenciatura, pretende lograr el desarrollo de fuertes habilidades matemáticas en los estudiantes, para que consigan desenvolverse de la mejor manera posible en su respectivo campo laboral.

Capítulo 2: Antecedentes del Problema

2.1 Enseñanza de Programación en el Plan Común

En la antigua malla curricular de la Carrera de Matemática existen dos asignaturas de primer año que incluyen programación en sus unidades temáticas, éstas son Computación (MLP 103) y Programación y Programación Lógica (MLP 113).

El primero consta de dos partes, una teórica en la que los estudiantes aprenden los conceptos básicos de computación, su historia y cómo se relaciona con otras disciplinas, y otra práctica en la que deben utilizar el sistema operativo Linux, además deben desarrollar la capacidad para diseñar algoritmos para resolver problemas de procesamiento de datos sencillos, y programarlos en lenguaje de programación C (Universidad de Valparaíso, Carrera de Matemática, s.f.).

En el segundo, los estudiantes profundizan sus conocimientos en programación aprendiendo la estructura general de un programa y el uso de subprogramas, arreglos, punteros, estructuras y archivos, logrando dar solución en lenguaje de programación a problemas cada vez más complejos (Universidad de Valparaíso, Carrera de Matemática, s.f.).

Al revisar los resultados de los estudiantes durante los últimos años en las asignaturas descritas anteriormente se puede observar una alta tasa de reprobación, lo que permite deducir que los estudiantes tienen dificultades para aprender a programar. A continuación se presentan las tablas de resultados de cada una de estas asignaturas:

MLP 103 - Computación			
Año	Número de estudiantes	Aprobados	Reprobados
2005	135	46,7 %	53,3 %
2006	125	52,0 %	48,0 %
2007	101	55,4 %	44,6 %
2008	81	66,7 %	33,3 %
2009	56	44,6 %	55.4 %
2010	68	41,2 %	58,8 %
Promedios	94	51,1 %	48,9 %

Tabla 1: Resultados asignatura Computación (MLP 103)

Como se puede observar en la tabla anterior, entre los años 2005 y 2010 el porcentaje de reprobación de la asignatura Computación no baja del 30%, cifra que no deja de ser menor. La más alta se percibe el año 2010 con un 58,8% de reprobación, con un promedio de 48,9% entre los años mencionados.

MLP 113 - Programación y programación lógica			
Año	Número de estudiantes	Aprobados	Reprobados
2005	89	60,7 %	39,3 %
2006	76	59,2 %	40,8 %
2007	68	33,8 %	66,2 %
2008	42	47,6 %	52,4 %
2009	41	63,4 %	36,6 %
2010	35	57,1 %	42,9 %
Promedios	59	53,6%	46,4 %

Tabla 2: Resultados asignatura Programación y programación lógica (MLP 113)

Como se observa en la tabla, en la asignatura de Programación y programación lógica (MLP 113) el porcentaje de reprobación no baja del 35% entre los años 2005 y 2010, el año 2007 llega al 66,2%. Esta cifra es la más alta en el porcentaje de reprobación de las asignaturas de primer año relacionadas al área de computación, y el promedio de reprobación para este rango de años es de 46,4%.

2.2 Metodología de Enseñanza

Se cree que uno de los factores importantes que ha influido en estos resultados, es la metodología de enseñanza y aprendizaje utilizada en las asignaturas mencionadas anteriormente, la que se ha sustentado fundamentalmente en clases expositivas y el trabajo individual de parte de los estudiantes.

En las clases, el profesor expone y transmite conocimientos a un grupo numeroso de estudiantes, los cuales son los receptores de la información y mantienen un rol mayoritariamente pasivo, pues cuando pueden hacer preguntas pocos lo hacen. Los contenidos de programación, entre los que se destacan algoritmos, pseudocódigos, lenguaje de programación, programas, subprogramas,

arreglos y estructuras, se presentan comenzando con una introducción al tema, luego el profesor expone los nuevos elementos a utilizar en los programas explicando cómo y cuándo utilizarlos, para esto se apoya en algunos ejemplos, posteriormente hace comparaciones de programas mostrando por qué uno funciona y el otro no, o por qué uno es más eficiente o práctico que el otro. A continuación da algunos ejercicios que los estudiantes desarrollan de manera individual y se finaliza con la revisión de los mismos. En estas últimas dos etapas el estudiante asume un rol más activo realizando preguntas sobre el desarrollo de los ejercicios.

El uso de la clase expositiva es beneficioso cuando se requiere enseñar una gran cantidad de información en poco tiempo y se trabaja con un grupo numeroso de estudiantes, también si la información es de difícil acceso, o si se encuentra en muchos textos con diferentes explicaciones, o cuando éstas son poco claras para el estudiante. En todos los casos anteriores, el profesor facilita la comprensión de los contenidos, además por ser experto en la materia puede motivar el interés de los estudiantes (Johnson, Johnson, & Smith, 1991).

Las asignaturas Computación (MLP 103) y Programación y programación lógica (MLP 113) cuentan con una larga lista de contenidos que deben ser tratados en un semestre, al ser asignaturas de primer año tienen gran cantidad de estudiantes por curso y los contenidos son nuevos para ellos, por lo que podría ser difícil de estudiar de manera independiente, esto último hace necesaria la intervención del profesor que debe presentar la materia de manera que sea comprensible para los estudiantes y despertar el interés de éstos por la asignatura. Dado lo anterior la clase expositiva parece ser el modelo más apropiado para las asignaturas revisadas.

Sin perjuicio de los beneficios de la clase expositiva señalados en el párrafo anterior, se debe considerar que este tipo de clase también posee algunas desventajas. Al estar centrada en el profesor y ser él quien presenta y explica los contenidos es necesario que los estudiantes presten el máximo de atención durante toda la clase, sin embargo estudios demuestran que los estudiantes comienzan poniendo atención, pero ésta decae a medida que avanza la clase teniendo un repunte en los últimos minutos, además el profesor determina el ritmo de la clase, siendo que no todos los estudiantes poseen los mismos conocimientos previos ni la misma capacidad de comprensión, de memoria y de tomar apuntes, por lo que no aprenderán de manera simultánea.

Como este tipo de clase permite abarcar gran cantidad de información en un corto plazo y los estudiantes mantienen un rol pasivo, puede que ellos se vean sobrepasados en sus capacidades de aprendizaje cuando se utilice esta estrategia.

La actitud y atención de los estudiantes es determinante para lograr el aprendizaje y éstas dependen en gran parte de su estado emocional, por ejemplo hay estudiantes que se cohíben por la presencia de algunos compañeros lo que lleva a que no hagan preguntas o que por sus preocupaciones personales no logran concentrarse, esto provocará que no logren comprender o que entiendan algo distinto de lo que el profesor esté explicando. Por último en las clases expositivas no hay predisposición de que el estudiante desarrolle el pensamiento, éste sólo recibe la información, por lo que no son las más eficaces cuando se quiere lograr un aprendizaje de alto nivel, es decir, cuando se espera que el estudiante analice, sintetice, integre o retenga el conocimiento por largo tiempo. (Johnson, Johnson, & Smith, 1991).

Las desventajas de la clase expositiva contribuyen a la alta reprobación de los estudiantes en las asignaturas en estudio, ya que el objetivo principal de este tipo de clase es transmitir información y no el desarrollar las habilidades de los estudiantes, lo que se contrapone con los objetivos de las asignaturas.

Además del tipo de clase que se utiliza en las asignaturas estudiadas es importante analizar otros factores que influyen en el proceso de enseñanza y aprendizaje de la programación.

2.3 Dificultades de Aprender a Programar

Para empezar se debe considerar que las carreras universitarias que incluyen programación en sus mallas curriculares generalmente lo hacen en el primer año de estudio, sin tener en cuenta que la mayoría de los estudiantes que ingresan a primer año son recién egresados de la enseñanza media de distintos establecimientos educacionales, por ende no han tenido las mismas experiencias previas y no están igualmente preparados para la educación superior. También se podrían estar enfrentando por primera vez a la experiencia de vivir fuera de casa, a tener que administrar sus finanzas, su tiempo personal y de estudio, a tener que establecer lazos personales y encontrarse en un entorno nuevo. En este contexto se dificulta el aprendizaje de los conocimientos básicos de la

programación claves para la comprensión de los siguientes contenidos de esta área (Jenkins, 2002) (Robins, Rountree, & Rountree, 2003).

En la antigua malla curricular la enseñanza de programación comenzaba en el primer semestre del primer año de carrera, lo que llevó a que los estudiantes vieran una mayor dificultad en aprender esta ciencia. En la nueva malla curricular la enseñanza de programación comienza en el segundo semestre de primer año, con lo que se espera lograr disminuir la complejidad del aprendizaje de programación.

Por otro lado, es importante destacar que la enseñanza de programación de computadores tiene como objetivo que los estudiantes aprendan metodologías y técnicas de resolución de problemas de procesamiento de datos. Al referirse a programación se alude a la creación de un programa que se presenta como solución a un problema planteado, para la práctica de este proceso primero se analiza el problema, una vez que se comprende lo que debe hacer el programa se pasa a la etapa de diseño del algoritmo, el que luego ha de ser codificado, es decir, se traspasa a lenguaje de programación, para finalmente probar el programa y hacer las correcciones y mejoras correspondientes. (Caneo, 2008)

En cada una de las etapas del proceso de creación de un programa es posible identificar dificultades que deben superar los estudiantes cuando se les presenta un problema de programación:

- En la etapa de análisis se ha encontrado que a los estudiantes les cuesta interpretar el contexto del problema, proceso necesario para identificar las condiciones y restricciones con las que se debe trabajar, además les resulta difícil discriminar la información relevante de la irrelevante y reconocer la información de entrada y de salida en el programa. (Caneo, 2008)

- La etapa de diseño del algoritmo es compleja para los estudiantes en cada uno de sus pasos, al comenzar se deben aplicar métodos heurísticos para encontrar posibles soluciones al problema planteado, a continuación corresponde descomponer dicho problema, identificando las acciones o procesos necesarios para ir resolviendo cada parte del mismo, luego se ha de organizar la secuencia de acciones siguiendo una estructura

lógica que permita resolver cada proceso que formará parte del algoritmo, después se procede a construir el algoritmo de manera coherente, finalmente se revisa y evalúa la eficiencia de éste, y si para un mismo problema se logra encontrar distintas soluciones se analizan para determinar cuál es más eficiente. (Caneo, 2008)

- Para la etapa de codificación del algoritmo los estudiantes deben utilizar el lenguaje de programación, siendo este distinto al que usan habitualmente se les dificulta el desarrollo de esta actividad y más aún si es primera vez que utilizan este tipo de lenguaje, pues es necesario el comprender y utilizar la sintaxis de las sentencias y estructuras semánticas con varias sentencias, identificar el tipo de variables que se requieren declarar en base a los probables valores que ellas pueden tomar, comprender y manejar estructuras de programación en funciones y estructuras de datos complejas, todo esto para organizar la codificación del programa conforme a una estructura modular y traducir el algoritmo a lenguaje de programación de manera correcta. (Caneo, 2008)

- En la última etapa de prueba y depuración del programa se reconocen dificultades en todo su desarrollo, tanto en crear procesos que permitan realizar pruebas funcionales, como en buscar e identificar los errores mediante el seguimiento de los procesos y las variables cuando el programa no funciona, además de la corrección en sus distintas representaciones. (Caneo, 2008)

Por lo expuesto en el párrafo anterior, es posible afirmar que en caso de haber un error en alguno de los pasos de cualquiera de las etapas el programa creado no entregará la solución al problema planteado, por lo que en la última etapa deberían corregirse las fallas, pero si el estudiante no domina los conocimientos requeridos en las etapas anteriores difícilmente logrará identificar el o los errores que hacen que el programa no funcione.

Hay contenidos en los que se requiere hacer un análisis en sus temas, realizar un estudio y comprender lo que se está aprendiendo, lo anterior se conoce como el estilo de aprendizaje profundo, ya que el estudiante profundiza el conocimiento para un tema en estudio, también está el estilo donde el estudiante, a diferencia del anterior, memoriza sin indagar más en el tema estudiado, esto se conoce como el estilo de aprendizaje superficial. El responsable de que los

estudiantes adopten el estilo de aprendizaje más adecuado para el tema en cuestión es el profesor. (Jenkins, 2002)

En programación el aprendizaje superficial sirve, por ejemplo, para aprender detalles de la sintaxis o la prioridad de los operadores, mientras que el aprendizaje profundo es útil para la comprensión del tipo de problemas estudiados en programación. A diferencia de otras asignaturas donde se utiliza el aprendizaje superficial primero para memorizar conceptos o fechas, y luego el profundo para comprender y analizar situaciones, para aprender a programar los dos estilos de aprendizaje deben ser utilizados de manera simultánea, ya que programar es una habilidad y no un conjunto de conocimientos. (Jenkins, 2002)

Cuando se programa, primero se debe encontrar la solución a un problema y esta traspasarla al lenguaje de programación, probarlo y corregirlo hasta que resulte sin errores. Para desarrollar la habilidad de programar lo más apropiado es el uso de ambos estilos descritos en el párrafo anterior, donde el estudiante primero profundice en el análisis del problema para crear el algoritmo solución del mismo, luego utilice el lenguaje aprendido, del que ha debido memorizar su semántica y su sintaxis, para traspasar el algoritmo al programa, y finalmente lo revise y estudie buscando mejorarlo.

Pese a lo anterior, en las asignaturas de programación el profesor presenta conceptos, atiende consultas de los estudiantes, da algunos ejemplos, en los que como ya conoce el problema y la solución suele explicarlos con toda claridad, luego plantea algunos problemas a modo de ejercicio, esperando que el estudiante detecte los patrones en los problemas planteados anteriormente, que logre asociar con las técnicas necesarias para desarrollar el ejercicio y que utilice la asociación patrón-técnica para resolver los nuevos problemas. Esto es lo que se conoce como aprendizaje por imitación, ya que se cree que el estudiante genera las habilidades necesarias tratando de imitar lo que el profesor hace. Cuando los problemas dados como ejemplo y los dados como ejercicios son similares, los estudiantes aprenden a resolverlos de forma mecánica, pero no aprenden a abordar un problema, cualquier cambio, por pequeño que sea, les genera dificultades, pues no tratan de analizar y descubrir como el profesor llegó a la solución, sino que la memorizan. Debido a lo anterior, el estudiante no logra generar las habilidades necesarias para crear un programa, incluso pueden establecer patrones incorrectos de resolución

que no son detectados durante el curso. (Villalobos, Casallas, & Marcos, 2006) (Satorre, Llorens, & Puchol, 1996)

La metodología de enseñanza descrita en el párrafo anterior también es utilizada en las asignaturas de programación de la Carrera de Matemática de la Universidad de Valparaíso, generando dificultades en el proceso de aprendizaje de los estudiantes, ya que el proceso educativo se centra en el lenguaje de programación y en la codificación del algoritmo, más que en su creación. Además, algunos estudiantes memorizan códigos de programas o parte de ellos sin comprenderlos y suponen que es la solución a cualquier problema de características similares, con lo que queda demostrado que no han desarrollado las habilidades necesarias para encontrar soluciones.

El profesor debería organizar el proceso educativo de tal manera que el estudiante adquiera la capacidad de descubrir y determinar la información relevante y necesaria para resolver un problema. Para este proceso se debe evitar que el estudiante entienda que la creación del algoritmo solución de un problema es un proceso mecánico, porque se aleja de la construcción y profundización del conocimiento.

Otros factores que influyen en el proceso de aprendizaje de la programación son la motivación y la frustración que puedan sentir los estudiantes al enfrentarse a esta ciencia, y a los nuevos conocimientos y habilidades que necesitan para tener éxito en su aprendizaje.

Cuando los estudiantes ingresan a una carrera y se encuentran con asignaturas como programación, que no tienen una relación clara o directa con su futura vida profesional, o no ven una ganancia real en dedicarse a este tipo de asignaturas, se vuelve difícil que logren la motivación necesaria para alcanzar con éxito los objetivos de la asignatura en cuestión, en este caso, aprender a programar. Por otra parte, se tiene que hay estudiantes que aun cuando entienden la teoría, logran escribir los algoritmos de los ejercicios dados en clases y a pesar de todo el tiempo que se dediquen a escribir un programa, la “suerte” y el “ingenio” parecen ser cruciales para poder crear un programa. El tiempo que se necesitará para resolver un problema es impredecible, el estudiante piensa que necesita suerte para crearlo y corregirlo, y que de lo contrario debe usar su ingenio, pues los problemas que enfrenta nunca fueron parte del curso. Por

estas razones es que se genera frustración en los estudiantes en el proceso de aprendizaje de la programación. (Villalobos, Casallas, & Marcos, 2006)

La programación es difícil de aprender, en especial cuando se desconoce por qué debe ser estudiada. Los factores nombrados anteriormente son determinantes en el proceso de enseñanza y aprendizaje de esta ciencia, por lo que deben ser considerados a la hora de planificar este proceso.

En otras palabras, la metodología que el profesor utiliza para enseñar a programar, siendo ésta el uso casi exclusivo de la clase expositiva, con el que espera que el estudiante aprenda tratando de imitar lo que él hace cuando resuelve un problema o presenta un programa, sin tener en cuenta las experiencias previas de los estudiantes, no es el mejor método de enseñanza cuando se quiere desarrollar una habilidad, como lo es el aprender a programar. Este tipo de enseñanza provoca que aumenten las dificultades que el estudiante enfrenta al aprender a crear un programa, ya que centra su atención en el lenguaje de programación y la etapa de codificación, descuidando las etapas de análisis, diseño y de prueba y depuración, por lo tanto no desarrolla su capacidad de resolución de problemas y su pensamiento lógico, que es el objetivo de estas asignaturas. Como los estudiantes desconocen dicho objetivo no ven la relación de la asignatura con su vida profesional, por lo que no tienen la motivación necesaria para aprender esta ciencia, por el contrario, cuando no logran resolver los problemas y crear el programa pueden creer que es cuestión de suerte descubrir la solución del problema, lo que genera el sentimiento de frustración en ellos.

Todo lo anterior influye en los resultados vistos al comienzo de esta sección en la tabla 1 (pág. 7) y en la tabla 2 (pág. 8), que muestran que las asignaturas Computación (MLP 103) y Programación y programación lógica (MLP 113) presentan un alto porcentaje de reprobación.

Capítulo 3: Planteamiento y Justificación del Problema

3.1 Planteamiento del Problema

En la nueva malla curricular de la carrera de matemática la primera asignatura del plan común que incluye programación en sus unidades temáticas, es Algoritmos y programación (MTM 124). De acuerdo a lo señalado en los antecedentes, especialmente las dificultades que deben superar los estudiantes para aprender a programar, consideramos necesario realizar una intervención pedagógica que contribuya a la metodología de enseñanza y aprendizaje utilizada por el profesor, y orientada a favorecer los aprendizajes de los estudiantes en esta asignatura.

3.2 Justificación del Problema

3.2.1 Aprendizaje

En el proceso de enseñanza y aprendizaje el objetivo es que el estudiante adquiera un nuevo conocimiento, y diversos autores se han referido a él en sus obras o escritos, cada uno desde su punto de vista.

Piaget se refiere al conocimiento como una construcción, la adquisición de éste es el resultado de la interacción entre la persona y el ambiente. Él explica que: el individuo realiza procesos de asimilación y acomodación, en el primero, se incorpora la información proveniente del medio a las estructuras o esquemas internos en las que pueda relacionarse la nueva información, y en el segundo, los esquemas se adaptan a las características de los nuevos conocimientos o informaciones procedentes del mundo exterior para hacer posible su asimilación. Es así, como relaciona y encaja un nuevo contenido dentro de los conocimientos ya existentes, esta capacidad es aprender, incorporar un conocimiento, y depende del nivel del desarrollo cognitivo. (Psicocode, s.f.)

Los postulados de Vygotsky señalan que el aprendizaje presupone un contexto social y un proceso de interacción. Él explica que el aprendizaje ocurre en un desarrollo cultural, que va

desde el exterior al interior del alumno, primero entre personas y después en el interior de la persona. (Psicocode, s.f.)

3.2.2 Constructivismo

Al relacionar el proceso donde el estudiante se hace cargo de la construcción de su conocimiento con las teorías de aprendizaje de Piaget y Vygotsky surge el Constructivismo Social. "El constructivismo trata de responder cómo se adquiere el conocimiento considerando a éste no como información, sino a capacidades, habilidades y hábitos; métodos, procedimientos, técnicas, actitudes, valores y convicciones." (Ferreiro, 2009)

El constructivismo explica que es importante el cómo se adquiere el nuevo contenido que se está aprendiendo y cómo se pasa de un estado inferior a un estado superior del conocimiento, también, hace énfasis en la actividad mental constructiva y auto constructiva del sujeto para lograr un aprendizaje, y que éste sea significativo, esta actividad debe ocurrir por el trabajo que desarrolle el profesor, donde cree situaciones de aprendizaje que le permitan a los alumnos una actividad mental, social y afectiva que favorezcan su desarrollo. (Ferreiro, 2009)

Vygotsky también plantea que el desarrollo del pensamiento y el lenguaje se produce después que en las actividades que el profesor realiza para la enseñanza, los alumnos han intentado, por sus propios medios, con la ayuda del profesor y de otros alumnos, pasar a su "zona de desarrollo próximo" de tal manera que pasen de "no saber" a "saber" o de "no saber hacer" a "saber hacer" (Ferreiro, 2009). Zona de desarrollo próximo, es lo que Vygotsky explica como la distancia de lo que el alumno puede aprender por si solo y lo que puede aprender con la ayuda del entorno. (Santiváñez, 2004)

Para el constructivismo la estrategia es: "el profesor diseña condiciones y el proceso de enseñanza aprendizaje tratando de incorporar los principios culturales de la familia y de la sociedad al mismo" (Santiváñez, 2004). Este diseño, el profesor lo prepara de tal manera que se realicen las tareas cognitivas en la familia, escuela, medio natural, o en cualquier otro ambiente. En este proceso, el profesor y el estudiante discuten y se comunican las expectativas y las tareas a realizar (Santiváñez, 2004).

Esta interacción está representada en la siguiente imagen:

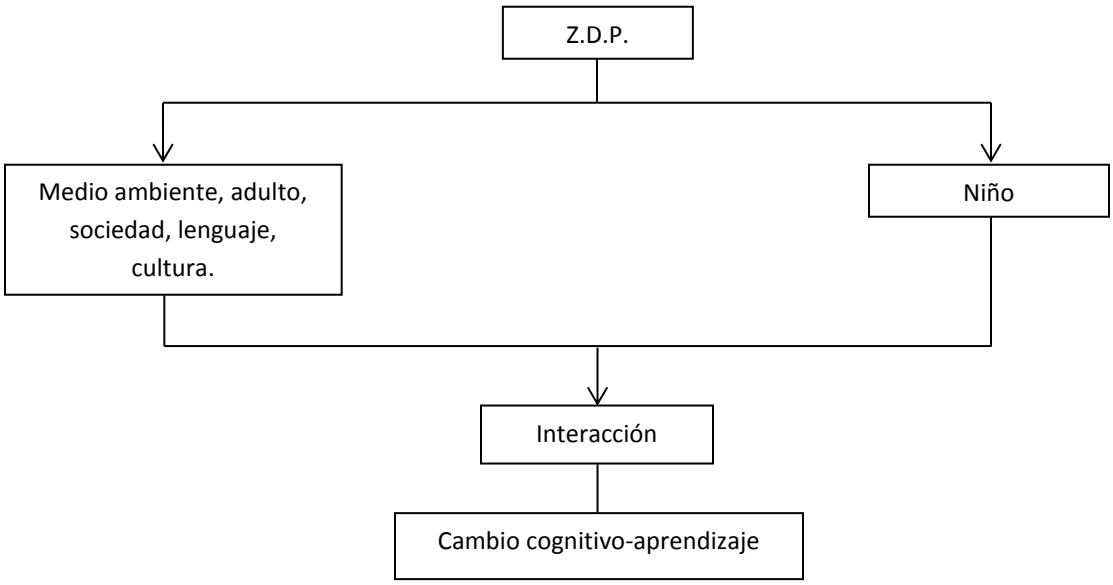


Imagen1: Representación esquemática de la aplicación del modelo de Vigotsky al aula (Santiváñez, 2004)

Con lo anterior expuesto, para desarrollar un proceso de enseñanza aprendizaje efectivo, donde se produzca el aprendizaje y éste no se considere un conocimiento como una información que se entrega al estudiante, se deben aplicar actividades donde se logre obtener la construcción de un nuevo conocimiento por la interacción del alumno con el medio, es decir, entre compañeros, siendo el profesor el facilitador de dicha interacción.

Para que esta interacción sea efectiva, el profesor puede realizar el proceso de enseñanza aprendizaje en el que existan actividades, trabajos, etc., donde ocurra cooperación entre estudiantes.

3.2.3 Aprendizaje Cooperativo

Cooperar es "obrar juntamente con otro u otros para un mismo fin" (Real Academia Española, s.f.). Si se hace una relación de esta definición con el proceso de enseñanza aprendizaje que se

realiza en el aula se tiene que los estudiantes pueden trabajar unos con otros con el fin de aprender. Esta relación es aplicada en el método de aprendizaje cooperativo.

Distintos autores han intentado definir este método, Goikoetxea & Gema, referenciando a Slavin, Johnson, & Johnson, exponen que "es un término genérico para referirse a numerosas técnicas de organizar y conducir la instrucción en el aula caracterizadas por el trabajo en grupos pequeños de alumnos heterogéneos para lograr objetivos comunes de aprendizaje" (Goikoetxea & Gema, 2002). Otra definición de aprendizaje cooperativo dice que "es el empleo didáctico de grupos reducidos en los que los alumnos trabajan juntos para maximizar su propio aprendizaje y el de los demás" (Johnson, Johnson, & Holubec, El Aprendizaje Coopertaivo en el Aula, 1999). También se ha explicado como "una serie de métodos de enseñanza en los que los alumnos trabajan en grupos pequeños para ayudarse a aprender entre ellos mismos" (Slavin, 1999).

Se puede concluir entonces que el aprendizaje cooperativo organiza el trabajo en el aula, de tal manera que los alumnos trabajen juntos y compartan sus conocimientos, lo que ayudará a que todos los integrantes del grupo aprendan.

Al trabajar y aplicar este método ya no se habla de una clase tradicional, donde el profesor es el encargado de transmitir conocimientos a los estudiantes y estos últimos son los que reciben la información, sino que se comienza a hablar de aulas cooperativas, ya que en ellas se desarrolla un trabajo cooperativo. "En las aulas cooperativas se espera que los alumnos se ayuden, que discutan con sus compañeros, que evalúen lo que saben los demás y los ayuden a superar sus problemas de comprensión" (Slavin, 1999). El trabajo cooperativo, entonces, trata de alcanzar que los alumnos se enseñen logrando así el aprendizaje, pero hay que tener presente que "este trabajo raramente sustituye la enseñanza del docente, pero reemplaza, sí, el trabajo, el estudio y la ejercitación individuales" (Slavin, 1999).

Existen una gran variedad de métodos de aprendizaje cooperativo que permiten que los alumnos logren el aprendizaje trabajando en forma grupal, estos diferentes métodos tienen su propia estrategia y forma de trabajo. Entre los más utilizados se encuentran: aprendizaje en equipo de alumnos, responsabilidad individual, trabajo en equipo - logro individual, torneo de juegos por equipo, y aprender juntos (Slavin, 1999). Sin embargo, aunque la forma de trabajo que utiliza cada método sea distinta, "todos los métodos de aprendizaje cooperativo comparten el principio

básico de que los alumnos deben trabajar juntos para aprender y son tan responsables del aprendizaje de sus compañeros como del propio" (Slavin, 1999).

Para que se desarrolle la cooperación entre los alumnos se deben cumplir cinco elementos: interdependencia positiva, responsabilidad individual, la interacción, habilidades interpersonales y grupales, evaluación grupal (Johnson, Johnson, & Holubec, El Aprendizaje Coopertaivo en el Aula, 1999).

1. Interdependencia positiva: los esfuerzos que realiza cada integrante del grupo benefician no solo a éste sino a todos los miembros del equipo.

2. Responsabilidad individual: Todos los miembros del equipo deben asumir la responsabilidad que le corresponda para realizar el trabajo.

3. La interacción: Los alumnos deben trabajar juntos para lograr los objetivos de las tareas, deben compartir ideas, conocimientos, conversaciones, y todo lo que sea necesario para realizar un buen trabajo.

4. Habilidades interpersonales y grupales: Los alumnos deben aprender de las prácticas interpersonales de los integrantes y así poder funcionar y complementarse como grupo para poder tener la responsabilidad de tomar decisiones, apoyarse en momentos difíciles, crear un clima de confianza y sentir la motivación de hacerlo.

5. Evaluación grupal: durante el desarrollo del trabajo grupal los alumnos deben evaluar el desempeño con el que están disponiendo personal y grupalmente, analizar como es el desempeño de los compañeros y tomar decisiones de lo que deben seguir o no haciendo para lograr un buen trabajo.

Estos cinco elementos en los que el alumno debe preocuparse tanto de su aprendizaje como el del compañero ayudan a que el trabajo cooperativo funcione, juntos logran que la labor en equipo sea satisfactoria tanto para el profesor como para los alumnos y se pueda alcanzar el tan anhelado aprendizaje.

3.2.4 Las relaciones interpersonales en el proceso de enseñanza y aprendizaje

Entre los participantes del proceso de enseñanza-aprendizaje existen dos tipos de relación posibles, la primera es la que se da entre el profesor y el alumno, y la segunda es la que se da entre alumnos (entre iguales). Actualmente en la organización de las aulas se privilegia el uso de la relación profesor-alumno, donde el profesor es el encargado de transmitir conocimientos a los alumnos y estos últimos son los que reciben la información. A este proceso se le conoce como educación tradicional. Si bien la relación alumno-alumno no está totalmente excluida de las aulas es menos aplicada que la relación profesor-alumno, pues cuando se utiliza la interacción entre dos o más alumnos se retoma después la estructura tradicional de la clase.

Dentro de las relaciones interpersonales dadas en el aula se encuentran las estudiadas bajo el criterio de consecución de objetivos que los autores Damon y Phelps clasifican en tres tipos distintos: la cooperación, la competición y la individualización (Serrano, González-Herrero, & Martínez-Herrero, 1997).

Se entiende una relación de cooperación aquella en la que un individuo puede alcanzar su objetivo siempre y cuando todos los demás logren los suyos. Distinto es en la competición, pues en este tipo de relación un sujeto consigue su objetivo si y sólo si los demás no logran el suyo. Finalmente, la individualización corresponde a una relación en la que los sujetos pueden alcanzar sus logros independientemente de si los demás logran los suyos (Serrano, González-Herrero, & Martínez-Herrero, 1997)

Diversas investigaciones sobre métodos de aprendizaje cooperativo, competitivo e individualista demuestran lo beneficioso que es utilizar el aprendizaje cooperativo. Los resultados de estos estudios permiten concluir que:

> "1. La cooperación conduce a manifestar un rendimiento más elevado por parte de todos los alumnos, mayor motivación para lograr un alto rendimiento, más tiempo dedicado a las tareas, un nivel superior de razonamiento y pensamiento crítico.
>
> 2. La cooperación da lugar a unas relaciones más positivas entre los alumnos, relaciones solidarias y comprometidas.

3. La cooperación produce mayor integración social, mejora la autoestima individual y refuerza la capacidad para enfrentar la adversidad y las tensiones." (Johnson, Johnson, & Holubec, 1999)

3.2.5 Programación colaborativa

Cuando se presenta una tarea de programación, ésta puede ser desarrollada por un programador de manera individual o por un grupo de programadores, los que deberán optar por trabajar de manera grupal o de manera colaborativa.

La programación grupal se refiere a programadores individuales que se dividen la tarea a desarrollar y coordinan su trabajo. En cambio, en programación colaborativa dos o más programadores trabajan conjuntamente sobre el mismo algoritmo o código. En la segunda, es necesario que la comunicación entre los programadores sea simple y confiable, para poder garantizar el intercambio de información entre ellos. (Rincón, Acurero, & Bracho, 2008)

Cuando la programación colaborativa es realizada por dos programadores se le llama programación par o en parejas, y la tarea se organiza de la siguiente manera: "mientras uno se enfoca en el código, el otro se encuentra continuamente revisando la calidad, haciendo preguntas, tratando de entender, y buscando alternativas que puedan mejorar lo realizado y evitar los defectos". Además, ambos programadores deben intercambiar roles cada cierto tiempo (Rincón, Acurero, & Bracho, 2008).

Existen variadas investigaciones sobre programación colaborativa, en las cuales se hace una comparación de ésta con la programación individual, tanto del proceso de creación del código y los resultados obtenidos, como en las relaciones entre los miembros del equipo de trabajo y sus procesos de aprendizaje. En general, los estudios realizados han encontrado que los programas creados de manera colaborativa son más efectivos, ya que la lluvia de ideas del equipo permite mejorar el diseño y la calidad del código, y la revisión continua del código ayuda a detectar y corregir los errores, teniendo una solución más confiable y con mayor correspondencia con el problema planteado. Además, en la programación individual se tiende a utilizar técnicas indisciplinadas ante situaciones de alto nivel de presión, lo que puede llevar a errores fatales, en

cambio, en la programación colaborativa es menos probable que todos los integrantes del equipo desvíen las técnicas, y esto permite asegurar la calidad del código. La programación colaborativa demostró también ser favorable para la relación de los miembros del equipo y el desarrollo de sus habilidades técnicas, ya que deben complementar sus conocimientos mientras realizan el programa, potenciando el desarrollo de las habilidades de los programadores y mejorando la comunicación entre ellos, lo que ha llevado a que disfruten más de su trabajo, y que aumenten la confianza entre los miembros del equipo y la motivación de ellos respecto a la tarea de programación. Todo lo anterior ayuda a mejorar la productividad de los programadores. Otro de los beneficios es que "cuando termina el proyecto más de una persona conoce los detalles del mismo, lo que evita la dependencia sobre una sola persona" (Rincón, Acurero, & Bracho, 2008).

3.2.6 El proceso de enseñanza y aprendizaje de programación

En resumen, la programación colaborativa permite mejorar las habilidades de resolución de problemas del equipo, el aprendizaje de cada uno de ellos, y la comunicación y la relación del equipo, resultando un mejor trabajo (Rincón, Acurero, & Bracho, 2008).

Para finalizar, es importante recordar que el profesor es el encargado de guiar el proceso de enseñanza aprendizaje, preocupándose de realizar una actividad eficiente y efectiva para que se produzca un aprendizaje significativo en el estudiante. Si se considera que este aprendizaje se construye en el estudiante por medio de la interacción social, una forma es que el profesor aplique en su metodología el trabajo cooperativo, generando situaciones donde los estudiantes en forma grupal resuelvan ejercicios, compartan conocimientos, evalúen y corrijan juntos y así construir en ellos un nuevo conocimiento por medio de la comunicación e interacción entre pares. En programación, lo más cercano es la programación colaborativa, ésta permite que los integrantes del equipo de trabajo compartan sus conocimientos y produzcan mejores programas, con soluciones más confiables.

PARTE EXPERIMENTAL

Capítulo 4: Objetivos

4.1 Objetivo General

Diseñar y probar una intervención pedagógica dirigida a favorecer y mejorar los aprendizajes de los estudiantes que cursan la asignatura Algoritmos y programación (MTM 124), incluida dentro de la nueva malla de la carrera de Matemática de la Universidad de Valparaíso.

4.2 Objetivos específicos

- Identificar las dificultades que presentan los estudiantes en el aprendizaje de los contenidos de la asignatura Algoritmos y programación.
- Identificar las habilidades de los estudiantes que faciliten el aprendizaje de los contenidos de la asignatura Algoritmos y programación.
- Identificar estrategias que favorezcan y mejoren el aprendizaje de los estudiantes.
- Diseñar actividades dirigidas a favorecer y mejorar los aprendizajes de los estudiantes.
- Probar las actividades diseñadas.

Capítulo 5: Metodología de Trabajo

5.1 Descripción de la Metodología

La intervención que se presenta, es una propuesta de trabajo aplicada en la sección 2 de la asignatura Algoritmos y programación (MTM 124) de la Carrera de Matemática de la Universidad de Valparaíso, esta intervención ha sido diseñada para los estudiantes que cursan esta asignatura, y está enfocada a mejorar sus aprendizajes, independiente de los resultados. Además se deja abierta la posibilidad de mejorar la presente propuesta y la opción de ser utilizada o no por el profesor del ramo en cuestión.

Esta intervención se basa en utilizar el aprendizaje cooperativo en las sesiones de ayudantía de la sección 2 de la asignatura Algoritmos y programación (MTM 124), estas son horas de clase extras a las de cátedra. En este tipo de asignatura ya se realizaban ayudantías, pero la metodología de trabajo consistía en reforzar lo visto por el profesor en las cátedras.

La intervención que se presenta muestra una nueva metodología y consta de dos partes:

- En la primera parte, se realizan actividades que permitan que los estudiantes se familiaricen con el software científico-matemático Scilab, el cual es un software libre, que se utilizará en la asignatura para programar, mientras en las cátedras el profesor enseña las primeras etapas de programación que son: análisis del problema y diseño del algoritmo.
- En la segunda parte, se trabaja en el proceso completo de programación, esto es cuando en las cátedras comienzan a trabajar en las etapas de codificación de los algoritmos, y prueba y depuración.

En las ayudantías se forman grupos de 3 estudiantes, estos grupos trabajarán durante todo el semestre de manera cooperativa, esto es realizando las actividades en conjunto, donde cada estudiante aporta con sus conocimientos, deben opinar y debatir sobre la actividad para lograr detectar y corregir los errores del trabajo desarrollado. Además, en cada ayudantía se realiza una actividad distinta y, de acuerdo a esta actividad, se da una tarea que será evaluada.

5.2 Aplicación de la metodología: Sesiones de ayudantías

A continuación se presenta la estructura detallada de las sesiones de ayudantías, especificando la etapa de inicio, desarrollo y cierre. Se explicitan además los objetivos y contenidos que se abordan en cada sesión. Para completar esto, se describe el análisis a priori con el fin de clarificar lo que se pretende en cada sesión.

Ayudantía 1

Fecha: 27 de agosto de 2012		
Objetivos Generales	**Objetivos Específicos**	**Contenidos**
Evaluar manejo de contenidos de los estudiantes, necesarios para el desarrollo del curso. Explicar la metodología de trabajo.	Realizar prueba de diagnóstico a los estudiantes para identificar manejo de los conocimientos básicos para el desarrollo del curso. Dar a conocer a los estudiantes la metodología a utilizar en las ayudantías.	Cálculo de porcentajes. Expresiones algebraicas. Algebra de Boole. Resolución de problemas.

Tabla 3: Objetivos y contenidos de la ayudantía 1

Inicio

Se da la bienvenida al curso y se explica que la actividad a realizar en esta sesión consiste en el desarrollo de la prueba de diagnóstico (anexo 2, pág. 93), en la cual se evaluarán los conocimientos básicos que se consideran de importancia para el éxito en la asignatura.

Seguido de esto, se entrega la prueba y se dan las indicaciones para su desarrollo.

Desarrollo

Cada estudiante desarrolla la evaluación diagnóstica de manera individual, la cual consta de cuatro ítems: cálculo de porcentajes, expresiones algebraicas, álgebra de Boole y resolución de problemas (anexo 2, pág. 93).

Cierre

Una vez finalizada la evaluación diagnóstica se procede a explicar la metodología de trabajo que se utilizará en las ayudantías. Esta metodología consiste en una primera parte de familiarización con el entorno de Scilab, que es un software científico-matemático gratuito que se utilizará en la asignatura, y una segunda parte de programación en SciNotes de Scilab, para lo cual se realizarán actividades desarrolladas de manera cooperativa. Se les explica a los estudiantes en que consiste el trabajo cooperativo y se les indica que deben formar grupos de tres personas los cuales deben mantener su conformación durante todo el semestre.

Se informa que se realizarán 10 sesiones de ayudantía, dentro de las cuales se efectuarán tareas evaluadas a medida que se avanza en los contenidos, éstas se desarrollarán con los grupos formados y se promediarán con los quices que se realizan en las cátedras, pudiendo eliminar dos de estas notas. Este promedio ponderará un 12% de la nota final del ramo.

Análisis a priori.

En esta sesión se realiza la prueba de diagnóstico con el objetivo de evaluar el manejo, por parte de los estudiantes, de los contenidos necesarios para el desarrollo del curso, esperando que reflejen en sus respuestas su dominio en cada uno de los ítems, respondiendo de forma clara, ordenada e individual. Con esto también se dispondrá de información importante para orientar el tratamiento de los contenidos en las sesiones de cátedra y en las sesiones de ayudantía.

Además se hace una explicación de la metodología a usar en las ayudantías, con el propósito que los estudiantes se involucren en éstas y se comprometan con su grupo y el trabajo a realizar.

Ayudantía 2

Fecha: 5 de septiembre de 2012.		
Objetivo General	**Objetivo Específico**	**Contenidos**
Familiarizarse con el entorno de trabajo de Scilab para el manejo de expresiones matemáticas.	Realizar operaciones matemáticas básicas (suma, resta, multiplicación, división, potencias y raíz cuadrada) simples y combinadas utilizando el software Scilab. Aplicar las operaciones aprendidas para resolver ejercicios.	Definición de software Scilab. Características del entorno de trabajo en Scilab Asignación de valor a una variable. Operaciones básicas: suma, resta, multiplicación, división, potencia y raíz cuadrada. Orden who. Obtención del valor de una variable. Expresiones matemáticas simples.

Tabla 4: Objetivos y contenidos de la ayudantía 2

Inicio

Se da inicio a la sesión indicando a los estudiantes que se utilizará el software Scilab en la asignatura y se continúa con la descripción de éste. Se explica que Scilab es un software científico matemático que se utiliza para realizar cálculos numéricos, programar y graficar, y que además permite resolver problemas de matemáticas aplicadas, física, ingeniería, procesamiento de señales, entre otros. Dentro de las características de este software se destaca que en programación usa un lenguaje simple; puede generar gráficos en dos y tres dimensiones; permite operaciones matriciales, con polinomios, funciones y resolución de sistemas de ecuaciones lineales y ecuaciones diferenciales; y posibilita la creación y definición de funciones propias.

Desarrollo

Se continúa con la explicación de:

- Asignación de valor a una variable utilizando el operador =. Ejemplo: a=2.

- El uso de punto y coma al final de una orden sirve para que Scilab no muestre la variable inmediatamente después de ingresada. Ejemplo: a=2;

- Para ingresar un número decimal se anotan utilizando el punto, no la coma. Ejemplo: b=3.2

Luego de la explicación se dan ejercicios a los estudiantes que permitan reforzar el contenido, y se pide que comenten los resultados con sus compañeros.

Ejercicio: Ingresar 5 variables (a, b, c, d, e) sin utilizar punto y coma al final, luego ingresar las mismas variables utilizando punto y coma al final. Al menos dos de las variables deben ser decimales.

Luego se enseña que la orden "who" lista las variables existentes, tanto las ingresadas como las que están predefinidas en Scilab, entre ellas las variables %e, %i, %pi.

Se prosigue con el desarrollo del siguiente ejercicio:

Ejercicio: Aplicar la orden who en la consola de Scilab.

Luego se explica que al escribir solamente la variable, Scilab entrega el valor guardado en ella y que el valor que guarda la variable es el último valor ingresado.

Ejemplo:

```
En Scilab
-->a=4
 a  =
    4.
 -->a=7
 a  =
    7.
 -->a
 a  =
   7.
```

Se asigna dos veces un valor a la variable a, la cual guarda el último valor ingresado.

Luego se enseñan las operaciones básicas y su simbología.

Operación	Símbolo
Suma	+
Resta	-
Multiplicación	*
División	/
Potencia	^
Raíz cuadrada	sqrt()
Número complejo	Parte entera + parte imaginaria * %i

Tabla 5: Operaciones básicas en Scilab

A continuación se da un ejercicio en el que los estudiantes utilizan la simbología enseñada.

Ejercicio: Utilizar las operaciones básicas para calcular el valor de las siguientes variables (puede utilizar los valores de las variables a, b, c, d que fueron ingresados en ejercicio anterior).

f=$a^2 + b^2 + c^2$

g=$\frac{\sqrt{c^2+d^2}}{a \cdot b}$

Luego de la explicación y la revisión del ejercicio se les indica a los estudiantes que deben realizar la tarea evaluada 1, ésta debe ser desarrollada en grupo.

Tarea evaluada 1

I. Ingrese valor a la variable x e y, luego encuentre el valor de las variables a, b, c, d, e.
   1. $b = x + y$
   2. $c = x \cdot y$
   3. $d = \frac{x}{y}$
   4. $e = b^2 + 2(c + d)^3$
   5. $a = \frac{x^3(2x^2+3.56x-5.21)}{\sqrt{3x+2.3}}$

II. Encontrar el área y perímetro de las siguientes circunferencias
   1. Circunferencia de radio 5.2
   2. Circunferencia de diámetro 18

Indicaciones de la tarea:
Debe copiar lo realizado en la consola de Scilab al block de notas y subirlo al Aula Virtual, el plazo de entrega es lunes 3 de septiembre a las 15:30 hrs.
La no entrega o copia del trabajo se evaluará con la nota mínima.

Cierre

Los estudiantes resuelven dudas, tanto de la materia vista en la sesión de ayudantía como de aspectos propios del trabajo cooperativo, esto último con el fin de ir mejorando y perfeccionando el trabajo y desempeño de los grupos. Al finalizar los estudiantes hacen entrega de su tarea.

Análisis a Priori

En el inicio de la sesión se entrega una descripción del software Scilab con el fin de que los estudiantes se familiaricen con él, ya que se utilizará en el desarrollo del curso.
Luego se hace una explicación sobre el ingreso de variables, donde el estudiante después de tomar atención realiza lo mismo en su computador, para ejercitar lo anterior se realiza una actividad.

Ejercicio: Ingresar 5 variables (a, b, c, d, e) probando lo antes indicado

```
En Scilab
-->a=5
 a  =
    5.
-->b=8;
-->c=3.4
 c  =
    3.4
-->d=8.1;
-->e=9
 e  =
    9.
```

Con este ejercicio se busca que el estudiante sea capaz de ingresar variables tanto enteras como decimales. Además que pueda diferenciar un método de ingreso de otro, es decir, que sepa la diferencia de utilizar o no el punto y coma al final de una asignación de valor.

Se continúa con la explicación y aplicación de la orden "who" en la consola para que los estudiantes comprueben que Scilab almacena las variables que ellos han ingresado y además variables predefinidas en el software.

Ejercicio: Aplicar la orden who en la consola de Scilab.

```
En Scilab
-->who
Sus variables son:
             p2                          a2                          r2
      p1
             a1                          r                             y
      x
             g                            f                            e
             d
             c                            b                           a
      WSCI
             home                  scinoteslib         modules_managerlib
      atomsguilib
             atomslib                matiolib          parameterslib
      simulated_annealinglib
             genetic_algorithmslib    umfpacklib               fft
      scicos_autolib
             scicos_utilslib           xcoslib               spreadsheetlib
      demo_toolslib
             development_toolslib       soundlib               texmacslib
      tclscilib
             m2scilib              maple2scilablib     compatibility_functilib
      statisticslib
```

```
              windows_toolslib              timelib               stringlib
      special_functionslib
              sparselib               signal_processinglib                 %z
      %s
              polynomialslib        overloadinglib          optimsimplexlib
      optimbaselib
              neldermeadlib          optimizationlib         interpolationlib
      inear_algebralib
              jvmlib                      output_streamlib
      iolib               integerlib
              dynamic_linklib             uitreelib                       guilib
      data_structureslib
              cacsdlib                graphic_exportlib              datatipslib
      graphicslib
              fileiolib          functionslib       elementary_functionslib
      differential_equationlib
              helptoolslib                       corelib                       PWD
      %tk
              %pvm                            MSDOS                         %F
      %T
              %nan                            %inf                            SCI
      SCIHOME
              TMPDIR                        %gui                          %fftw
      $
              %t                                    %f
      %eps                        %io
              %i                                    %e                          %pi
 usando       7992 elementos de    5000000.
 y         95 variables de       9231.
Sus variables globales son:
           %modalWarning                demolist              %driverName
%exportFileName
             %toolboxes           %toolboxes_dir
usando        616 elementos de        999.
 y          6 variables de        767.
```

Se sigue con la explicación de cómo Scilab entrega el valor de una variable y una aclaración sobre su ingreso, y se enseña la simbología para las operaciones matemáticas básicas. Estas explicaciones se realizan con el fin de que el estudiante se habitúe al trabajo en el software de Scilab.

Se prosigue la sesión con el siguiente ejercicio.

Ejercicio: Utilizar las operaciones básicas para calcular el valor de las siguientes variables (puede utilizar los valores de las variables a, b, c, d que fueron ingresados en el ejercicio anterior).

1) f=$a^2 + b^2 + c^2$

```
En Scilab
-->f=a^2+b^2+c^2
f  =
     100.56
```

2) g=$\frac{\sqrt{c^2+d^2}}{a \cdot b}$

```
En Scilab
-->g=sqrt(c^2+d^2)/(a*b)
g  =
     0.2196161
```

Estos ejercicios fueron diseñados para que el estudiante practique las operaciones básicas en Scilab, reconociendo las diferencias entre la notación matemática y la forma en que esta notación debe ser trabajada en el software. Por ejemplo la diferencia de los signos de multiplicación o potencia, o los signos de multiplicación que comúnmente se omiten, pero que en Scilab es imprescindible escribir para que pueda realizar el cálculo correspondiente.

Con la tarea evaluada 1 se busca que el estudiante demuestre que aprendió a realizar operaciones matemáticas en Scilab, utilizando los operadores correctos y los paréntesis necesarios para mantener el orden de la operación. Además, se evaluará si el estudiante es capaz de aplicar lo aprendido en un problema matemático, incluyendo la utilización del número π, que en Scilab viene predefinido y es representado por %pi. (Resolución de tarea en anexo 3.1, pág. 98).

Ayudantía 3

Fecha: 24 de septiembre de 2012.		
Objetivo General	**Objetivo Específico**	**Contenidos**
Familiarizarse con el entorno de Scilab en el manejo de operaciones matemáticas y funciones predefinidas.	Reaalizar operaciones con funciones predefinidas en Scilab.	Operaciones matemáticas en Scilab. Uso de funciones predefinidas en Scilab.

Tabla 6: Objetivos y contenidos de la ayudantía 3

Inicio

Se da inicio a la sesión comentando y recordando los contenidos vistos en la clase anterior para luego continuar con los contenidos a revisar en la ayudantía, el uso de funciones predefinidas en Scilab, se exponen a los estudiantes estas funciones, mostrando el nombre y la operación que realiza, estas funciones se encuentran disponibles en el aula virtual del ramo.

abs : valor absoluto	exp : función exponencial: e^x
ceil : parte entera superior	log : logaritmo natural
fix : redondeo hacia cero (igual a int)	log10 : logaritmo decimal
int : redondeo hacia cero (igual a fix)	log2 : logaritmo en base dos
floor : parte entera inferior	sin : seno
max : máximo	cos : coseno
min : mínimo	acos : arcocoseno
modulo : residuo entero	asin : arcoseno
rand : número aleatorio	tan : tangente
round : redondeo	atan : arcotangente
sqrt : raíz cuadrada	cotg : cotangente

Tabla 7: Funciones predefinidas

Desarrollo

Una vez ya presentadas estas funciones se les explica a los estudiantes que las funciones predefinidas son funciones que al ingresarlas a Scilab, el programa las reconoce y realiza la operación correspondiente, por ejemplo, al ingresar "abs" permite encontrar el valor absoluto de un número o "acos" permite obtener el arcoseno de un ángulo.

```
En Scilab

      -- >x=3;
      -->abs(x)
          ans  =
             3.
```

Se ingresa una variable $x = 3$ y luego se encuentra el valor absoluto escribiendo el nombre de la función abs(x), haciendo énfasis que se debe colocar la variable entre paréntesis.

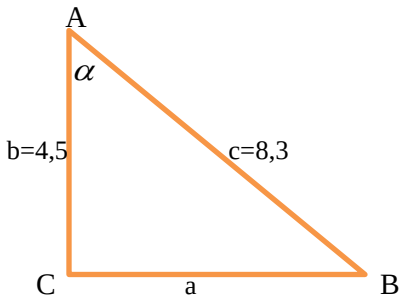
Cada estudiante cuenta con un computador, por lo que se pide que ingresen estas funciones para practicar su uso comentando con sus compañeros los resultados obtenidos.

A continuación se explica que los ángulos ingresados a Scilab son reconocidos como como radianes, no como grados, por lo que deben transformar el ángulo de radianes a grados, o viceversa (según corresponda).

Luego de que los estudiantes ingresen a Scilab distintas funciones se les presentan ejercicios en los que aparecen operaciones con las funciones anteriores, estos ejercicios deben ser desarrollados con sus respectivos grupos de trabajo para que comenten el desarrollo, compartan sus conocimientos y las estrategias que encuentra cada estudiante.

Ejercicios

1. Para el triángulo rectángulo encuentre el valor de "a" y " α "

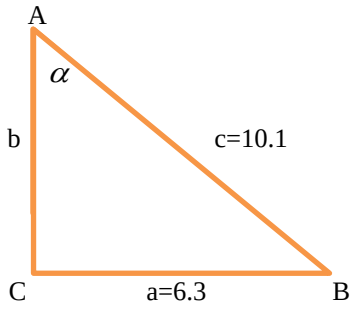


2. Evaluar $x = 2$ en la función $y = sen(x)\sqrt{x^2 + 2}$
   Luego evaluar para distintos valores de x.
3. $y = (arcos(x))^2 + |x|$ para $x = 2{,}5$, luego encontrar la parte entera.
4. Crear una función que tenga funciones predefinidas. Evaluar para cualquier valor de x, luego calcular valor absoluto y redondear.

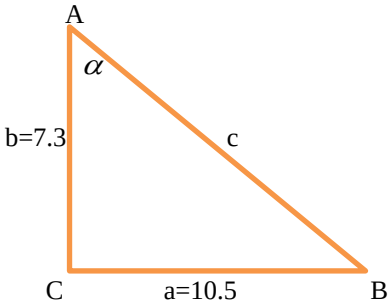
A continuación de que los estudiantes desarrollan los ejercicios practicando el ingreso de funciones, los estudiantes debe realizar la tarea evaluada 2, que debe ser desarrollada en los grupos de trabajo ya formados, esta actividad será evaluada con nota y debe ser terminada en la clase.

Tarea evaluada 2.

Ejercicio 1.
Para el triángulo rectángulo encuentre el valor de b y α



Ejercicio 2.
Para el triángulo rectángulo encuentre el valor de "c" y " α "



Ejercicio 3.
Crear una Función que contenga 4 funciones predefinidas. Evaluar la función para un x cualquiera. Encuentra la parte entera superior e inferior y redondeo.

Ejercicio 4.
Sea la función $y = |\sqrt{x^2 - |x-2|}|$
Evaluar para los valores $x = 1,\ x = 2, x = 0, x = -1, x = -2$.

Ejercicio 5
Encontrar las funciones trigonométricas para $\beta = 30°$.

Cierre

Los estudiantes finalizan el desarrollo de la tarea evaluada 2, resuelven dudas, corrigen posibles errores, comentan y revisan la resolución de cada ejercicio, también comentan como este trabajo grupal les favorece en el desarrollo de cada ejercicio y como les beneficia para la obtención de un nuevo conocimiento. Así, evalúan el trabajo cooperativo que desarrollan para realizar las tareas.

Análisis a priori.

En el inicio de la clase se realiza una conversación de los contenidos vistos en la clase anterior, donde se espera que los estudiantes comenten y compartan lo que entendieron sobre el ingreso de variables y operaciones con éstas en Scilab, si el contenido fue dominado por ellos y cuáles fueron sus debilidades.

Luego se comenta los contenidos a pasar en la clase, las funciones predefinidas de Scilab, se explica que estas funciones ya están definidas en Scilab y tan solo basta con ingresarlas para que esta realice la operación. Se muestra las funciones predefinidas por nombre y como se escriben en Scilab, se espera que los estudiantes recuerden que representan estas funciones.

Mientras realizan el ingreso de estas funciones, se espera que encuentren errores y descubran la forma correcta de ingresarlas, además que comenten y compartan con los compañeros sus resultados.

Los estudiantes deben descubrir que si ingresa una variable cualquiera y luego se hace la llamada a la función con la variable entre paréntesis, por ejemplo:

```
En Scilab
-->x=3.1
x  =
     3.1
 -->ceil(x)
 ans  =
          4.
 -->fix(x)
 ans  =
  3.
```

También se hace la llamada a la función directamente con el número a trabajar.

Ejemplo

```
En Scilab
-->ceil(3.1)
 ans  =
    4.
-->fix(3.1)
 ans  =
    3.
```

Los estudiantes deben hacer ingreso de todas las funciones, también las que involucran ángulos, como seno, coseno, etc, y deben transformar el ángulo dado a radianes ya que se les explica que el ingreso de los ángulos en Scilab es en radianes.

Los estudiantes deben descubrir entre ellos cual es la fórmula para transformar los grados en radianes.

Ejemplo

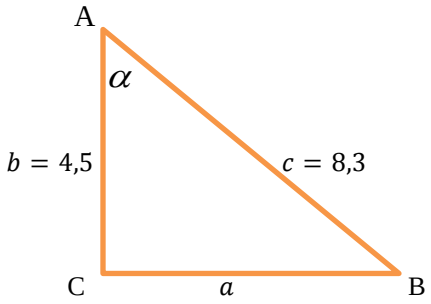
```
En Scilab
-->a=45
 a  =
    45.
-->ar=(45*3.14)/180
 ar  =
     0.785
 -->sin(ar)
 ans  =
     0.7068252
 -->cos(ar)
 ans  =
     0.7073883
 -->tan(ar)
 ans  =
     0.9992040
```

Así el estudiante debe hacer ingreso de todas las funciones e ir descubriendo su uso e implementación.

Luego de que los estudiantes hacen uso de las funciones en Scilab, se les entrega una serie de ejercicios para que se familiaricen con el uso de éstas.

Se espera que los estudiantes resuelvan los ejercicios de forma correcta en cooperación con los compañeros, donde deben compartir y comparar resultados.

Ejercicio 1. Para el triángulo rectángulo encuentre el valor de a y α



```
En Scilab
    -->b=4.5
 b  =
      4.5
 -->c=8.3
 c  =
      8.3
 -->a=sqrt(c^2-b^2)
 a  =
       6.9742383
```

Los estudiantes deben reconocer la fórmula para encontrar el valor de a en el triángulo y luego encontrar las funciones a utilizar en Scilab, en este caso se usa "sqrt" para encontrar la raíz cuadrada y c^2 para elevar al cuadrado la variable c.

```
-->alfa=asin(a/c)
       alfa  =
        0.9977804
```

Los estudiantes deben reconocer, trabajando cooperativamente con sus compañeros, la fórmula para encontrar el ángulo pedido, la fórmula a utilizar es α=arcoseno(a/c), ya que para encontrar el ángulo se debe usar "seno es igual al cateto opuesto dividido por la hipotenusa", a esto se le aplica la función inversa para encontrar α por lo que queda "α = arcoseno de cateto opuesto dividido por la hipotenusa". Luego deben encontrar las funciones a utilizar en Scilab. En este caso se usa la función *asin* para encontrar el arcoseno del ángulo.

Otra forma es la del coseno "coseno de alfa es igual al cateto adyacente dividido por la hipotenusa" y de igual manera le aplicamos la función inversa arcocoseno y queda "α = arcocoseno de cateto adyacente dividido por la hipotenusa", los estudiantes deben encontrar las funciones en Scilab y hacer el ingreso, en este caso se usa *acos* para el arcocoseno.

```
En Scilab
      -->alfa=acos(b/c)
       alfa  =
         0.9977804
```

Luego que los estudiantes encuentran el valor del ángulo pedido deben darse cuenta que el ángulo esta dado en radianes y deben transformarlo en grados con la fórmula:

Grados= (radianes * 180)/3,14.

```
      En Scilab
      -->alfa=0.9977804
       alfa  =
```

```
        0.9977804
        -->alfa_grados=(0.9977804*180)/3.14
        alfa_grados  =
        57.197603
```

Ejercicio 2. Evaluar $x = 2$ en la función $y = sen(x)\sqrt{x^2 + 2}$
Luego evaluar para los distintos valores de x

En este ejercicio los estudiantes deben reconocer el ingreso de las distintas funciones que se presentan, como seno, raíz cuadrada y elevar al cuadrado.

```
En Scilab
-->x=2
 x  =
  2.
-->y=sin(x)*sqrt(x^2+2)
y  =
   2.2273147
-->x=1
 x  =
    1.
 -->y=sin(x)*sqrt(x^2+2)
y  =
    1.4574705
-->x=3
x  =
    3.
-->y=sin(x)*sqrt(x^2+2)
 y  =
    0.4680421
-->x=10
 x  =
    10.
-->y=sin(x)*sqrt(x^2+2)
 y  =
  - 5.4943439
```

Ejercicio 3. $y = (arcos(x))^2 + |x|$ para x=2,5, luego encontrar parte entera

En este ejercicio los estudiantes deben reconocer como se realiza el ingreso de las funciones pedidas.

```
      En Scilab
     -->x=2.5
      x  =
      2.5
```

```
--
>y=(acos(x))^2*abs(x
)
 y  =
 - 6.1371496
 -->ceil(y)
 ans  =
  - 6.
 -->floor(y)
 ans  =
     - 7.
```

Ejercicio 4. Crear una función que tenga funciones predefinidas. Evaluar para cualquier valor de "x", luego calcular valor absoluto, redondear.

En este ejercicio los estudiantes deben hacer ingreso de funciones que contengan las funciones predefinidas por ejemplo raíz cuadrada (sqrt), seno (sin), tangente (tan), valor absoluto (abs), etc para evaluar si manejan el uso de las funciones predefinidas.

Luego que los estudiantes hayan realizado los ejercicios dados, en conjunto con sus compañeros, comenten y comparen resultados y se ayuden entre sí, se procede a desarrollar la tarea evaluada 2, la que se realiza en forma grupal, con el objetivo de que al realizar esta tarea compartan sus conocimientos obtenidos, que puedan encontrar errores y corregirlos, que compartan un mismo vocabulario y el trato entre ellos sea familiar. El desarrollo de esta tarea debe ser sin problema ya que en los ejercicios anteriores hacen uso de las funciones predefinidas (Resolución de tarea en anexo 3.2, pág. 100).

Ayudantía 4

Fecha: 24 de septiembre de 2012.		
Objetivo General	**Objetivo Específico**	**Contenidos**
Familiarizarse con el entorno de Scilab en el manejo de operaciones matemáticas con números complejos y polinomios.	Realizar operaciones con números complejos y polinomios en Scilab.	Operaciones con números complejos. Operaciones con polinomios.

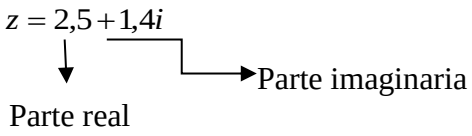
Tabla 8: objetivos y contenidos de la ayudantía 4

Inicio

Para iniciar la sesión se realiza una introducción sobre los números complejos explicando que un número complejo describe la suma de un número real y un número imaginario siendo este el producto entre un número real y la unidad imaginaria "i", se pide la cooperación de los estudiantes sobre la escritura de estos en el lenguaje algebraico, la parte real e imaginaria, escribiendo en la pizarra algunos ejemplos para que a continuación se ingresen en la consola de Scilab

Ejemplo.

Sea z número complejo.



Desarrollo

Para que Scilab reconozca a i como número imaginario debe ser ingresado como $\%i$ tal como se había mencionado en clases anteriores en el ingreso de variables en Scilab.

Para trabajar con números complejos se hace por medio de letras, por ejemplo "sean los números complejos $z1$, $z2$ y $z3$", en el enunciado anterior se asigna a $z1$, $z2$ y $z3$ números complejos distintos, en Scilab el ingreso de un número complejo se hace como una asignación de variable, a la letra que determinará el número complejo, se le agrega el signo = ingresando posteriormente su estructura.

La explicación del ingreso de un número complejo en Scilab se hace escribiendo en la pizarra que i se ingresa como $\%i$ y el ingreso de cualquier número complejo es:

$z = 2 + 4 * \%i$, donde z es la letra que se le asigna al número complejo, 2 es la parte real, 4 la parte imaginaria, el * indica que 4 se multiplica con la unidad imaginaria i. Luego se muestra el ingreso de distintos números complejos en la consola de Scilab.

```
En Scilab
-->z1=3.2+5.6*%i
 z1  =
  3.2 + 5.6i
```

```
-->z2=2+5*%i
 z2  =
     2. + 5.i
 -->z3=1.6+3*%i
z3  =
     1.6 + 3.i
```

A continuación de la explicación se les pide a los estudiantes que hagan ingreso a la consola de Scilab distintos números complejos comparando y comentando con sus compañeros, para luego proseguir con la explicación de la existencia de funciones que permiten obtener la parte imaginaria y la parte real, el conjugado, la magnitud y el argumento de números complejos ingresados.

Las funciones son real(z), imag(z), conj(z), abs(z), atan(imag(z)/rea(z)).

real(z)→ Parte real del número complejo z
imag(z)→ Parte imaginaria del número complejo z
conj(z)→ Conjugado de numero complejo z
abs(z) → Magnitud numero complejo z
atan(imag(z)/rea(z))→ Argumento número complejo z.

Tabla 9: Funciones

Se muestra a los estudiantes el ingreso y la función que realiza cada una para que trabajen con estas en los números complejos ya ingresados.

```
En Scilab.
-->z=2+5*%i
 z  =
    2. + 5.i
-->real(z)
 ans  =
   2.
 -->imag(z)
 ans  =
   5.
 -->zc=conj(z)
 zc  =
    2. - 5.i
 -->abs(z)
 ans  =
    5.3851648
-->atan(imag(z)/real(z))
 ans  =
     1.1902899
```

Los estudiantes deben familiarizarse con el ingreso de números complejos y el uso de las funciones antes mencionadas trabajando con el manejo de estos en la consola de Scilab, para continuar con la explicación de operaciones matemáticas básicas de estos números: suma, resta, multiplicación y división.

Para los números complejos se definen las operaciones matemáticas suma, resta, multiplicación y división, en Scilab estas operaciones se realizan primero ingresando el número complejo y luego realizando la operación con la letra que se le ha asignado al número o ingresando el número complejo directamente en la operación. Los estudiantes deben trabajar con estas operaciones, comparar y comentar, encontrar conjeturas sobre el ingreso de estas operaciones y comentar los resultados con sus compañeros.

```
En Scilab
-->z4=6+9.1*%i
 z4  =
     6. + 9.1i
 -->z5=8.4+3.6*%i
 z5  =
     8.4 + 3.6i
 -->z4+z5
 ans  =
     14.4 + 12.7i
 -->z4-z5
 ans  =
  - 2.4 + 5.5i
-->z4*z5
 ans  =
    17.64 + 98.04i
-->z4/z5
 ans  =
 0.9956897 + 0.6566092i
```

```
//la   operaciones   sumas   y   restas
ingresando el número directamente con la
operación

-->6+9.1*%i+8.4+3.6*%i
 ans  =
        14.4 + 12.7i
-->6+9.1*%i-8.4+3.6*%i
 ans  =
   - 2.4 + 12.7i
-->6+9.1*%i*8.4+3.6*%i
 ans  =
 6. + 80.04i
 -->6+9.1*%i/8.4+3.6*%i
```

```
ans  =
   5.+ 4.6833333i
```

Los estudiantes hacen uso de los números complejos en la consola de Scilab, realizando distintas operaciones y analizando conjetura encontradas, estos comentan y comparten con los compañeros el trabajo y descubrimiento que se produce acerca del uso de los números complejos, logrando un ambiente de cooperación.

Una vez que los estudiantes lograron el aprendizaje se continúa con el uso de polinomios en Scilab, estos pueden definirse de dos formas distintas, especificando las raíces y especificando los coeficientes.

1. Especificando raíces.

   p=poly([1,2,'x'])

Esta forma específica que los números 1 y 2 son las raíces del polinomio.

```
En Scilab
-->p=poly([1,2],'x')
p =
                2
    2 - 3x + x
```

El polinomio que se muestra como resultado es el polinomio que tiene como raíces 1 y 2.

2. Especificando coeficientes.

   p=poly([1,2],'x','c')

Esta forma específica los coeficientes del polinomio

```
En Scilab
-->p=poly([1,2],'x','c')
 p  =
     1 + 2x
```

El polinomio que se muestra como resultado tiene como coeficientes 1 y 2.

Para que los estudiantes comprendan el trabajo con los polinomios y manejen su uso se pide que ingresen distintos de estos en Scilab y comenten con sus compañeros, para poder proseguir con las operaciones suma, resta, multiplicación y división.

Los polinomios se pueden sumar, restar, multiplicar y dividir, primero ingresando el polinomio de cualquier forma antes enseñada y luego con la letra asignada se realiza la operación, los estudiantes deben ingresar polinomios y experimentar con sus operaciones y los resultados compartirlos con los compañeros para que se pueda lograr el aprendizaje.

Se ingresará el polinomio p1 y p2 para luego realizar las distintas operaciones

```
En Scilab
-->p1=poly([3,4,5],'x','c')
 p1  =
                2
    3 + 4x + 5x
 -->p2=poly([1,2,3],'x','c')
 p2  =
                2
    1 + 2x + 3x
-->p1+p2
 ans  =
                2
    4 + 6x + 8x
 -->p1*p2
ans  =
                  2     3     4
    3 + 10x + 22x + 22x + 15x
 -->p1/p2
ans  =
                 2
    3 + 4x + 5x
    -----------
                2
    1 + 2x + 3x
-->p1-p2
 ans  =
                2
    2 + 2x + 2x
```

El objetivo de la sesión es que el estudiante se familiarice con el entorno de Scilab realizando la distintas operaciones enseñadas, a continuación se les entrega a los estudiantes una serie de ejercicios de números complejos y polinomios para que practiquen el ingreso y operaciones, para que luego se les entregue una nueva tarea evaluada para realizar en la clase.

Tarea evaluada 3

Ejercicios 1.

Sea el número complejo z1=5.3+10i

a) Obtener de z1:

Parte real, imaginaria, conjugado, argumento, magnitud. (cada uno asignarle una variable distinta)

b) Ingresar un numero complejo z2 cualquiera

c) Obtener de z2:

Parte real, imaginaria, conjugado, argumento, magnitud. (a cada uno asignarle una variable distinta)

d) Obtener:

- Z1+z2
- Z1-z2
- Z2-z1
- Z1*z2
- Z1/z2
- Z2/z1
- $\frac{(z1 * z2) - z1}{z2} + (2{,}2 + 8i)$

Ejercicios 2.

a) Ingresar polinomio "p1" y "q1" en base a sus coeficientes (3 coeficientes), obtener sus raíces.

b) Ingresar polinomio "p2" y "q2" en base a sus raíces. (2 raíces).

c) Obtener:

- P1+q2
- P1-p2
- Q1-p1
- q1*p2
- P1/p2
- q2/p1
- p1+p2+q1+q2

Esta tarea la deben realizar con los grupos ya formados y subir los resultados al aula virtual del ramo al final de la clase.

Cierre

Los estudiantes terminan el desarrollo de la tarea evaluada, comentan los resultados obtenidos entre los integrantes del grupo, corrigen posible errores y comentan cual fue el nuevo conocimiento adquirido y como ayudo el trabajo grupal el lograr obtenerlo, a continuación envían los resultados al aula virtual de la asignatura para que luego sean revisados y evaluados.

Análisis a priori.

En el inicio de la clase se hace un pequeño recordatorio de la estructura de un número complejo donde se espera que los estudiantes recuerden que un número complejo está formado por una parte real y una parte imaginaria, se pretende que los estudiantes respondan al preguntar la estructura de éste.

Luego del recordatorio se les enseña el ingreso de un número complejo en la consola de Scilab

```
-->z=3.2+5.6*%i
 z  =
    3.2 + 5.6i
```

Los estudiantes no deben tener problema en el ingreso de éstos y comprender la estructura.

Se espera que no olviden que el uso de los símbolos $*$ y % para la multiplicación y % anteponerlo en la variable i.

Los estudiantes están cada uno con un computador por lo que deben hacer el ingreso de números complejos sin ningún problema, deben comentar y compartir sus resultados con los compañeros.

A continuación se le dan a conocer distintas funciones que permiten obtener la parte imaginaria, la parte real, el conjugado, la magnitud y el argumento del número complejo, el estudiante no debe tener problema en el uso de estas funciones ya que en clases anteriores ya se practicó el ingreso de funciones predefinidas. Solo deben recordar el uso correcto para el ingreso.

Luego se les explica que si se ingresan distintos números complejos, éstos se pueden sumar, restar, multiplicar y dividir, por ejemplo:

```
-->z1=3.2+5.6*%i
 Z1  =
     3.2 + 5.6i
-->z2=2+5*%i
 z2  =
     2. + 5.i
 -->z3=1.6+3*%i
 z3  =
     1.6 + 3.i
-->suma=z+z2+z3
 suma  =
     6.8 + 13.6i
```

```
-->resta=z+z2+z3
 resta  =
    6.8 + 13.6i
```

Este ingreso no debe tener mayor dificultad, ya que deben manejar el ingreso de números complejos y la estructura correcta de operaciones, ya visto en clases anteriores.

Es importante que el estudiante se dé cuenta que hay dos formas de sumar números complejos, asignándole valor a una variable y luego operando con el nombre de la variable o ingresando directamente los números con las operaciones correspondientes.

Luego de que el estudiante maneje sin problemas el ingreso de números complejos y operaciones entre éstos, se hace la explicación del ingreso de polinomios.

A continuación de la explicación el estudiante debe comprender el ingreso de polinomios, hay dos formas de ingresarlos, por coeficientes y por raíces, mediante la práctica el estudiante debe captar las diferencias y terminar por manejar su ingreso, luego para las operaciones suma, resta, multiplicación y división no debe existir dificultad ya que es igual a los números complejos.

Para continuar, los estudiantes deben hacer ingreso a Scilab de distintos polinomios para practicar su uso, especificando raíces y coeficientes y realizar operaciones con estos, al realizar esta actividad los estudiantes deben encontrar estrategias para el ingreso, practicar la forma que más les acomode, compartir y comentar con sus compañeros los resultados, ayudarse y corregir posible errores.

A continuación se les entrega la tarea evaluada 3, que deben realizar en los grupos de trabajo como en las tareas realizadas en las clases anteriores, los estudiantes deben compartir sus conocimientos obtenidos con los integrantes del grupo, realizar juntos los ejercicios y comentar como este trabajo ayuda a la obtención de un nuevo conocimiento con la ayuda de sus compañeros (Resolución de tarea en anexo 3.3, pág. 104).

Ayudantía 5

Fecha: 22 de octubre de 2012.		
Objetivo General	**Objetivo Específico**	**Contenidos**
Familiarizarse con el entorno de Scilab en el manejo de expresiones matemáticas. Comenzar a utilizar el lenguaje de programación en Scilab.	Ingresar y realizar operaciones básicas con vectores y matrices en la consola de Scilab. Utilizar el lenguaje de programación para implementar algoritmos selectivos en SciNotes de Scilab.	Ingreso y operaciones básicas de vectores y matrices en la consola de Scilab. Ejercicios de implementación para algoritmos de estructura selectiva.

Tabla 10: Objetivos y contenidos de la ayudantía 5

Inicio

Se comienza con la definición de vectores y matrices

Vectores:

$x = [3 \quad 2 \quad 1]$

$y = [7 \quad 6 \quad 5 \quad 4 \quad 3 \quad 2 \quad 1]$

En un vector se pueden almacenar varios valores dependiendo de su tamaño, en los ejemplos anteriores el vector x tiene tamaño 3, y el vector y tiene tamaño 7.

Matrices:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$B = \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix}$$

Una matriz también puede almacenar varios valores. Una matriz está compuesta por filas y columnas, en los ejemplos anteriores la matriz A tiene 2 filas y 3 columnas, por lo tanto se dice

que tiene tamaño $2x3$ y la matriz B tiene 3 filas y 3 columnas, por lo tanto se dice que su tamaño es $3x3$.

Desarrollo

Luego de la explicación anterior se procede a enseñar cómo crear un vector.

Para ingresar el vector $x = [3 \quad 2 \quad 1]$ en Scilab se puede hacer de 2 maneras

Escribir

1) x=[3 2 1]
2) x=[3,2,1]

En ambos casos en la consola de Scilab se obtendrá

```
x  =
    3.    2.    1.
```

A continuación se da un ejercicio para reforzar lo anterior.

Ejercicio
Ingrese un vector de tamaño 5, utilizando los dos métodos para el mismo vector.

Una vez terminado el ejercicio se explica cómo se puede mostrar un elemento de un vector.

Para que Scilab muestre un elemento de un vector se debe indicar su posición, por ejemplo, para el elemento de la primera posición del vector x se debe escribir x(1).

```
En Scilab
x(1)
 ans  =
     3.
```

Luego de la explicación se da un ejercicio para que los estudiantes practiquen lo anterior.

Ejercicio
Muestre el elemento de la posición 3 del vector ingresado anteriormente.
Muestre el elemento de la posición 5 del vector ingresado anteriormente.

Se continúa la sesión enseñando como se hace la creación de una matriz y se sigue con la realización de un ejercicio.

Para crear la matriz $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ existen 4 opciones:

1) A=[1,2,3;4,5,6]
2) A=[1 2 3 ; 4 5 6]
3) A=[1 2 3
   4 5 6]
4) A=[1,2,3
   4,5,6]

```
En Scilab
-->A=[1 2 3; 4 5 6]
 A  =
    1.    2.    3.
    4.    5.    6.
```

Ejercicio

Cree una matriz de tamaño 3x4 usando dos métodos distintos

Una vez terminado el ejercicio se sigue explicando cómo se muestra un elemento de una matriz y se realiza una actividad, lo mismo para mostrar una fila y una columna de una matriz.

Para mostrar un elemento de una matriz se debe anotar el nombre de la matriz y entre paréntesis indicar la fila y la columna en que se encuentra ubicado el elemento. Por ejemplo, para mostrar el elemento de la matriz A, de la fila 2, columna 3 debemos anotar *A(2,3)*.

```
En Scilab
-->A(2,3)
 ans  =
    6.
```

Ejercicio

Muestre el elemento de la fila 3, columna 2 de la matriz creada

Cómo mostrar una fila: para mostrar una fila de una matriz, al igual que para mostrar un elemento, se debe indicar el nombre de la matriz y la fila deseada, en el lugar en que se anotaba la columna se escriben dos puntos *(:)*. Por ejemplo si se desea mostrar la fila 1 de la matriz A se anota *A(1,:)*.

```
En Scilab
-->A(1,:)
 ans  =
    1.    2.    3.
```

Ejercicio

Muestre la fila 2 de la matriz creada

Cómo mostrar una columna: para mostrar una columna de una matriz, al igual que para mostrar un elemento, se debe indicar el nombre de la matriz y la columna deseada, en el lugar en que se anotaba la fila se escriben dos puntos *(:)*. Por ejemplo si se desea mostrar la columna 2 de la matriz A se anota *A(:,2)*.

```
En Scilab
-->A(:,2)
 ans  =
    2.
    5.
```

Ejercicio

Muestre la columna 4 de la matriz creada

Luego del desarrollo de los ejercicios anteriores se explica que con lo anterior se da por finalizado el proceso de familiarización con Scilab, para continuar las ayudantías realizando actividades que refuercen los contenidos vistos en las cátedras del curso.

A continuación de la aclaración anterior se da una tarea evaluada con nota en la que los estudiantes deben aplicar lo aprendido en cátedra y en ayudantías. La tarea debe realizarse con su grupo de trabajo y se aclararán dudas respecto a su desarrollo.

Tarea evaluada 4

Para realizar en SciNotes

1. Escriba un programa que permita ingresar la edad de una cierta persona, y muestre el mensaje "NIÑO" si la edad es hasta 10; el mensaje "PUBER" si la edad es mayor que 10 y menor o igual a 14; el mensaje "ADOLESCENTE" si la edad es mayor o igual que 15 y menor o igual a 18; el mensaje "JOVEN" si la edad es mayor o igual que 19 y menor o igual a 25; el mensaje "ADULTO" si la edad es mayor o igual que 26 y menor o igual a 65; y el mensaje "ANCIANO" si la edad es mayor que 65

   .
2. Escriba un programa que determine si tres enteros ingresados a, b y c son o no una terna pitagórica, es decir, si es posible que $a^2 + b^2 = c^2$ ó $b^2 + c^2 = a^2$ ó $a^2 + c^2 = b^2$

3. En una cierta empresa comercial, el sueldo mensual de los vendedores corresponde a la suma del sueldo base más una comisión que corresponde a un porcentaje de las ventas que el vendedor haya efectuado en el mes. El cálculo del porcentaje de las ventas se realiza según la siguiente tabla:
   - Si las ventas fueron inferiores a $4.000.000 el vendedor no recibe comisión
   - Si las ventas fueron iguales a $4.000.000 y menores de $10.000.000 la comisión es de un 3% de las ventas
   - Si las ventas fueron iguales a o superiores a $10.000.000 la comisión es de un 7% de las ventas

   Escriba un programa tal que ingresado el sueldo base de un trabajador y el monto de las ventas efectuadas en el mes, calcule y muestre el sueldo mensual del vendedor.

Cierre

Se atienden las consultas de los estudiantes para finalizar con la entrega de trabajos.

Análisis a Priori

Se inicia la sesión con la definición de vectores y matrices ya que los estudiantes no dominan este contenido y se hace necesario para el desarrollo de las siguientes ayudantías. Luego se continúa con la explicación de cómo crear un vector y cómo mostrar un elemento de un vector en Scilab y ejercitación de los mismos con el propósito de que el estudiante comprenda este contenido ya que será utilizado en sesiones de ayudantía y cátedras posteriores.

Antes de realizar los ejercicios se propone a los estudiantes verificar los ejemplos dados en su computador.

Desarrollo de los ejercicios.

Ingrese un vector de tamaño 5, utilizando los dos métodos para el mismo vector

```
En Scilab
-->v=[2 4 6 8 10]
 v  =
     2.    4.    6.    8.    10.
-->v=[2,4,6,8,10]
 v  =
     2.    4.    6.    8.    10.
```

Este ejercicio busca que el estudiante verifique que con ambos métodos de ingreso de un vector se obtiene el mismo resultado.

Muestre el elemento de la posición 3 del vector ingresado anteriormente

```
-->v(3)
 ans  =
     6.
```

Muestre el elemento de la posición 5 del vector ingresado anteriormente

```
-->v(5)
 ans  =
    10.
```

Con estos ejercicios se busca que el estudiante practique la forma de mostrar un elemento de un vector

Luego del desarrollo de los ejercicios se sigue con la explicación de cómo crear una matriz, como mostrar un elemento, una fila o una columna de ella y los ejercicios respectivos que permitan reforzar el contenido.

Cree una matriz de tamaño 3x4 usando dos métodos distintos

```
-->m=[1 3 5 7; 2 4 6 8; 3 6 9 12]
 m  =
     1.    3.    5.    7.
    2.    4.    6.    8.
    3.    6.    9.    12.
-->m=[1 3 5 7
-->2 4 6 8
-->3 6 9 12]
 m  =
```

```
     1.    3.    5.    7.
    2.    4.    6.    8.
    3.    6.    9.    12.
```

Con este ejercicio el estudiante puede probar distintos métodos de ingreso de una matriz

Muestre el elemento de la fila 3, columna 2 de la matriz creada

```
-->m(2,3)
 ans  =
    6.
```

Muestre la fila 2 de la matriz creada

```
-->m(2,:)
 ans  =
    2.    4.    6.    8.
```

Muestre la columna 4 de la matriz creada

```
-->m(:,4)
 ans  =
    7.
    8.
    12.
```

Con los 3 ejercicios anteriores se espera que el estudiante pueda diferenciar cómo mostrar un elemento, una fila o una columna de una matriz, teniendo claro que debe indicar el nombre de la matriz, la fila y columna deseada, respetando el orden correspondiente a estos tres factores.

Una vez terminado los ejercicios se explica que se da por finalizado el proceso de familiarización con Scilab, para continuar las ayudantías realizando actividades que refuercen los contenidos vistos en las cátedras del curso.

Se comienza con la primera actividad que consiste en crear programas con estructura selectiva.

Como los estudiantes no dominan la creación de códigos se espera que ellos hagan los algoritmos o pseudocódigos y luego los traspasen a lenguaje de programación.

Con la tarea evaluada 4 se busca que los estudiantes ejerciten la implementación de programas en Scilab, pues en las cátedras ya han comenzado a programar, sólo utilizando selectividad, lo que

en lenguaje de programación corresponde a las órdenes *if…then/else*. (Resolución de tarea en anexo 3.4, pág. 108).

Ayudantía 6

Fecha: 29 de octubre de 2012.		
Objetivo General	**Objetivo Específico**	**Contenidos**
Utilizar el lenguaje de programación en Scilab.	Utilizar el lenguaje de programación para implementar algoritmos repetitivos en SciNotes de Scilab.	Ejercicios de implementación para algoritmos de estructura repetitiva.

Tabla 11: Objetivos y contenidos de la ayudantía 6

Inicio

Al iniciar la sesión se recuerda lo que han visto en las cátedras de la asignatura, donde construyen algoritmos de estructura repetitiva en diagrama de flujo y el paso a pseudocódigo, el que se traspasa a SciNote, se realizan preguntas abierta y se construyen conceptos con las respuestas de los estudiantes.

A continuación se muestran programas desarrollados a modo de ejemplo haciendo distinción entre los ciclos *for* y *while*.

Ejemplos:

- Suma de los primeros 1000 enteros

While	For

s=0;
c=1;
while (c<=1000)
  s=s+c;
  c=c+1;
end
printf('La suma de los primeros 1000 enteros es %d',s);
	s=0;
	for c=1:1000
	  s=s+c;
	end
	printf('La suma de los primeros 1000 enteros es %i',s);

Este programa realiza las suma desde el número 1 al 1000, este dato se utiliza para crear la condición en el ciclo *while*, como en esta condición el término del ciclo (1000) es un número entero mayor a la variable *c*, en la que se inicia el ciclo, se puede utilizar el ciclo *for* para crear un programa que realice la misma tarea.

- División de dos números usando restas sucesivas

While

```
a=input('Divisor');
b=input('Dividendo');
r=a;
c=0;
while (r>=b)
 r=r-b;
 c=c+1;
end
printf('%i dividido en %i es %i ',a,b,c );
```

En este caso solo se puede utilizar el ciclo *while* para crear el programa, ya que en la condición del ciclo hay 2 variables. Para hacer un programa con el ciclo *for* es necesario conocer los valores en que se inicia y termina el ciclo.

Desarrollo

Luego del repaso y explicación se pide a los estudiantes que trabajen con la guía número dos entregada por el profesor en clases de la asignatura e ingresen los códigos en SciNotes.
Se les indica a los estudiantes que trabajen comentando el desarrollo con sus compañeros de grupo y consulten si existen dudas para que sean aclaradas.

Cierre

Una vez que los estudiantes han ingresado distintos programas en SciNotes se revisan algunos en la pizarra, comentando entre todos el desarrollo de éstos, haciendo énfasis en los errores que podrían cometer y las diferencias entre el uso de *while* y *for*.

Ejemplos

- Programa que suma desde 1 hasta un n ingresado por el usuario.

While	For

s=0;
n=input('Ingrese el número');
c=1;
while (c<n | c==n)
 s=s+c;
 c=c+1;
end
printf('La suma de todos los enteros desde 1 hasta %i es %i',n,s);
	s=0;
	n=input('Ingrese el número');
	for c=1:n
	 s=s+c;
	end
	printf('La suma de todos los enteros desde 1 hasta %i es %i',n,s);

- Programa que suma, cuenta y saca el promedio de 10 números ingresados por el usuario.

While	For

sp=0;
cp=0
c=1;
while (c<10 | c==10)
 n=input('ingrese número');
 if (modulo(n,2)==0) then
 sp=sp+n;
 cp=cp+1;
 end
 c=c+1;
end
printf('La suma de los pares es %i la cantidad es %i y el promedio es %f',sp,cp,sp/cp);
	sp=0;
	cp=0
	for c=1:10
	 n=input('ingrese número');
	 if (modulo(n,2)==0) then
	 sp=sp+n;
	 cp=cp+1;
	 end
	end
	printf('La suma de los pares es %i la cantidad es %i y el promedio es %f ',sp,cp,sp/cp);

- Programa que calcula y muestra en pantalla los divisores de un número ingresado por el usuario.

While	For

n=input('Número');
if n>0 then
c=1;
while (c<n |c==n)
 if modulo(n,c)==0 then
 printf('%i ',c);
 end
 c=c+1;
end
else
 printf('¡ERROR!');
end
	n=input('Número');
	if n>0 then
	for c=1:n
	 if modulo(n,c)==0 then
	 printf('%i ',c);
	 end
	end
	else
	 printf('¡ERROR!');
	end

n=input('Número');
if n>0 then
c=1;
	n=input('Número');
	if n>0 then
	for c=1:int(n/2)

```
l=int(n/2);
while (c<l |c==l)
    if modulo(n,c)==0 then
       printf('%i  ',c );
    end
 c=c+1;
end
printf('%i  ',n );
else
    printf('¡ERROR!');
end
```

```
    if modulo(n,c)==0 then
       printf('%i  ',c );
    end
end
printf('%i  ',n );
else
    printf('¡ERROR!');
end
```

Análisis a priori

En el inicio de la clase se hace un recordatorio de lo visto en cátedras donde construyen algoritmos de estructura repetitiva en diagrama de flujo y el paso a pseudocódigo, el que se traspasa a SciNote, con el propósito que logren comprender los ejemplos que se mostrarán y las diferencias entre un ciclo *while* y un ciclo *for*.

Entre los ejemplos hay programas que se pueden crear con ambos ciclos y otros que sólo se pueden crear utilizando el ciclo *while*. Es importante que el estudiante logre distinguir cuando utilizar un ciclo u otro, al lograr darse cuenta de cómo y cuándo usar tales ciclos deben compartir sus descubrimientos con sus compañeros, comentando y enseñando las técnicas usadas, es importante que se produzca una trabajo cooperativo entre los estudiantes, ya que hablaran un mismo lenguaje y así se facilitara el aprendizaje y la obtención de un nuevo conocimiento.

Una vez aclaradas las dudas de los estudiantes se les indica que deben seleccionar ejercicios de la guía 2 para crear los programas y así reforzar los contenidos vistos.

Se finaliza revisando los ejercicios en la pizarra con el fin de que compartan sus dudas y puedan ser aclaradas con la participación de los compañeros.

Ayudantía 7

Fecha: 5 de noviembre de 2012.		
Objetivo General	**Objetivo Específico**	**Contenidos**
Reforzar el manejo de expresiones matemáticas en Scilab. Utilizar el lenguaje de programación para implementar algoritmos en Scilab.	Recordar y reforzar el manejo de expresiones matemáticas en Scilab. Traspasar de algoritmo con estructura repetitiva y/o seudocódigo a código de programación de Scilab.	Ingreso de variables y operaciones básicas en la consola de Scilab. Ejercicios de implementación para algoritmos de estructura repetitiva.

Tabla 12: Objetivos y contenidos de la ayudantía 7

Inicio

Se entrega a los estudiantes un resumen de la materia vista en las ayudantías (anexo 5, pág. 141)

Desarrollo

Luego de la entrega de la guía y revisión de los estudiantes del documento entregado se sigue con el desarrollo de una tarea evaluada, la que consiste en crear programas de estructuras simples y repetitivas, deben desarrollarla en la sesión con los grupos ya formados de manera colaborativa y se da un plazo de dos días para su entrega.

Tarea evaluada 5

1) Haga un programa que permita ingresar las notas (5 notas) de un estudiante en la asignatura de matemática y luego muestre el promedio del estudiante, la nota más alta y la más baja que obtuvo.

2) Realice un programa que permita dar como salida la población de dos países (a y b), teniendo en cuenta para tal propósito lo siguiente:
   En el Primer Año el País a tiene menos población que el país b
   Las Tazas de crecimiento de los países a y b son de 6% y 3% anuales respectivamente.
   Se debe dar como salidas las poblaciones desde el segundo año hasta que la población de a exceda a la población de b, además la cantidad de años que transcurrieron para que esto sucediera.

3) Haga un programa que permita ingresar el promedio de notas de 10 estudiantes y muestre la cantidad de estudiante en las siguientes categorías
   - con promedio entre 60 y 70 “destacado”
   - con promedio entre 50 y 59 “bueno”
   - con promedio entre 40 y 49 “suficiente”
   - con promedio bajo 40 “insuficiente”

4) Haga un programa que multiplique los números del 1 al 10

5) Haga un programa que permita ingresar la base y el exponente, y luego calcule la potencia.

6) Haga un programa que permita calcular el factorial de un número

7) Haga un programa que determine el valor de la serie

$$s = 1 + (1 + x) + \frac{(2 + x)^2}{2!} + \frac{(3 + x)^3}{3!} + \frac{(4 + x)^4}{4!} + ... + \frac{(n + x)^n}{n!}$$

para un cierto *x* y *n* dados.

Cierre

Se finaliza la sesión atendiendo las dudas de los estudiantes sobre el desarrollo de la tarea evaluada.

Análisis a Priori

En el inicio de la sesión se hace entrega de un documento resumen de los contenidos vistos en ayudantías, ya que estos son utilizados en la creación de programas en SciNotes, se espera que los estudiantes revisen y recuerden dichos contenidos.

Luego se procede con el desarrollo de una tarea evaluada con nota que consiste en siete ejercicios donde deben crear programas de estructura selectiva y repetitiva.

Con el desarrollo de la tarea evaluada5 se espera que los estudiantes practiquen los contenidos vistos en cátedras, compartan y resuelvan con sus compañeros sus dudas. (Resolución de tarea en anexo 3.5, pág. 110).

Ayudantía 8

Fecha: 19 de noviembre de 2012.		
Objetivo General	**Objetivo Específico**	**Contenidos**
Utilizar el lenguaje de programación para implementar funciones en Scilab.	Traspasar de algoritmo con estructura repetitiva y/o pseudocódigo a código de programación Scilab utilizando la estructura de funciones.	Repaso de la estructura de funciones. Ejercicios de implementación de funciones en Scilab.

Tabla 13: objetivos y contenidos de la ayudantía 8

Inicio

Se comenta con los estudiantes lo visto en las ayudantías anteriores respecto a programas de estructura selectiva y repetitiva y se les indica que en esta ayudantía se utilizarán ejercicios ya resueltos, cambiando su estructura para incluir el uso de funciones.

A continuación se hace un repaso de la estructura de una función con ayuda de los estudiantes.

```
function [vd1,vd2,…,vdn]=nombre_funcion(vr1,vr2,…,vrm)
instrucciones
endfunction
```

[vd1,vd2,…,vdn] = → Son las variables que devuelve la función, la función puede no devolver variables por lo que no siempre se incluye esta parte en la función. Cuando devuelve alguna variable es necesario que la función también reciba por lo menos una de las mismas, debido a su estructura predefinida.

nombre_funcion → El nombre de la función que el programador define con las condiciones que no puede tener espacios ni tilde y no puede ser igual al nombre de alguna función predefinida por Scilab.

(vr1,vr2,…,vrm) → Son las variables que recibe la función, la función puede no recibir variables por lo que no siempre se incluye esta parte en la función. Cuando la función no recibe variables, debido a su estructura predefinida, no puede devolver una de éstas.

instrucciones → Corresponde al cuerpo de la función, en el cual se puede incluir operaciones con variables, instrucciones de repetición y/o selección, leer datos que ingrese el usuario, entre otras.

endfunction → es necesario escribir esta orden para el cierre de la función.

La llamada a una función: para hacer uso de una función se debe hacer la llamada a ésta en el programa principal.

`nombre_funcion (v1,v2,v3,…,vn)`	`a=nombre_funcion (v1,v2,v3,…,vn)`

nombre_funcion → Para hacer la llamada a la función se debe indicar el nombre de ésta.

(v1,v2,v3,…,vn) → Cuando la función recibe variables éstas deben incluirse en la llamada respetando el orden que tengan en la función.

a= → Cuando la función devuelve algún valor éste debe ser guardado en una variable.

Desarrollo

Luego del repaso se les indica que deben realizar los ejercicios 5, 6, 7 de la guía 1 y que comenten la resolución de estos con sus compañeros de grupos comparando los códigos.

Ejercicio 5: programa que calcule y muestre el área de un triángulo cualquiera.

Programa sin función

```
b=input('Ingrese base del triángulo: ')
h=input('Ingrese altura del triángulo: ')
area=(b*h)/2
printf('El área del triángulo es %f',area)
```

Programas con función

1. Función que no recibe ni devuelve variables

```
function area_triangulo
    b=input('Ingrese base del triángulo: ')
    h=input('Ingrese altura del triángulo: ')
    a=(b*h)/2
    printf('El área del triángulo es %f',a)
endfunction
area_triangulo
```

2. Función que recibe dos variables y devuelve una, la cual es guardada en una variable auxiliar.

```
function a=area_triangulo(b, h)
    a=(b*h)/2
endfunction
b=input('Ingrese base del triángulo: ')
h=input('Ingrese altura del triángulo: ')
area=area_triangulo(b,h)
printf('El área del triángulo es%f',area)
```

3. Función que recibe dos variables y devuelve una, la cual es usada directamente en la orden “printf”

4. Función que recibe dos variables, pero no devuelve.

```
function area_triangulo(b, h)
```

```
function a=area_triangulo(b, h)
    a=(b*h)/2
endfunction
b=input('Ingrese base del triángulo: ')
h=input('Ingrese altura del triángulo: ')
printf('El área del triángulo es
%f',area_triangulo(b,h))
```

```
    a=(b*h)/2
    printf('El área del triángulo es %f',a)
endfunction
b=input('Ingrese base del triángulo: ')
h=input('Ingrese altura del triángulo: ')
area_triangulo(b,h)
```

Ejercicio 6: Programa que determina si un número es par o impar.

Programa sin función

```
n=input('Número para determinar si es par o
impar ');
if modulo(n,2) ==0 then
    printf('\nEl número es par');
else
    printf('\nEl número es impar');
end
```

Programa con función

1. función que recibe una variable, pero no devuelve

```
function par_impar(n)
    if modulo(n,2) ==0 then
    printf('\nEl número es par');
else
    printf('\nEl número es impar');
end
endfunction
n=input('Número para determinar si es par
o impar: ');
par_impar(n)
```

2. función que recibe y devuelve una variable, esta última es guardada en una variable auxiliar

```
function parimpar=par_impar(n)
    if modulo(n,2) ==0 then
    parimpar='par'
else
    parimpar='impar'
end
endfunction
n=input('Número para determinar si es
par o impar: ');
parimpar=par_impar(n)
printf('\nEl número es %s',parimpar);
```

3. función que recibe y devuelve una variable, esta última es utilizada directamente en la orden "printf"

```
function parimpar=par_impar(n)
    if modulo(n,2) ==0 then
    parimpar='par'
else
    parimpar='impar'
end
endfunction
n=input('Número para determinar si es par
o impar: ');
printf('\nEl número es %s',par_impar(n));
```

4. función que no recibe ni devuelve variables

```
function
    n=input('Número para determinar si
es par o impar: ');
    if modulo(n,2)==0 then
    printf('\nEl número es par')
else
    printf('\nEl número es impar')
end
endfunction

par_impar
```

Ejercicio 7: Programa que permite ingresar dos números y muestra si fueron ingresados en orden ascendente o descendente.

Programa sin función

```
n1=input('Ingrese un número: ')
n2=input('Ingrese un número: ')
if n1==n2 then
    printf('\nLos números ingresados son iguales')
    else
    if n1<n2 then
        printf('\nLos números fueron ingresados en orden ascendente')
    else
        printf('\nLosnúmeros fueron ingresados en orden descendente')
    end
end
```

Programa con función

1. función que recibe dos variables, pero no devuelve

```
function orden(n1, n2)
    if n1==n2 then
    printf('\nLos números ingresados son iguales')
    else
    if n1<n2 then
        printf('\nLos números fueron ingresados en orden ascendente')
    else
        printf('\nLos números fueron ingresados en orden descendente')
    end
end
endfunction
num1=input('Ingrese un número: ')
num2=input('Ingrese un número: ')
orden(num1,num2)
```

2. función que recibe y devuelve una variable, esta última es guardada en una variable auxiliar

```
function ordnum=orden(n1, n2)
    if n1==n2 then
    ordnum='Los números ingresados son iguales'
    else
    if n1<n2 then
        ordnum='Los números fueron ingresados en orden ascendente'
    else
        ordnum='Los números fueron ingresados en orden descendente'
    end
end
endfunction
num1=input('Ingrese un número: ')
num2=input('Ingrese un número: ')
ord=orden(num1,num2)
printf('%s',ord)
```

3. función que recibe y devuelve una variable, esta última es utilizada directamente en la orden “printf”

```
function ordnum=orden(n1, n2)
    if n1==n2 then
    ordnum='Los números ingresados son iguales'
    else
    if n1<n2 then
        ordnum='Los números fueron ingresados en orden ascendente'
```

4. función que no recibe ni devuelve variables

```
function
    n1=input('Ingrese un número: ')
    n2=input('Ingrese un número: ')
    if n1==n2 then
    printf('\nLos números ingresados son iguales')
    else
    if n1<n2 then
        printf('\nLos números fueron
```

```
    else
        ordnum='Los números fueron
ingresados en orden descendente'
    end
end
endfunction

num1=input('Ingrese un número: ')
num2=input('Ingrese un número: ')
printf('%s',orden(num1,num2))
```

```
ingresados en orden ascendente')
    else
        printf('\nLos números fueron
ingresados en orden descendente')
    end
end
endfunction

orden
```

Cierre

Una vez que hayan realizado los ejercicios se revisan éstos en la pizarra, con ayuda de los estudiantes. Dando énfasis a la estructura de la función y las diferencias entre los códigos.

Análisis a priori

Se comenta lo visto en las sesiones anteriores esperando que los estudiantes recuerden lo estudiado de programación en Scilab para facilitar la actividad de esta ayudantía.

Se hace un repaso de la estructura de una función vista en la cátedra del ramo pues es un contenido reciente y es necesario que los estudiantes lo entiendan y sepan aplicar.

Los estudiantes deben resolver los ejercicios 5, 6, 7 de la guía 1, estos ejercicios ya han sido resueltos sin la implementación de función, ahora lo deben resolver incluyendo dicha implementación. Como hay cuatro casos para ésta y los estudiantes desarrollarán el ejercicio utilizando distintos métodos podrán compararlos encontrando las diferencias que existen en la estructura de la función y en el programa principal.

Se espera que los estudiantes al desarrollar los ejercicios dominen y entiendan el uso de funciones y su estructura. Al cierre de la sesión de ayudantía se hace una revisión de los ejercicios para detectar errores de los estudiantes y poder reforzar el contenido.

Ayudantía 9

Fecha: 26 de noviembre de 2012.		
Objetivo General	**Objetivo Específico**	**Contenidos**
Utilizar el lenguaje de programación para implementar funciones y operaciones sobre arreglos en Scilab.	Realizar operaciones sobre arreglos en las que se incluya el uso de la estructura de funciones.	Repaso de la definición de vector. Ejercicios de implementación de funciones y operaciones sobre vectores en Scilab.

Tabla 14: objetivos y contenidos de la ayudantía 9

Inicio

Se comienza la sesión repasando la definición de vector.

Se indica que un Vector es una variable que guarda más de un valor y que su declaración es b[x] → corresponde a un vector de nombre b y tamaño x.

Ejemplos:

b[3] → corresponde a un vector de nombre b y tamaño 3.
Para guardar valores en el vector se debe indicar el nombre del vector y la posición en la que se desea guardar el valor.
b(1)=15 → se guarda el valor 15 en la posición 1 del vector b
b(2)=4 → se guarda el valor 4 en la posición 2 del vector b
b(3)=8 → se guarda el valor 8 en la posición 3 del vector b
Para obtener valores se debe indicar el nombre del vector y la posición de la que se quiere conocer el valor.
b(1)
b(2)
b(3)

Desarrollo

Luego del recordatorio de lo visto en cátedras se procede a dar ejercicios en los que deben crear funciones en las que se utilizan vectores.

Ejercicios

- Crear una función que permita llenar un vector de tamaño n
- Crear una función que permita mostrar un vector de tamaño n
- Crear una función que devuelva el número más alto del vector
- Crear una función que devuelva el producto de los elementos del vector
- Hacer el programa principal que permita utilizar las funciones creadas.

Se dan las indicaciones para que los estudiantes trabajen en forma grupal, comentando y comparando resultado.

Cierre

Una vez que los estudiantes hayan realizado los ejercicios y compartido entre sus compañeros los resultados, se revisan en la pizarra las funciones creadas y se escriben con aportes de los estudiantes, anotando todos los posibles resultados, se corrigen y comentan diferencias y errores.

Análisis a priori

Al iniciar la clase se hace un repaso de la definición de vector vista en cátedras con el fin que tengan un mayor dominio del contenido ya que es un concepto nuevo para los estudiantes y así puedan realizar sin problemas las actividades que se darán en clases.

Luego del repaso se dan ejercicios en los que deben crear funciones que utilizan vectores con el fin de reforzar el concepto de función, su estructura y la operatoria de vectores.

Desarrollo de ejercicios:

- Crear una función que permita llenar un vector de tamaño n

```
function v=llena_vector(v, n)
    for i=1:n
        v(i)=input('Ingrese un entero:')
    end
    printf('\n')
endfunction
```

- Crear una función que permita mostrar un vector de tamaño n

```
function muestra_vector(v, n)
    for i=1:n
        printf('El valor de %i es %i \n',i,v(i))
    end
    printf('\n')
endfunction
```

- Crear una función que devuelva el número más alto del vector

```
function a=alto(v, n)
    a=v(1)
    for i=2:n
        if v(i)>a then
            a=v(i)
```

```
        end
    end
endfunction
```

- Crear una función que devuelva el producto de los elementos del vector

```
function m=multiplica(v, n)
    m=1
    for i=1:n
        m=m*v(i)
    end
endfunction
```

En estos ejercicios los estudiantes deben recordar y aplicar la estructura de función y el uso de vectores.

- Programa principal que permita utilizar las funciones creadas.

```
v=[]
n=input('Ingrese el tamaño del vector: ')
v=llena_vector(v,n)
muestra_vector
printf('\nEl número más alto del arreglo es: %d',alto(v,n))
printf('\nEl producto de los elementos del vector es: %d', multiplica(v,n))
```

En el programa principal deben aplicar la declaración del vector y la llamada a las funciones.
Se hace la revisión de los ejercicios en la pizarra con el fin de que los estudiantes comparen y comenten sus programas.

Ayudantía 10

Fecha: 03 de diciembre de 2012.		
Objetivo General	**Objetivo Específico**	**Contenidos**
Reforzar el contenido de programación en Scilab	Utilizar el lenguaje de programación para implementar un menú que incluya uso de funciones y vectores	Ejercicios de funciones y vectores incluidos en la estructura de un menú.

Tabla 15: objetivos y contenidos de la ayudantía 10.

Inicio

Al iniciar la sesión de ayudantía se les indica a los estudiantes que deben realizar una tarea evaluada con su grupo de trabajo, que se desarrollará de manera colaborativa, y que en ella aplicarán lo aprendido en cátedras y en ayudantías.

Tarea evaluada 6
Deben crear un programa que incluye el uso de menú y funciones.
En el programa principal se debe crear y llenar 3 vectores, además crear un menú que incluya todas las posibles llamadas a las demás funciones.
Las funciones que deben crear son:

1. Función que llene un vector de tamaño 5.
2. Función que muestre el vector de tamaño 5.
3. Función que cuente la cantidad de números pares almacenados en el vector.
4. Función que calcule el promedio de los valores almacenados en el vector.
5. Función que sume dos vectores.
6. Función que busque un número en el vector.
7. Función que cuente los números divisibles por 3 almacenados en el vector.
8. Función que cuente los números negativo de los tres vectores.
9. Función que muestre los divisores de los elementos del vector.
10. Función que cuente los números primos presentes en el vector.

Desarrollo

Los estudiantes realizan la tarea evaluada 6 en forma grupal y cooperativa, consultando las dudas que puedan surgir por posibles errores que ellos no sean capaces de detectar.

Cierre

Los estudiantes revisan sus tareas consultando las últimas dudas y entregando el programa finalizado.

El programa se debe subir al aula virtual del ramo.

Análisis a priori

En esta ayudantía sólo se da la indicación de realizar una tarea evaluada 6 en la que los estudiantes deben crear un programa que incluye el uso de menú y funciones, la que es desarrollada durante la sesión de ayudantía lo que les permite ir aclarando sus dudas respecto a por qué se producen determinados errores, la tarea la deben realizar en grupo para que comenten entre ellos el desarrollo de ésta. Se espera que con esta tarea los estudiantes logren dominar el uso de funciones y vectores. (Resolución de tarea en anexo 3.6, pág. 114).

Capítulo 6: Análisis de resultados

6.1 Resultados en la asignatura

El objetivo del trabajo realizado en las sesiones de ayudantía de la asignatura Algoritmos y programación (MTM 124) era lograr un aprendizaje significativo por parte de los estudiantes, para así mejorar la tasa de aprobación en las asignaturas de primer año relacionadas con el área de programación.

Para ello se planteó como primer objetivo identificar las dificultades y habilidades que presentan los estudiantes para el aprendizaje de los contenidos de la asignatura. Para esto al iniciar con las sesiones de ayudantías se realiza una prueba de diagnóstico para conocer cuál es el estado de inicio de los estudiantes, es decir, si se encuentran con los conocimientos necesarios para rendir de buena manera en la asignatura. Esta evaluación abarca contenidos de Cálculo, Expresiones algebraicas, Álgebra de Boole y Resolución de problemas, contenidos que son necesarios para tener éxito en la asignatura y para recibir sin problemas los nuevos contenidos y constituir nuevos conocimientos.

A continuación se presentan la estructura de la prueba y los resultados de 19 estudiantes que rindieron la prueba.

Ítem	Cantidad de preguntas
1- Cálculo	2
2- Expresiones algebraicas	15
3- Álgebra de Boole	4
4- Resolución de Problemas	7

Tabla 16: Estructura de prueba de diagnóstico

Ítem	No responde	Realiza desarrollo incorrecto	Responde correctamente
1	0,0 %	15,8 %	84,2 %
2	25,3 %	23,2 %	51,6 %
3	10,5 %	10,5 %	78,9 %
4	96,2 %	3,8 %	0,0 %

Tabla 17: Resultados de prueba de diagnóstico

Como se puede apreciar en la tabla 16 (pág. 72) la cantidad de preguntas no están distribuidas equitativamente para cada ítem, aun así se puede analizar y deducir cual o cuales son los contenidos que mejor manejan los estudiantes.

Al revisar los resultados presentados en la tabla 17 (pág.72) se puede observar que los contenidos que mejor manejan los estudiantes son Cálculo y Álgebra de Boole, tienen el mayor porcentaje de respuestas correctas con un 84,2 % y 78,9% respectivamente. Se puede deducir que los estudiantes presentan un mayor desarrollo de estos contenidos por lo que no debieran tener mayores dificultades al enfrentarse a estos contenidos.

En las preguntas de expresiones algebraicas los estudiantes tuvieron un 51,6 % de respuestas correctas, por lo que se puede concluir que el conocimiento y manejo que tienen de este contenido es de un nivel medio y necesitaran repasar cuando deban trabajar con él. Como este ítem contaba con la mayor cantidad de preguntas consideramos que las respuestas de los estudiantes podrían haberse visto afectadas por creer que les faltaría tiempo o por la ansiedad de pasar al siguiente ítem, lo que les podría haber llevado a responder rápido y cometer errores.

En el ítem de resolución de problemas es donde se puede deducir que los alumnos se presentan débiles, ya que se obtuvo un 0% de respuestas correctas, aunque se presentan 5 respuestas, su desarrollo es incorrecto, para este contenido se pueden presentar distintos obstáculos, tales como ser el último ítem y no poder responderlo por tiempo y/o cansancio, o simplemente no han desarrollado la capacidad de análisis que se necesita para resolver este tipo de problemas.

Además de las dificultades percibidas en la prueba de diagnóstico se consideraron las dificultades para aprender a programar que presentan los estudiantes de distintos lugares y de distintas carreras que incluyen programación en su malla curricular, y que fueron descritas en los antecedentes del problema.

También se observó a los estudiantes en el transcurso de las sesiones de ayudantías y en el desarrollo de los ejercicios y tareas evaluadas dadas y se logró identificar algunas de las dificultades y/ habilidades que estos presentaban para el aprendizaje de los contenidos de la asignatura, por ejemplo: poca comprensión del enunciado lo que provocaba que los algoritmos creados no fuesen solución al problema planteado; mecanización en el uso de los códigos, tratando de usar un mismo trozo de código para distintos problemas, los que no siempre estaban

relacionados. Además, fue posible distinguir a los estudiantes más aventajados, los que manejaban mejor el contenido que se estaba trabajando y que terminaron realizando la mayor parte de las tareas dadas, mientras los demás integrantes de los grupos hacían solo algunos aportes, perdiéndose el trabajo en equipo que se esperaba de ellos.

Otro de los objetivos de la investigación era identificar estrategias que favorecieran el aprendizaje y que pudiesen ser utilizadas para diseñar las actividades de las sesiones de ayudantía. Para lograr este objetivo se investigó, ente otras cosas, sobre los tipos de programación y el proceso de enseñanza y aprendizaje, lo que permitió concluir que los mejores resultados en programación se obtienen trabajando en equipo, y para aprender a programar lo más eficiente es el aprendizaje cooperativo. Es por ello que el diseño de las sesiones de ayudantía se basó en actividades donde se aplicara el aprendizaje cooperativo.

El último objetivo de esta investigación fue probar las actividades diseñadas, es decir, aplicar las sesiones de ayudantía diseñadas y revisar los resultados obtenidos por los estudiantes en la asignatura.

En los ejercicios y tareas evaluadas de las sesiones de ayudantía los estudiantes debían trabajar de manera cooperativa, es decir, trabajar en equipo para llegar a un resultado y obtener un aprendizaje. Como ya se explicó antes, las sesiones de ayudantía se dividieron en dos etapas:

- En la primera se realizó un proceso de familiarización con el software científico-matemático Scilab, en la cual los estudiantes conocían el entorno del programa y trabajaban con él, aplicando distintas técnicas, descubriendo qué, cuándo y cómo usar las herramientas que posteriormente utilizarían al momento de programar, tanto en las cátedras cómo en las sesiones de ayudantías, mientras en las cátedras se trabajaban las primeras etapas del proceso de programación: análisis del problema y diseño del algoritmo. En esta primera etapa se observó que los estudiantes lograron realizar los ejercicios y tareas con su equipo de trabajo de manera cooperativa, logrando el aprendizaje esperado.

- En la segunda etapa se trató el proceso completo de programación, es decir, análisis del problema, diseño del algoritmo, codificación del algoritmo, y prueba y depuración del código, esto fue al mismo tiempo que se trabajaba este proceso en las cátedras. Los

estudiantes fueron perdiendo el trabajo cooperativo en esta etapa, como ya se mencionó en un párrafo anterior, el estudiante más aventajado del grupo era el que realizaba la mayor parte de los ejercicios y de las tareas, con este sistema de trabajo no lograban el aprendizaje esperado, pues no todos los estudiantes eran capaces de crear un programa, cuando se les presentaba un problema trataban de copiar el código solución de un problema similar y no lograban identificar con claridad las partes del código que debían modificar.

Por otro lado, se analizó el rendimiento de cada estudiante en la asignatura, para esto se evalúo la situación final de cada uno al termino del semestre.

Al iniciar el semestre la sección 2 de la asignatura contaba con 30 estudiantes inscritos, de los cuales 6 no rindieron ningún certamen (evaluación de cátedras), por lo que se presume que estos estudiantes des-inscribieron la asignatura o se cambiaron de sección. Al no tener acceso a la lista oficial de la sección 2 de la asignatura al finalizar el semestre se ha optado por no considerar a estos estudiantes en el presente análisis de resultados. Por otra parte, en la nómina de estudiantes de la asignatura Algoritmos y programación (MTM 124), también hay estudiantes de Programación y programación lógica (MLP 113). Como los estudiantes de ambas asignaturas participaron del proceso de ayudantías todos están incluidos en el análisis que se presenta a continuación.

Al finalizar el semestre los resultados obtenidos por los estudiantes en la asignatura Algoritmos y programación (MTM 124) se mantienen cercanos al promedio de los resultados obtenidos entre los años 2005 y 2010 en las asignaturas Computación (MLP 103) y Programación y programación lógica (MLP 113), en la siguiente tabla se presenta esta situación:

Asignatura	Período	Promedio de número de estudiantes	Promedio de estudiantes aprobados	%	Promedio de estudiantes reprobados	%
Computación	2005-2010	94	49	51,1%	46	48,9%
Programación y programación lógica	2005-2010	59	31	53,6%	27	46,4%
Algoritmos y programación (Sección 2)	2012	24	13	54,2%	11	45,8%

Tabla 18: Resultados de asignaturas del área de Computación

Como se puede observar en la tabla anterior los resultados obtenidos por la sección 2 de la asignatura Algoritmos y programación no muestran un aumento considerable en la tasa de aprobación como se esperaba, por lo cual se analizarán las variables que pudieron afectar los resultados de la asignatura.

Una de las variables que afecta los resultados que obtienen los estudiantes en la asignatura es la asistencia, tanto a clases como a sesiones de ayudantía. Según el registro de asistencia a las sesiones de ayudantía que se presenta a continuación en la tabla 19 el porcentaje de asistencia no logra superar el 83,3%, llegando a su porcentaje más bajo en la última sesión con un 33,3% de asistencia.

Fecha	27 ago	03 sep	24 sep	01 oct	22 oct	29 oct	05 nov	19 nov	26 nov	03 dic
Asistencia	17	20	17	14	16	16	17	13	13	8
%	70,8	83,3	70,8	58,3	66,7	66,7	70,8	54,2	54,2	33,3

Tabla 19: Asistencia a ayudantías de la sección 2 de Algoritmos y programación (MTM 124), 2012

Este factor perjudicó el trabajo que se realizaba en las sesiones de ayudantía, ya que los grupos de trabajo estaban incompletos, incluso habían estudiantes que trabajaron de manera individual en algunas de las sesiones, lo que imposibilitaba cumplir el objetivo de trabajar cooperativamente para construir el aprendizaje.

En la tabla que se muestra a continuación se puede observar la relación entre el número de días asistidos a las sesiones de ayudantías y el número de estudiantes aprobados y reprobados, se puede observar que fueron 11 estudiantes los que reprobaron asistiendo entre 1 a 7 sesiones y los

13 estudiantes que aprobaron asistieron a 5 o más sesiones de ayudantías. Aquellos estudiantes que asistieron a 8 o más de las 10 sesiones de ayudantías realizadas aprobaron la asignatura.

Asistencia	Aprobados	Reprobados
1	0	2
2	0	0
3	0	1
4	0	1
5	2	3
6	2	0
7	1	4
8	3	0
9	4	0
10	1	0
Total	13	11

Tabla 20: Resultado final de estudiantes según su asistencia a ayudantías

Es importante mencionar que los 4 estudiantes que asistieron a 7 ayudantías y reprobaron la asignatura presentan nota mínima en por lo menos una de las evaluaciones de cátedra, lo que podría significar que el estudiante no la rindió. Además, no presentó todas las tareas evaluadas dadas en las sesiones de ayudantía, en estos casos también fueron evaluados con nota mínima.

Para visualizar mejor la situación presentada en la tabla **20** se muestra el siguiente gráfico:

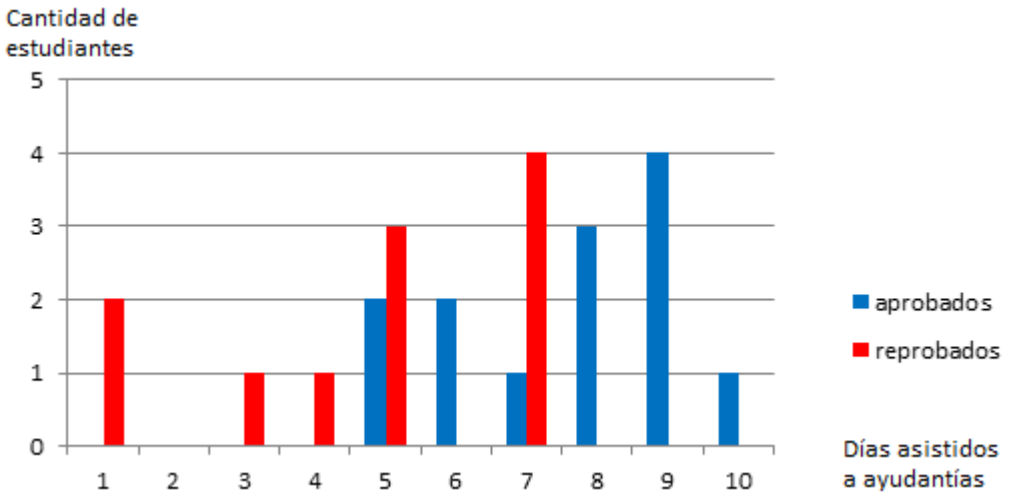


Gráfico 1: Resultado final de estudiantes según su asistencia a ayudantías

La asistencia a las sesiones de ayudantía era un factor determinante para tener éxito en la asignatura, ya que, como se explicó antes, en una primera parte se trabajó la familiarización con el software que se utiliza para programar en la asignatura, y en una segunda parte se trabajaba a la par con el profesor lo visto en las cátedras, además que en las sesiones de ayudantías se daban tareas evaluadas. La inasistencia a las sesiones de ayudantía, en cualquiera de las etapas, perjudicaba el trabajo en equipo y dificultaba el aprendizaje para los estudiantes, ya que no adquirían el aprendizaje necesario para rendir en la próxima ayudantía y en las cátedras.

No rendir todas las evaluaciones es otra variable que afecta los resultados obtenidos por los estudiantes en la asignatura, ya que los 11 estudiantes que reprobaron se encuentran en esta situación. Ninguno de ellos rindió todos sus certámenes, ni entregó todas las tareas dadas en sesiones de ayudantía; esto fue evaluado con nota mínima y provocó que cada uno de ellos reprobara la asignatura.

Creemos que la situación descrita en el párrafo anterior y la inasistencia a las sesiones de ayudantía se podrían explicar por falta de compromiso de los estudiantes con la asignatura. Además, como Algoritmos y programación se dicta en primer año algunos estudiantes no logran percibir la relación que tiene con la carrera que estudian, ni cómo podrían aplicar lo aprendido en un futuro profesional, al parecer desconocen que esta asignatura les ayuda en el desarrollo de su pensamiento lógico-matemático y en su capacidad de resolución de problemas, lo que explicaría que la descuiden faltando o llegando tarde a clases y/o a sesiones de ayudantía y, peor aún, no rindiendo las evaluaciones.

La mayoría de las sesiones de ayudantía de las distintas asignaturas son reforzamientos, donde la persona que las dicta resuelve dudas y ayuda a los estudiantes para tener un mejor rendimiento en la asignatura, y el estudiante decide si asiste a estas sesiones o no. Las sesiones de ayudantía dictadas en esta asignatura durante la investigación también tuvieron el sistema de asistencia libre, pero se informó a los estudiantes sobre las evaluaciones que se realizarían en las sesiones y que serían parte de la nota final de la asignatura, a lo que algunos de ellos no dieron importancia.

6.2 Limitaciones de la investigación

La asistencia a la asignatura era libre, tanto para las cátedras como para las ayudantías. Esta fue una de las limitaciones de la presente investigación, ya que si alguno de los estudiantes faltaba a la sesión, éste ya no estaría aplicando el trabajo cooperativo para aprender, y si faltaban 2 de los integrantes del grupo, quedaba un estudiante trabajando solo en las sesiones de ayudantía, por lo que tampoco aplicaba el aprendizaje cooperativo. La única forma de comprobar que los estudiantes utilizaran el aprendizaje cooperativo era que asistieran a las sesiones de ayudantía, la inasistencia dificulta la evaluación del sistema de trabajo aplicado en esta investigación.

Otra limitación que se presentó durante la investigación fue la dificultad para tener acceso a la lista oficial de la asignatura, ya que sólo se tuvo acceso a la lista inicial, lo que no permitió confirmar si es que algún estudiante des-inscribió la asignatura. Además no había claridad de si los estudiantes pertenecían a la asignatura Algoritmos y programación (MTM 124) o a Programación y programación lógica (MLP 113), ya que la carrera optó por juntar a los estudiantes de ambas asignaturas en el período de la investigación. Esto afecta para el análisis de resultados, en el cual se optó por no considerar a aquellos estudiantes que no rindieron ningún certamen, suponiendo que se habían cambiado de sección o que des-inscribieron la asignatura.

La metodología de trabajo propuesta en esta investigación sólo fue aplicada durante un semestre. Esta también es una limitación a la investigación, ya que si la nueva metodología se aplicara por más períodos seguidos sería posible comparar la variación de los resultados de los estudiantes en la asignatura, revisando si se produce un incremento en la tasa de aprobación con la aplicación de la metodología de trabajo propuesta. Además, como la asignatura Algoritmos y programación (MTM 124) pertenece a la nueva malla curricular, no fue posible comparar con resultados de la misma asignatura, debiendo comparar los resultados con las asignaturas afines de la antigua malla curricular.

6.3 Propuestas de mejora

Con el propósito de recoger más información sobre la efectividad de la metodología de trabajo propuesta en esta investigación, se sugiere continuar aplicando esta metodología, incluyendo las mejoras que se proponen a continuación.

En primer lugar, debido a las dificultades que se presentan al incluir programación en una asignatura de primer año, se propone aumentar la cantidad de horas de sesiones de ayudantía, para poder trabajar en la familiarización de los estudiantes con el entorno de programación del software a utilizar en la asignatura, y al mismo tiempo ir reforzando los contenidos vistos en las cátedras. Además, el tener más horas para las sesiones de ayudantía permitiría realizar más ejercicios en el que se aplique el proceso completo de programación.

Para mejorar el problema de la inasistencia de los estudiantes a las sesiones de ayudantía se propone exigir un porcentaje mínimo de asistencia para aprobar la asignatura, el cual podría ser de un 70%. El asegurar la asistencia de los estudiantes a las sesiones de ayudantía permite una mayor ejercitación de los estudiantes y confirmar que aplican el aprendizaje cooperativo, lo que ayudaría a mejorar sus resultados en la asignatura. Si al exigir un porcentaje mínimo de asistencia aumenta el porcentaje de aprobación se comprobaría que este es uno de los factores claves para lograr aprobar la asignatura.

Y, por último, para aumentar el compromiso de los estudiantes con la asignatura es importante que ellos comprendan la relación que existe entre programación y matemática, y cómo les ayuda aprender a programar en la resolución de problemas. Esto se puede explicar en las cátedras y en las sesiones de ayudantías, varias veces durante el semestre, con el fin de reforzar la idea y que los estudiantes se interesen en el aprendizaje de la programación.

Capítulo 7: Conclusión

En las asignaturas de mayor complejidad se realizan sesiones de ayudantía, con el fin de reforzar los contenidos vistos por el profesor en las cátedras y resolver las dudas de los estudiantes. El presente trabajo mostró una intervención con una nueva metodología para las sesiones de ayudantía, la que consistió en aplicar el aprendizaje cooperativo, es decir, los estudiantes debían trabajar juntos para construir nuevos aprendizajes. Luego de aplicar la intervención hemos podido concluir lo siguiente:

1. Al finalizar las 10 sesiones implementadas se pudo comprobar que la metodología aplicada favoreció a mejorar los aprendizajes de los estudiantes, revisando los resultados obtenidos en la asignatura, es decir, si lograron aprobar o no, ya que el 54,2% de los estudiantes aprobaron.

2. El objetivo de nuestra investigación fue diseñar y probar una intervención pedagógica dirigida a favorecer y mejorar los aprendizajes de los estudiantes que cursan la asignatura Algoritmos y programación (MTM 124), incluida en la nueva malla curricular, para lo cual se estudiaron las posibles causas de las altas tasas de reprobación que presentaban las asignaturas afines de la malla curricular antigua en los últimos años, y las estrategias que se podrían aplicar en nuestra intervención para mejorar la situación descrita. Si bien se diseñó y aplicó una intervención con la que se logró aumentar la tasa de aprobación, se hizo necesario analizar resultados en base a la asistencia a las sesiones de ayudantía, ya que el porcentaje de aprobación varió sólo en un 3,1% y un 0,6%, comparando con las asignaturas Computación (MLP 103) y Programación y programación lógica (MLP 113) respectivamente.

3. Aunque la tasa de reprobación sigue siendo alta (45,8%), se puede apreciar que los estudiantes que tenían una alta asistencia a sesiones de ayudantía, es decir, desde un 80% de asistencia, lograron aprobar la asignatura. En cambio, los estudiantes con un porcentaje de asistencia menor o igual a un 40% reprobaron la asignatura (ver tabla 20, pág. 77). Con esta información podemos concluir que quienes más asistieron aplicaron el aprendizaje cooperativo y cumplían con las tareas evaluadas dadas en sesiones de ayudantía

obtuvieron resultados favorables en la asignatura, en cambio, aquellos que no asistían probablemente estudiaban de manera individual, lo que no les permitió aprender lo necesario para rendir en las evaluaciones, además si no realizaron las tareas evaluadas fueron calificados con nota mínima, perjudicando su situación final.

4. Por lo descrito en el párrafo anterior consideramos que la intervención realizada fue exitosa, en el sentido que quienes participaron de ella sí lograron un mejor aprendizaje que quienes no participaron. Pero, lamentablemente, por la falta de compromiso de los estudiantes que faltaban a ayudantía no es posible ver reflejada esa mejora en la situación final de la asignatura, y sólo es posible notar esta diferencia al revisar la situación individual de cada estudiante según su propia asistencia a las sesiones de ayudantía.

De todas formas, se deja abierta la posibilidad para realizar cambios en la intervención propuesta con el fin que resulte más beneficiosa para el aprendizaje de los estudiantes, pero se sugiere cambiar la modalidad de asistencia libre a obligatoria, y que se exija un porcentaje mínimo de asistencia para aprobar la asignatura; que se aumenten las horas de sesiones de ayudantía y que se mantenga la metodología de trabajo cooperativo, ya que ha demostrado favorecer el aprendizaje y mejorar los resultados. Además, consideramos importante que los estudiantes conozcan la relación que existe entre programación y matemática, para aumentar su compromiso con la asignatura y esto fomente la mayor asistencia y participación de cada uno en la asignatura. También debe existir un compromiso del estudiante para trabajar en equipo, lo que es fundamental para el éxito de la intervención propuesta.

Bibliografía

Caamaño, C., Lewin, R., & Soto, J. (2012). *Informe del Comité de Pares Evaluadores, Carrera de Matemática, Universidad de Valparaíso.*

Caneo, O. (2008). El aprendizaje cooperativo aplicado en cursos de programación de computadores. *Ponencia presentada ante el 2° Congreso Nacional de Enseñanza de Ingeniería. Universidad de Antofagasta*.

Ferreiro, R. (2009). *Educación para el talento.* Recuperado el 2012, de http://educacionparaeltalento.com/files/WEBSITE_Revista_Magister_Articulo_6.pdf

Galleguillos, J., Pizarro, A., & Juyumaya, J. (2012). *Informe Proceso de Autoevaluación con fines de Acreditación Programa de Pedagogía en Matemáticas.*

Goikoetxea, E., & Gema, P. (2002). Aprendizaje cooperativo: Bases teóricas y hallazgos empíricos que explican su eficacia. *Educación XX1*, 199-226.

Jenkins, T. (2002). *University of Glagow.* Recuperado el 2012, de http://www.psy.gla.ac.uk/~steve/localed/jenkins.html

Johnson, D. W., Johnson, R., & Holubec, E. J. (1999). *El Aprendizaje Cooperativo en el Aula.* Buenos Aires: Editorial Paidós.

Johnson, D., Johnson, R. T., & Smith, K. A. (1991). Cooperative Learning: Increasing College Faculty Instructional Productivity. *ASHE-ERIC Higher Education Reports, 20*(4).

Psicocode. (s.f.). *Psicocode.* Recuperado el 2012, de http://www.psicocode.com/resumenes/6educacion.pdf

Real Academia Española. (s.f.). *Real Academia Española*. Recuperado el 2012, de http://lema.rae.es/drae/?val=cooperar

Rincón, C., Acurero, A., & Bracho, D. (2008). Aspectos de diseño de un entorno de programación colaborativo. *Enl@ce: Revista Venezolana de Información, tecnología y conocimiento 2008*(3), 11-23.

Robins, A., Rountree, J., & Rountree, N. (2003). *Georgia Institute of Technology.* Recuperado el 2012, de http://home.cc.gatech.edu/csed/uploads/2/robins03.pdf

Santiváñez, V. (2004). Recuperado el 2012, de Facultad de Ciencias de la Comunicación, Turismo y Psicología. Universidad de San Martín de Porres: http://www.fcctp.usmp.edu.pe/cultura/imagenes/pdf/18_07.pdf

Satorre, R., Llorens, F., & Puchol, J. A. (1996). *Universidad de Alicante.* Recuperado el 2012, de http://www.dccia.ua.es/~faraon/docs/programacion.pdf

Serrano, J. M., González-Herrero, M. E., & Martínez-Herrero, M. d. (1997). *Aprendizaje Cooperativo en Matemáticas: un método de aprendizaje cooperativo-individualizado para la enseñanza de las matemáticas.* Servicio de Publicaciones. Universidad de Murcia.

Slavin, R. E. (1999). *Aprendizaje Cooperativo.* AIQUE.

Universidad de Valparaíso, Carrera de Matemática. (s.f.). Programa de Asignatura MLP103.

Universidad de Valparaíso, Carrera de Matemática. (s.f.). Programa de Asignatura MLP113.

Universidad de Valparaíso, Carrera de Matemática. (s.f.). Programa de Asignatura MTM124.

Universidad de Valparaíso, Departamento de Matemática. (s.f.). Obtenido de matematica.uv.cl

Villalobos, J., Casallas, R., & Marcos, k. (2006). El reto de diseñar un primer curso de programación de computadores. Bogotá, Colombia.

Anexos

Anexo 1: Mallas Curriculares

Anexo 1.1: Malla Curricular Antigua

1° AÑO		2° AÑO	
PLAN COMÚN MATEMÁTICA PEDAGOGÍA Y LICENCIATURA			
1° SEM.	**2° SEM.**	**3° SEM.**	**4° SEM.**
MLP 100 MATEMÁTICAS GENERALES 4.5 HR.	*R 100 MLP 110 ARITMÉTICA 4.5 HR	R 110 MLP 200 ÁLGEBRA LINEAL I 4.5 HR.	R 200 MLP 210 TEORÍA DE GRUPOS 4.5 HR.
MLP 101 INTRODUCCIÓN AL CÁLCULO 4.5 HR.	R 101 MLP 111 CÁLCULO I 4.5 HR.	R 111 MLP 201 CÁLCULO II 4.5 HR.	R 201 MLP 211 CÁLCULO III 4.5 HR.
MLP 102 INGLÉS I 3.0 HR.	R 102 MLP 112 INGLÉS II 3.0 HR.	R 112 MLP 202 INGLÉS III 3.0 HR.	R 101 200 MLP 212 GEOMETRÍA 4.5 HR.
MLP 103 COMPUTACIÓN 3.0 HR	R 103 MLP 113 PROGRAMACIÓN Y PROGRAMACIÓN LÓGICA 4.5 HR.	R 103 110 111 MLP 203 INTRODUCCIÓN A LA DIDÁCTICA MATEMÁTICA 4.5 HR.	R 113 200 201 MLP 213 PROGRAMACIÓN CIENTÍFICA 4.5 HR.
		R 100 111 MLP 204 ESTADÍSTICA 3.0 HR.	R - - ASIGNATURA DE FORMACIÓN GENERAL I 1.5 HR.

3° AÑO

4° AÑO

PLAN PEDAGOGÍA

5° SEM.

R	MLP 300
1° AÑO APR	PSICOLOGÍA GENERAL
	1.5 HR.

R	MLP 301
200 201 Y 1° AÑO APR	FÍSICA
	4.5 HR.

R	MLP 302
200 Y 1° AÑO APR	GEOMETRÍA EUCLIDIANA Y NO EUCLIDIANA
	4.5 HR.

R	MLP 303
1° AÑO APR	HISTORIA DE LA EDUCACIÓN EN CHILE
	1.5 HR.

R	MLP 304
203 Y 1° AÑO APR	PAQUETES DE COMPUTACIÓN **(MD)
	4.5 HR

R	MLP 307
203 Y 1° AÑO APR	DIDÁCTICA Y COMPUTACIÓN ***(MC)
	4.5 HR

6° SEM.

R	MLP 310
300	SOCIOLOGÍA EDUCACIONAL
	3.0 HR.

R	MLP 311
300	PROCESOS COGNITIVOS
	4.5 HR

R	MLP 312
200 204 Y 1° AÑO APR	MATEMÁTICAS FINITAS
	4.5 HR.

R	MLP 313
300	FILOSOFÍA EDUCACIONAL
	3.0 HR

R	MLP 314
304	DIDÁCTICA DE LA MATEMÁTICA (MD)
	4.5 HR

R	MLP 317
200 213 Y 1° AÑO APR	ELEMENTOS DE ESTRUCTURA Y BASE DE DATOS (MC)
	4.5 HR

7° SEM.

R	MLP 400
300 311 Y 3° SEM APR	PSICOLOGÍA DEL ADOLESCENTE
	3.0 HR.

R	MLP 401
3° SEM APR	CURRÍCULUM
	3.0 HR.

R	MLP 402
302 Y 3° SEM APR	HISTORIA DE LA MATEMÁTICA
	3.0 HR.

R	MLP 403
313 Y 3° SEM APR	ÉTICA Y EDUCACIÓN
	3.0 HR

R	MLP 404
314 Y 3° SEM APR	MATEMÁTICA Y SU DIDÁCTICA I (MD)
	4.5 HR.

R	MLP 404
317 Y 3° SEM APR	SISTEMAS OPERATIVOS (MC)
	4.5 HR

8° SEM.

R	MLP 410
400	DIMENSIÓN PROFESIONAL
	3.0 HR.

R	MLP 411
2° AÑO APR	FORMULACIÓN Y EVALUACIÓN DE PROYECTOS
	3.0 HR

R	MLP 412
303 403	ADMINISTRACIÓN Y LEGISLACIÓN EDUCACIONAL
	4.5 HR.

R	MLP 413
400	PLANIFICACIÓN Y EVALUACIÓN DEL APRENDIZAJE
	3.0 HR.

R	MLP 414
404	MATEMÁTICA Y SU DIDÁCTICA II (MD)
	4.5 HR

R	MLP 417
407	REDES DE COMPUTADORES (MC)
	4.5 HR

5° AÑO

9° SEM.

R	MLP 500
8° SEM APR	PRÁCTICA PROFESIONAL
	9.0 HR.

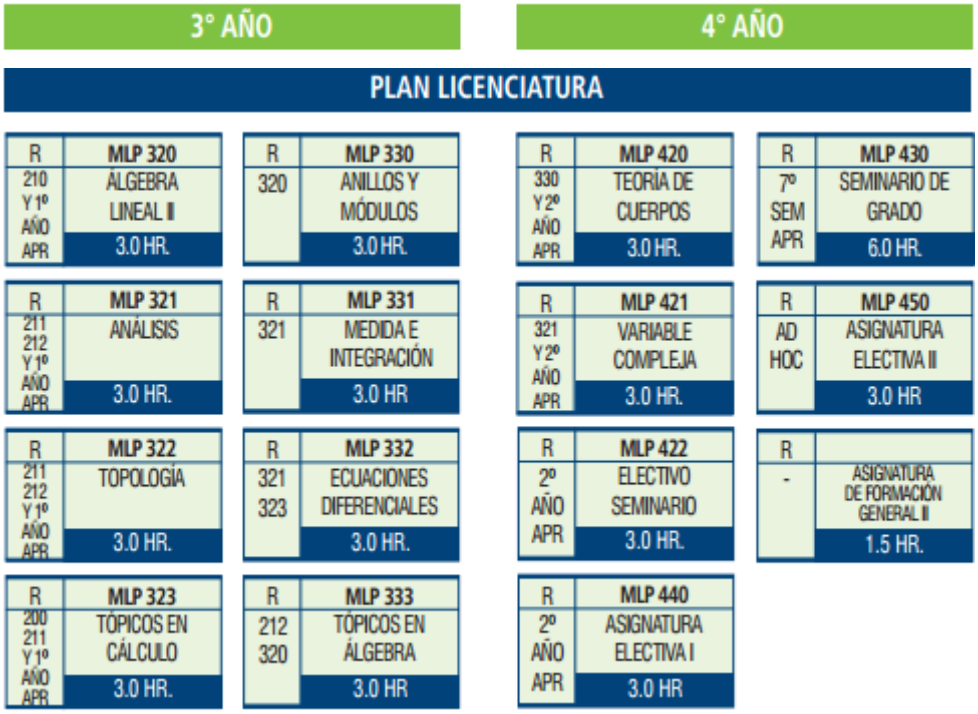
R	MLP 501
-	SEMINARIO
	3.0 HR

R	MLP 520
-	ASIGNATURA DE FORMACIÓN GENERAL II
	1.5 HR.

10° SEM.

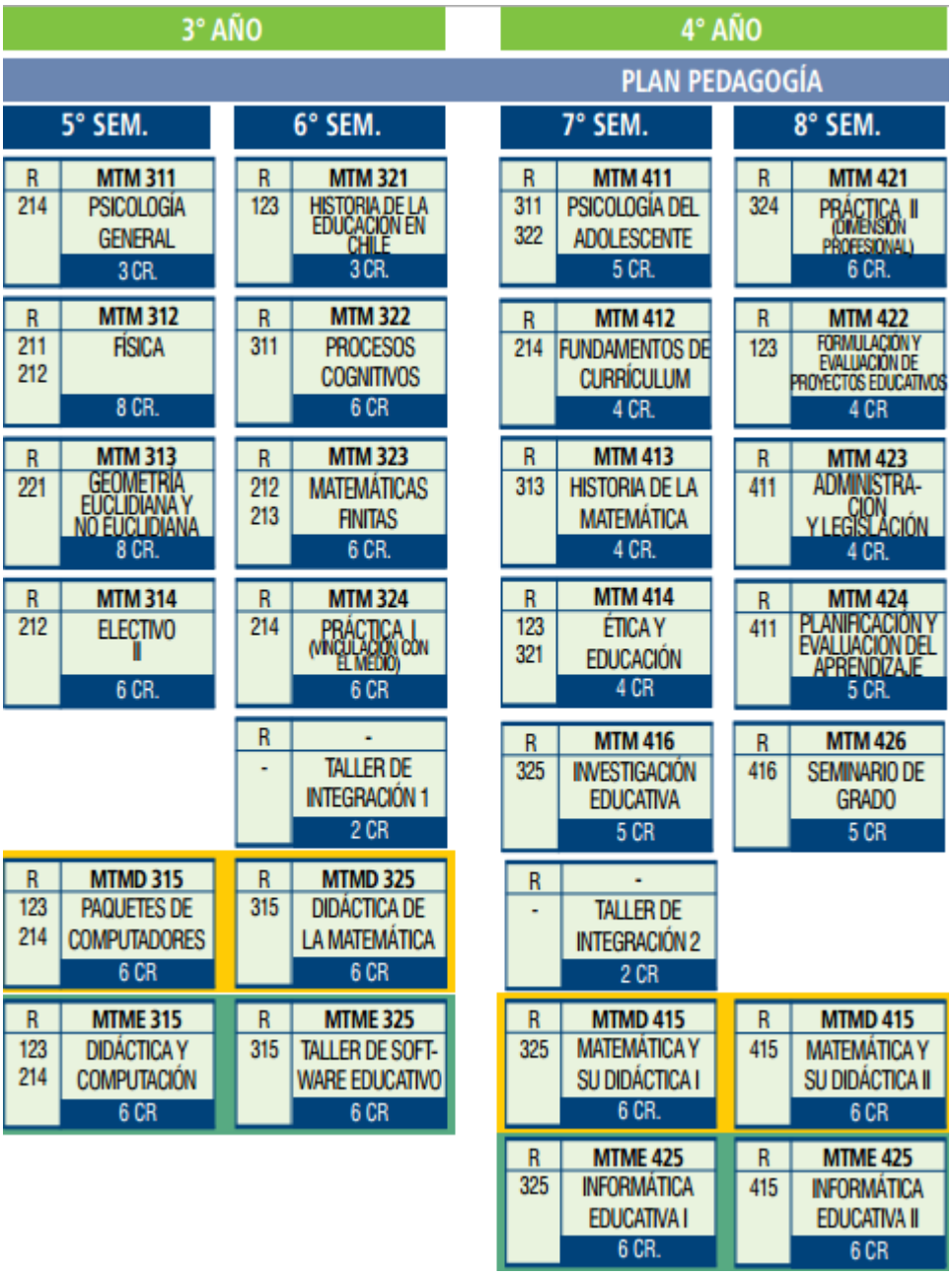
R	MLP 510
8° SEM APR	SEMINARIO DE TÍTULO
	4.5 HR.

*MD: Mención Didáctica
** MC: Mención Computación


3° AÑO
4° AÑO
PLAN LICENCIATURA
R
MLP 320
210 Y 1º AÑO APR
ÁLGEBRA LINEAL II
3.0 HR.
R
MLP 330
320
ANILLOS Y MÓDULOS
3.0 HR.
R
MLP 420
330 Y 2º AÑO APR
TEORÍA DE CUERPOS
3.0 HR.
R
MLP 430
7º SEM APR
SEMINARIO DE GRADO
6.0 HR.
R
MLP 321
211 212 Y 1º AÑO APR
ANÁLISIS
3.0 HR.
R
MLP 331
321
MEDIDA E INTEGRACIÓN
3.0 HR
R
MLP 421
321 Y 2º AÑO APR
VARIABLE COMPLEJA
3.0 HR.
R
MLP 450
AD HOC
ASIGNATURA ELECTIVA II
3.0 HR
R
MLP 322
211 212 Y 1º AÑO APR
TOPOLOGÍA
3.0 HR.
R
MLP 332
321 323
ECUACIONES DIFERENCIALES
3.0 HR.
R
MLP 422
2º AÑO APR
ELECTIVO SEMINARIO
3.0 HR.
R
-
ASIGNATURA DE FORMACIÓN GENERAL II
1.5 HR.
R
MLP 323
200 211 Y 1º AÑO APR
TÓPICOS EN CÁLCULO
3.0 HR.
R
MLP 333
212 320
TÓPICOS EN ÁLGEBRA
3.0 HR
R
MLP 440
2º AÑO APR
ASIGNATURA ELECTIVA I
3.0 HR

Anexo 1.2: Malla curricular 2012

1° AÑO				2° AÑO			
PLAN COMÚN MATEMÁTICA PEDAGOGÍA Y LICENCIATURA							
1° SEM.		2° SEM.		3° SEM.		4° SEM.	
MTM 111	MATEMÁTICAS GENERALES	**R 111	MTM 121 ARITMÉTICA	R 121	MTM 211 ÁLGEBRA LINEAL I	R 211	MTM 221 TEORÍA DE GRUPOS
* 9 CR.		8 CR		9 CR.		9 CR.	
MTM 112	INTRODUCCIÓN AL CÁLCULO	R 112	MTM 122 CÁLCULO I	R 122	MTM 212 CÁLCULO II	R 212	MTM 222 CÁLCULO III
9 CR.		9 CR.		9 CR.		8 CR.	
MTM 113	TALLER DE LENGUAJE	R -	MTM 123 FUNDAMENTOS FILOSÓFICOS Y SOCIOLÓGICOS DE LA EDUCACIÓN	R 111 122	MTM 213 ESTADÍSTICA	R 121 122	MTM 223 GEOMETRÍA
3 CR.		4 CR.		4 CR.		7 CR.	
MTM 114	COMPUTACIÓN	R 114	MTM 124 ALGORITMOS Y PROGRAMACIÓN	R 114 121 122	MTM 214 INTRODUCCIÓN A LA DIDÁCTICA DE LA MATEMÁTICA	R 124	MTM 224 PROGRAMACIÓN CIENTÍFICA
6 CR		6 CR.		4 CR.		6 CR.	
MTM 115	ELECTIVO I	R -	MTM 125 INGLÉS I	R 125	MTM 215 INGLÉS II		
2 CR		4 CR.		4 CR.			


3° AÑO
4° AÑO
PLAN PEDAGOGÍA
5° SEM.
6° SEM.
7° SEM.
8° SEM.
R 214 MTM 311 PSICOLOGÍA GENERAL 3 CR.
R 123 MTM 321 HISTORIA DE LA EDUCACIÓN EN CHILE 3 CR.
R 311 322 MTM 411 PSICOLOGÍA DEL ADOLESCENTE 5 CR.
R 324 MTM 421 PRÁCTICA II (DIMENSIÓN PROFESIONAL) 6 CR.
R 211 212 MTM 312 FÍSICA 8 CR.
R 311 MTM 322 PROCESOS COGNITIVOS 6 CR
R 214 MTM 412 FUNDAMENTOS DE CURRÍCULUM 4 CR.
R 123 MTM 422 FORMULACIÓN Y EVALUACIÓN DE PROYECTOS EDUCATIVOS 4 CR
R 221 MTM 313 GEOMETRÍA EUCLIDIANA Y NO EUCLIDIANA 8 CR.
R 212 213 MTM 323 MATEMÁTICAS FINITAS 6 CR.
R 313 MTM 413 HISTORIA DE LA MATEMÁTICA 4 CR.
R 411 MTM 423 ADMINISTRACIÓN Y LEGISLACIÓN 4 CR.
R 212 MTM 314 ELECTIVO II 6 CR.
R 214 MTM 324 PRÁCTICA I (VINCULACIÓN CON EL MEDIO) 6 CR
R 123 321 MTM 414 ÉTICA Y EDUCACIÓN 4 CR
R 411 MTM 424 PLANIFICACIÓN Y EVALUACIÓN DEL APRENDIZAJE 5 CR.
R - - TALLER DE INTEGRACIÓN 1 2 CR
R 325 MTM 416 INVESTIGACIÓN EDUCATIVA 5 CR
R 416 MTM 426 SEMINARIO DE GRADO 5 CR
R 123 214 MTMD 315 PAQUETES DE COMPUTADORES 6 CR
R 315 MTMD 325 DIDÁCTICA DE LA MATEMÁTICA 6 CR
R - - TALLER DE INTEGRACIÓN 2 2 CR
R 123 214 MTME 315 DIDÁCTICA Y COMPUTACIÓN 6 CR
R 315 MTME 325 TALLER DE SOFTWARE EDUCATIVO 6 CR
R 325 MTMD 415 MATEMÁTICA Y SU DIDÁCTICA I 6 CR.
R 415 MTMD 415 MATEMÁTICA Y SU DIDÁCTICA II 6 CR
R 325 MTME 425 INFORMÁTICA EDUCATIVA I 6 CR.
R 415 MTME 425 INFORMÁTICA EDUCATIVA II 6 CR

5° AÑO

9° SEM.		10° SEM.	
R	MTM 511	R	MTM 510
421	PRÁCTICA PROFESIONAL	512	TRABAJO DE TÍTULO II
	22 CR.		28 CR.
R	MTM 501	R	-
426	TRABAJO DE TÍTULO I	-	TALLER DE INTEGRACIÓN 3
	8 CR		2 CR.

MENCIÓN DIDÁCTICA DE LA MATEMÁTICA

MENCIÓN INFORMÁTICA EDUCATIVA

3° AÑO

4° AÑO

PLAN LICENCIATURA EN MATEMÁTICAS

R	MTML 311	R	MTML 321	R	MTML 411	R	MTML 421
221	ÁLGEBRA LINEAL II	311	ANILLOS Y MÓDULOS	321	TEORÍA DE CUERPOS	414	SEMINARIO DE GRADO
	8 CR.		7 CR.		7 CR.		21 CR.
R	MTML 312	R	MTML 322	R	MTML 412	R	MTML 422
222	ANÁLISIS	312	MEDIDA E INTEGRACIÓN	322	VARIABLE COMPLEJA	221 222 223	ASIGNATURA ELECTIVA II
	8 CR.		7 CR		7 CR.		7 CR
R	MTML 313	R	MTML 323	R	MTML 413	R	
222 223	TOPOLOGÍA	312 314	ECUACIONES DIFERENCIALES	221 222 223	ELECTIVO SEMINARIO	-	TALLER DE INTEGRACIÓN 3
	8 CR.		7 CR.		7 CR.		2 CR.
R	MTML 314	R	MTML 324	R	MTML 414		
222	TÓPICOS EN CÁLCULO	311 223	TÓPICOS EN ÁLGEBRA	221 222 223	ASIGNATURA ELECTIVA I		
	6 CR.		7 CR.		7 CR.		
		R		R			
		-	TALLER DE INTEGRACIÓN 1	-	TALLER DE INTEGRACIÓN 2		
			2 CR.		2 CR.		

Anexo 2: Prueba de diagnóstico

PRUEBA DE DIAGNÓSTICO

En la siguiente prueba de diagnóstico encontrará una serie de preguntas que tienen que ver con los conocimientos básicos que Ud. debería disponer con el objeto de asegurar su éxito en la asignatura.

Como comprobará las preguntas no son difíciles pero requieren de su esfuerzo para comprender exactamente lo que se le está preguntando, esto con la finalidad de que su respuesta sea lo más precisa posible.

Por esta razón es importante que lea atentamente cada pregunta y reflexione sobre la misma antes de construir una respuesta. Verifique siempre que no se ha equivocado en la respuesta que quería construir.

1. Cálculo

1.1 Sabemos que 33 es el resultado de haber aplicado un porcentaje a la cantidad 213. Se requiere determinar qué porcentaje es 33 con relación a 213

Resultado	**33 es de 213 el %:**

1.2 Hemos realizado la compra de un artículo cuyo precio de venta al público es de $1.240. Necesitamos saber cuánto se deberá pagar por el impuesto de IVA, sabiendo que éste es del 18%.

Resultado	**El I V A de $ 1.240 es:**

2. Expresiones algebraicas:

2.1 A continuación se le presentan varias expresiones algebraicas que es necesario reducir a la forma más sencilla posible. Se le solicita que reduzca las siguientes expresiones:

2.1.1 $2x + 5 - 3xy - 4 + 3x - xy$

Expresión reducida =	

2.1.2 $-3x^2+5xy^3+8x-x^2+6x^3y+10+x^3-5y^3x$

Expresión reducida =	

2.2 Se tienen varias expresiones que representan una igualdad y en las que existe una incógnita. Le pedimos que determine el valor de la incógnita para que se verifique la igualdad propuesta en cada una de las siguientes expresiones:

2.2.1 $9y-11=-10+12y$

Valor de la incógnita es	y =

2.2.2 $16+7t-5+t=11t-3-t$

Valor de la incógnita es	t =

2.3 Disponemos de tres canastos que contienen en total 575 frutas. El primer canasto tiene 10 frutas más que el segundo y 15 más que el tercero. Encuentre la cantidad de fruta en cada uno de los canastos.

Nº de frutas en cada canasto	1º canasto =	
	2º canasto =	
	3º canasto =	

2.4 Se nos han dado tres sistemas en los que en cada uno de ellos hay dos valores desconocidos. Se le solicita que determine el valor de cada uno de esos valores en cada uno de los sistema para que se cumpla las igualdades en cada caso:

2.4.1 Calcule los valores x e y en el siguiente sistema:

$5x+6y=20$

$8x-6y=-46$

Resultados	**x =**	
	y =	

2.4.2 Determine el valor de m y n que cumplan:

$\frac{3m}{2}+n=11$

$m+\frac{n}{2}=7$

Resultados	**m =**	
	n =	

2.4.3 Calcule el valor de r y s que cumplen con:

$(r-s)-(6r+8s)=-(10r+5s+3)$

$(r+s)-(9s-11r)=2s-2r$

Resultados	**r =**	
	s =	

2.5 Un sistema de ecuaciones de primer grado como el del ejercicio anterior puede ser resuelto mediante métodos conocidos como *Eliminación por igualación, Eliminación por sustitución* o mediante *Reducción*. Tome como referencia uno de estos métodos de solución (a su elección) y describa con el mayor detalle posible la secuencia de pasos y las operaciones necesarias que se deberán ejecutar para resolver un sistema cualquiera de ecuaciones de primer grado.

2.6 Nos han sido dados los siguientes valores para $a=2$; $b=\frac{1}{3}$; $x=-2$; $y=-1$; $m=3$; y $n=\frac{1}{2}$, con lo cual deberá calcular ahora el valor de cada una de las siguientes expresiones:

2.6.1 $\frac{x^4}{8}-\frac{x^2y}{2}+\frac{3xy^2}{2}-y^2$

Resultado	

2.6.2 $(a-x)^2+(x-y)^2+(x^2-y^2)(m+x-n)$

Resultado	

2.7 Más abajo encontrará unas expresiones algebraicas que corresponden a ecuaciones de 2° grado, para las que es necesario determinar los valores de *x*, que cumplen con la igualdad especificada en cada caso:

2.7.1 $3(3x-2)=(x+4)(4-x)$

Resultado de x_1	
Resultado de x_2	

2.7.2 $(x+2)^3-(x-1)^3=x(3x+4)+8$

Resultado de x_1	
Resultado de x_2	

2.7.3 $\frac{4x^2}{x-1} - \frac{1-3x}{4} = \frac{20x}{3}$

Resultado de x_1	
Resultado de x_2	

2.8 El problema que tiene que resolver ahora consiste en lo siguiente: Considerando como forma general de una ecuación de segundo grado la expresión $Ax^2 + Bx + C = 0$, describa la secuencia de acciones y operaciones que se deben realizar para encontrar los valores de x que satisfacen la forma general.

3 Álgebra de Boole

3.1. Las tareas que debe resolver ahora tiene que ver con estructuras lógicas de acuerdo con los principios que rigen el álgebra de Boole. Así pues, considerando $p = Verdadero$; $q = Falso$ y $r = Verdadero$ determine el resultado de las siguientes expresiones lógicas. Considere que $\cap$ simboliza el conector lógico y o *and*, en tanto $\cup$ simboliza el conector lógico *o* u *or*

3.1.1. $p \cup q$

Resultado	

3.1.2. $r \cap p$

Resultado	

3.1.3. $(q \cup p) \cap (r \cap p)$

Resultado	

3.1.4. $p \cap (p \cup r) \cup (q \cup r)$

Resultado	

4 Resolución de problemas

En este apartado se le van a proponer algunos problemas sobre los que tendrá que analizar y describir los procedimientos lógicos que deben seguirse al objeto de hallar la solución que se le solicita a partir de los datos y elementos que se le proponen. Le invitamos a que lea con atención la formulación de cada problema antes de que trate de ofrecer la solución.

4.1. Describa la secuencia de acciones y operaciones para encontrar el área de un rectángulo sabiendo que ésta es equivalente a la mitad del cuadrado de su diagonal.

4.2. Describa la secuencia de acciones y operaciones que deberían realizarse para determinar cuando ganó cada día una persona, sabiendo que en tres días ganó $G, y que cada día ganó la mitad de lo que ganó el día anterior.

4.3. Sabiendo que la edad de A es el triple de la edad de B, y la de B es 5 veces la de C. Además se sabe que B tiene 12 años más que C. Describa la secuencia de acciones y operaciones que debieran efectuarse para determinar la edad de cada uno.

4.4. Suponga que se tiene una ecuación de primer grado de la forma $ax + b = c$ Describa la secuencia de acciones y operaciones que deberían efectuarse para encontrar el valor de la incógnita x.

4.5. Supongamos que vamos a una tienda que se encuentra en liquidación. Supongamos también que los precios marcados en cada artículo, no se encuentran con el descuento aplicado, considerando además que los artículos pueden tener distintos porcentajes de descuentos. Describa la secuencia de acciones y operaciones que se debieran realizar para determinar el valor final a apagar por un artículo cualquiera de la tienda.

4.6. Cuando se emite la factura por la compra de un artículo en ésta se debe indicar el valor del mismo, sin el impuesto IVA, lo que constituye su valor neto. Después deberá expresarse el monto del impuesto IVA del artículo, y finalmente se debe expresar el valor total a pagar por el artículo, una vez incluido el impuesto IVA.

Describa la secuencia de acciones y operaciones que son necesarias ejecutar para determinar el valor neto y el impuesto IVA de un artículo cualquiera a partir de su valor final a pagar.

4.7. Las compañías telefónicas, al igual que otras suministradoras de servicio, suelen establecer valores mínimos por el uso del servicio durante un tiempo establecido, incrementándose después en función del tiempo de uso que sobrepase lo fijado.

Así, una cierta compañía telefónica en su servicio de llamadas locales cobra $85 por los primeros 3 minutos de llamada, incluso aunque estos no se completen. Pero si la llamada excede de este tiempo, se cobran $30 por cada minuto adicional.

Teniendo en cuenta esto describa la secuencia de acciones y operaciones que es necesario realizar para calcular el valor total de una llamada cualquiera, de acuerdo a su duración real.

Anexo 3: Desarrollo de tareas evaluadas

Anexo 3.1: Desarrollo tarea evaluada 1

I. Ingrese valor a la variable x e y, luego encuentre el valor de las variables a, b, c, d, e.

1. $b = x + y$
2. $c = x \cdot y$
3. $d = x/y$
4. $e = b^2 + 2(c + d)^3$
5. $a = \frac{x^3(2x^2+3{,}56x-5{,}21)}{\sqrt{3x+2{,}3}}$

```
En Scilab
-->x=7
 x  =
     7.
 -->y=5
 y  =
     5.
 -->b=x+y
 b  =
     12.
 -->c=x*y
 c  =
     35.
 -->d=x/y
 d  =
     1.4
 -->e=b^2+2*(c+d)^3
 e  =
     96601.088
 --
>a=x^3*(2*x^2+3.56*x-
5.21)/sqrt(3*x+2.3)
 a  =
    8364.2984
```

II. Encontrar el área y perímetro de las siguientes circunferencias

1. Circunferencia de radio 5,2
2. Circunferencia de diámetro 18

```
En Scilab
-->r1=5.2
```

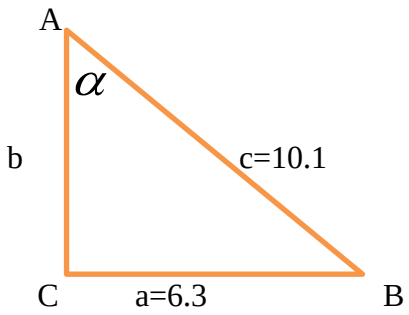
```
 r1  =
     5.2
 -->a1=%pi*r1^2
 a1  =
     84.948665
 -->p1=2*%pi*r1
 p1  =
     32.672564
 -->r2=9
 r2  =
     9.
 -->a2=%pi*r2^2
 a2  =
     254.469
 -->p2=2*%pi*r2
 p2  =
     56.548668
```

Cabe mencionar que r1, a1 y p1 corresponden al radio, área y perímetro de la circunferencia 1, y que r2, a2 y p2 corresponden al radio, área y perímetro de la circunferencia 2.

Anexo 3.2: Desarrollo tarea evaluada 2

Ejercicio 1.

Para el triángulo rectángulo encuentre el valor de "b" y "α"



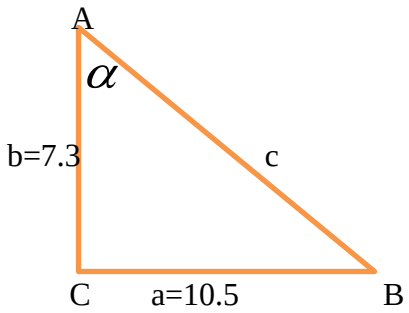
En este ejercicios los alumnos deben reconocer la formula a usar para encontrar los valores pedidos y luego saber cómo se ingresa las funciones a utilizar en Scilab.

La fórmula a usar es ""b" al cuadrado es igual a la raíz cuadrada de "c" al cuadrado menos "a" al cuadrado"

```
En Scilab
-->a=6.3
 a  =
    6.3
-->c=10.1
c  =
    10.1
-->b=sqrt(c^2-a^2)
b  =
      7.8943017
-->alfa=asin(a/c)
 alfa  =
   0.6735471
-->alfagrados=(alfa*180)/3.14
alfagrados  =
    38.610981
```

Ejercicio 2.

Para el triángulo rectángulo encuentre el valor de c y α




```
-->a=10.5
 a  =
  10.5
-->b=7.3
 b  =
    7.3
-->c=sqrt(b^2+a^2)
 c  =
 12.788276
 -->alfa=acos(b/c)
 alfa  =
    0.9632734
 -->alfagrados=(alfa*180)/3.14
 alfagrados  =
    55.219496
```

Ejercicio 3.

Crear una Función que contenga 4 funciones predefinidas. Evaluar la función para un x cualquiera. Encuentra la parte entera superior e inferior y redondeo.

En este ejercicio el alumno debe hacer ingreso de funciones predefinidas de Scilab en una sola función, y luego ser evaluada para cualquier valor de x.

Ejercicio 4.

Sea la función $y = \left|\sqrt{x^2 - |x+2|}\right|$

Evaluar para los valores x=1, x=2, x=0, x= -1, x= -2.

```
En Scilab
-->x=1
 x  =
     1.
 -->y=abs(sqrt(x^2-abs(x+2)))
y  =
  1.4142136
 -->x=2
 x  =
    2.
 -->y=abs(sqrt(x^2-abs(x+2)))
 y  =
    0.
 -->x=0
 x  =
    0.
 -->y=abs(sqrt(x^2-abs(x+2)))
 y  =
    1.4142136
 -->x=-1
 x  =
  - 1.
 -->y=abs(sqrt(x^2-abs(x+2)))
 y  =
  0.
 -->x=-2
 x  =
  - 2.
 -->y=abs(sqrt(x^2-abs(x+2)))
 y  =
    2.
```

Ejercicio 5

Encontrar las funciones trigonométricas para $\beta = 30°$.

En el siguiente ejercicio los alumnos deben primero ingresar el valor del ángulo en grados y luego transformarlo a radianes para luego encontrar las funciones trigonométricas predefinidas en Scilab.

```
-->a=30
 a  =
    30.
-->ar=(30*3.14)/180
 ar  =
    0.5233333
-->sin(ar)
ans  =
    0.4997701
```

```
-->cos(ar)
 ans  =
    0.8661581
 -->acos(ar)
 ans  =
    1.0200383
 -->asin(ar)
 ans  =
    0.5507580
 -->tan(ar)
 ans  =
    0.5769964
 -->atan(ar)
 ans  =
    0.4821396
-->cotg(ar)
 ans  =
    1.7331131
```

Anexo 3.3: Desarrollo tarea evaluada 3

Ejercicio 1. Ingresar números complejos, encontrar su parte imaginaria, parte real, conjugado y argumento.

```
En Scilab
-->z1=5.3+10*%i
 z1  =
  5.3 + 10.i
 -->imag1=imag(z1)
 imag1  =
    10.
 -->real1=real(z1)
  real1  =
      5.3
 -->conjugado=conj(z1)
 conjugado  =
   5.3 - 10.i
 -->magnitud=abs(z1)
 magnitud  =
 11.317685
-->argumento=atan(imag1/real1)
 argumento  =
 1.0834377
```

Ejercicio 2. Ingresar números complejos, encontrar su parte imaginaria, parte real, conjugado y argumento.

```
 En Scilab
-->z2=16+2*%i
 z2  =
    16. + 2.i
-->imag2=imag(z2)
 imag2  =
     2.
 -->real2=real(z2)
 real2  =
    16.
 -->conjugado2=conj(z2)
 conjugado2  =
    16. - 2.i
 -->magnitud2=abs(z2)
 magnitud2  =
     16.124515
 >argumento2=atan(imag2/real2)
 argumento2  =
    0.1243550
```

Ejercicio 3. Realizar las operaciones suma, resta, multiplicación y división de los números complejos antes ingresados.

```
-->z1+z2
 ans  =
     21.3 + 12.i
 -->z1-z2
 ans  =
  - 10.7 + 8.i
 -->z2-z1
 ans  =
    10.7 - 8.i
 -->z1*z2
 ans  =
     64.8 + 170.6i
 -->z1/z2
 ans  =
    0.4030769 + 0.5746154i
 -->z2/z1
 ans  =
    0.8181747 - 1.1663674i
 -->(((z1*z2)-z1)/z2)+(2.2+8*%i)
 ans  =
     7.0969231 + 17.425385i
```

Ejercicio 4. Ingresar polinomios especificando sus coeficientes y obtener sus raíces

```
-->p1=poly([1,2,3],'x','c')
 p1  =
                        2
    1 + 2x + 3x
 -->roots(p1)
 ans  =
  - 0.3333333 + 0.4714045i
  - 0.3333333 - 0.4714045i
 -->q1=poly([4,5,6],'x','c')
 q1 =
                       2
    4 + 5x + 6x
 -->roots(q1)
 ans  =
  - 0.4166667 + 0.7021791i
  - 0.4166667 - 0.7021791i
```

Ejercicio 5. Ingresar polinomios especificando sus raíces y obtener sus raíces con las funciones antes enseñadas.

```
-->p2=poly([0,9],'x')
 p2  =
          2
  - 9x + x
 -->roots(p2)
 ans  =
    9.
    0
 -->q2=poly([10,11],'x')
 q2  =
                           2
    110 - 21x + x
 -->roots(q2)
 ans  =
    11.
   10.
```

Ejercicio 6. Realizar operaciones con los polinomios antes ingresados.

```
-->p1+q2
 ans  =
                  2
    111 - 19x + 4x
-->p1-p2
 ans  =
                2
    1 + 11x + 2x
-->q1-p1
 ans  =
               2
    3 + 3x + 3x
-->q1*p2
 ans  =
       2       3      4
  - 36x - 41x - 49x + 6x
-->p1/p2
 ans  =
               2
    1 + 2x + 3x
    -----------
                2
    - 9x + x
-->q2/p1
 ans  =
                 2
    110 - 21x + x
    -------------
                 2
```

```
     1 + 2x + 3x
-->p1+p2+q1+q2
 ans  =
                    2
    115 - 23x + 11x
```

Anexo 3.4: Desarrollo tarea evaluada 4

1. Escriba un programa que permita ingresar la edad de una cierta persona, y muestre el mensaje "NIÑO" si la edad es hasta 10; el mensaje "PUBER" si la edad es mayor que 10 y menor o igual a 14; el mensaje "ADOLESCENTE" si la edad es mayor o igual que 15 y menor o igual a 18; el mensaje "JOVEN" si la edad es mayor o igual que 19 y menor o igual a 25; el mensaje "ADULTO" si la edad es mayor o igual que 26 y menor o igual a 65; y el mensaje "ANCIANO" si la edad es mayor que 65.

```
En Scilab

      e=input('ingrese edad');
          if (e<=10) then
              printf('niño');
          else
              if (e<=14) then
                  printf('puber');
              else
                  if (e<=18) then
                      printf('adolecente');
                  else
                      if (e<=25) then
                          printf('joven');
                      else
                          printf('anciano');
                      end
                  end
              end
          end
```

2. Escriba un programa que determine si tres enteros ingresados a, b y c son o no una terna pitagórica, es decir, si es posible que $a^2 + b^2 = c^2$ ó $b^2 + c^2 = a^2$ ó $a^2 + c^2 = b^2$

```
En Scilab

      a=input('ingrese medida lado a');
      b=input('ingrese medida lado b');
      c=input('ingrese medida lado c');
      if (a^2+b^2==c^2) | (a^2+c^2==b^2) | (b^2+c^2==a^2) then
          printf('forman terna pitagórica');
      else
          printf('no forman terna pitagórica');
      end
```

3. En una cierta empresa comercial, el sueldo mensual de los vendedores corresponde a la suma del sueldo base más una comisión que corresponde a un porcentaje de las ventas que el vendedor haya efectuado en el mes. El cálculo del porcentaje de las ventas se realiza según la siguiente tabla:
   - Si las ventas fueron inferiores a $4.000.000 el vendedor no recibe comisión
   - Si las ventas fueron iguales a $4.000.000 y menores de $10.000.000 la comisión es de un 3% de las ventas
   - Si las ventas fueron iguales a o superiores a $10.000.000 la comisión es de un 7% de las ventas

   Escriba un programa tal que ingresado el sueldo base de un trabajador y el monto de las ventas efectuadas en el mes, calcule y muestre el sueldo mensual del vendedor.

```
En Scilab
sb=input('ingrese sueldo base');
v=input('ingrese ventas');
if sb>0 & (v>=0) then
    if v<4000000 then
        s=sb;
    else
        if v<10000000 then
            s=sb+v*0.03;
        else
            s=sb+v*0.07;
        end
    end
    printf('\n el sueldo mensual es %f',s);
else
    printf('error');
    end
```

Anexo 3.5: Desarrollo tarea evaluada 5

1) Haga un programa que permita ingresar las notas (5 notas) de un alumno en la asignatura de matemática y luego muestre el promedio del alumno, la nota más alta y la más baja que obtuvo.

```
En Scilab

      n=input('La calificación está en el  rango desde 1.0 hasta 7.0,ingresar
      primera calificación de alumno:');
      al=n;
      ba=n;
      s=n;
      c=2;
      while (c<=5)
          a=input('La  calificación  está  en  el    rango  desde  1.0  hasta
      7.0,ingresar siguiente calificación de alumno:');
          s=s+a;
          if a>al then
              al=a;
          end
          if a<ba then
              ba=a;
          end
          c=c+1;
      end
      printf('\n El promedio del alumno es:%f \n La nota más alta del alumno
      es:%f \n La nota más baja del alumno es:%f',s/5,al,ba)
```

2) Realice un programa que permita dar como salida la población de dos países (a y b), teniendo en cuenta para tal propósito lo siguiente:

   En el Primer Año el País “a” tiene menos población que el país “b”

   Las Tazas de crecimiento de los países “a” y “b” son de 6% y 3% anuales respectivamente.

   Se debe dar como salidas las poblaciones desde el segundo año hasta que la población de “a” exceda a la población de “b”, además la cantidad de años que transcurrieron para que esto sucediera

```
En Scilab

      a=input('ingresar poblacion mayor de los paises (A):');
      b=input('ingresar poblacion menor de los paises (B):');
      if a>b & b>0 then
          c=0;
          while int(a)>=int(b)
              a=(a*1.03);
              b=(b*1.06);
              c=c+1;
              printf('\nEl año %d la poblacion del pais A es %d y la poblacion
```

```
del pais B es %d',c+1,a,b);
    end
    printf('\n Los años que tarda la poblacion de B en superar a la
poblacion de A es:%d',c);
else
    printf('\n error');
end
```

3) Haga un programa que permita ingresar el promedio de notas de 10 alumnos y muestre la cantidad de alumnos en las siguientes categorías

- con promedio entre 60 y 70 “destacado”
- con promedio entre 50 y 59 “bueno”
- con promedio entre 40 y 49 “suficiente”
- con promedio bajo 40 “insuficiente”

```
En Scilab

c=1
while (c<=10)
    a=input('En escala desde 10 hasta el 70,ingresar promedio del
alumno:');
    if a<40 then
        printf(' El alumno corresponde a la categoria de INSUFICIENTE');
    else
        if (a<50) then
            printf('  El  alumno  correspoonde  a  la  categoria  de
SUFICIENTE');
        else
             if (a<60) then
                printf(' El alumno corresponde a la categoria de BUENO');
            else
                printf('El  alumno  corresponde  a  la  categoria  de
DESTACADO');
            end
        end
    end
    c=c+1;
end
```

4) Haga un programa que multiplique los números del 1 al 10

```
En Scilab

c=1;
p=1;
while (c<=10)
    p=p*c;
```

```
        c=c+1;
    end
    printf('El producto de los números enteros desde el 1 hasta el 10
    es:%d',p)
```

5) Haga un programa que permita ingresar la base y el exponente, y luego calcule la potencia.

```
En Scilab

    b=input('ingresar base:');
    e=input('ingresar exponente natural:');
    c=1;
    p=1;
    while (c<=e)
        p=p*b;
        c=c+1;
    end
    printf('el valor de la potencia de base %d y de exponente %d es:
    %d',b,e,p)
```

6) Haga un programa que permita calcular el factorial de un número

```
En Scilab

    n=input('ingrese un entero mayor o igual a 0:');
    if n==0 then
        printf('el valor de %d! es: 1',n);
    else
        c=1;
        p=1;
        while (c<=n)
            p=p*c;
            c=c+1;
        end
        printf('el valor de %d! es: %d',n,p);
    end
```

7) Haga un programa que determine el valor de la serie

$$s = 1 + (1 + x) + \frac{(2 + x)^2}{2!} + \frac{(3 + x)^3}{3!} + \frac{(4 + x)^4}{4!} + \cdots + \frac{(n + x)^n}{n!}$$

para un cierto *x* y *n* dados.

```
En Scilab

    n=input('ingresar un numero entero (n) mayor o igual a 0:');
    x=input('ingresar un numero real (x):');
```

```
if n==0 then
    printf('el valor de la serie es 1');
else
    c=1;
    f=1;
    s=1;
    while (c<=n)
        f=f*c;
        b=1;
        d=1;
        while (b<=c)
            d=d*(c+x);
            b=b+1;
        end
        s=(d/f)+s;
        c=c+1;
    end
    printf('el valor de la serie es:%f',s);
end
```

Anexo 3.6: Desarrollo tarea evaluada 6

Deben crear un programa que incluye el uso de menú y funciones.

En el programa principal se debe crear y llenar 3 vectores, además crear un menú que incluya todas las posibles llamadas a las demás funciones.

Las funciones que deben crear son:

1. Función que llene un vector de tamaño 5.
2. Función que muestre el vector de tamaño 5.
3. Función que cuente la cantidad de números pares almacenados en el vector.
4. Función que calcule el promedio de los valores almacenados en el vector.
5. Función que sume dos vectores.
6. Función que busque un número en el vector.
7. Función que cuente los números divisibles por 3 almacenados en el vector.
8. Función que cuente los números negativo de los tres vectores.
9. Función que muestre los divisores de los elementos del vector.
10. Función que cuente los números primos presentes en el vector.

```
function v=llena_vector(v)
   for i=1:5
   v(i)=input('Ingrese un entero:')
   end
   printf('\n')
endfunction

function muestra_vector(v)
    for i=1:5
         printf('El valor en %i es %i \n',i, v(i))
   end
   printf('\n')
endfunction

function cp=pares(v)
   cp=0
   for i=1:5
     if modulo(v(i),2)==0 then
   cp=cp+1
   end
   end
endfunction
```

```
function p=promedio(v)
   s=0
   for i=1:5
   s=s+v(i)
   end
   p=s/5
endfunction

function suma=suma_vectores(v1, v2)
   for i=1:5
   suma(i)=v1(i)+v2(i)
   end
endfunction

function f=busqueda(v, bus)
   f=0
   for i=1:5
   if bus==v(i) then
   f=f+1
   end
   end
endfunction

function c=divisores(v)
   c=0
   for i=1:5
   if modulo(v(i),3)==0 then
   c=c+1
   end
   end
endfunction

function c=negativo(v1, v2, v3)
   c=0
   for i=1:5
       if v1(i)<0 then
           c=c+1
           if v2(i)<0 then
               c=c+1
               if v3(i)<0 then
                   c=c+1
               end
           end
       end
       end
 endfunction

function div_elementos(v)
   for i=1:5
       for j=1:v(i)
           if modulo (v(i),j)==0 then
               printf('El %i es divisible por %i\n',v(i),j)
           end
       end
```

```
   end
   printf('\n')
   endfunction

function cp=can_pri(v)
   cp=0
   for i=1:5
       cd=0
       for j=1:5
           if modulo(v(i),j)==0 then
               cd=cd+1
               end
           end
           if cd==2 then
               cp=cp+1
           end
       end
endfunction

a=[ ]
b=[ ]
c=[ ]
a=llena_vector(a)
b=llena_vector(b)
c=llena_vector(c)

op=1
while op~=0
   printf('1-Mostrar el primer vector \n')
   printf('2-Mostrar el segundo vector\n')
   printf('3-Mostrar el tercer vector \n')
   printf('4-Contar la cantidad de elementos pares en el primer vector\n')
   printf('5-Contar la cantidad de elementos pares en el segundo vector\n')
   printf('6-Contar la cantidad de elementos pares en el tercer vector\n')
   printf('7-Calcular y mostrar el promedio de los elementos del primer
vector\n')
   printf('8-Calcular y mostrar el promedio de los elementos del segundo
vector\n')
   printf('9-Calcular y mostrar el promedio de los elementos del tercer
vector\n')
   printf('10-Sumar vectores: primer vector y segundo vector\n')
   printf('11-Sumar vectores: primer vector y tercer vector\n')
   printf('12-Sumar vectores: segundo vector y tercer vector\n')
   printf('13-Buscar un número determinado e indicar la cantidad de veces que
se encuentra en el primer vector\n')
   printf('14-Buscar un número determinado e indicar la cantidad de veces que
se encuentra en el segundo vector\n')
   printf('15-Buscar un número determinado e indicar la cantidad de veces que
se encuentra en el tercer vector\n')
   printf('16-Mostrar la cantidad de valores que son divisibles por 3 en el
primer vector \n')
   printf('17-Mostrar la cantidad de valores que son divisibles por 3 en el
segundo vector \n')
   printf('18-Mostrar la cantidad de valores que son divisibles por 3 en el
```

```
tercer vector \n')
   printf('19-Mostrar el total de valores negativos que existen entre los tres
vectores \n')
   printf('20-Buscar y mostrar los divisores de cada elemento del primer
vector\n')
   printf('21-Buscar y mostrar los divisores de cada elemento del segundo
vector\n')
   printf('22-Buscar y mostrar los divisores de cada elemento del tercer
vector\n')
   printf('23-Mostrar la cantidad de números primos en el primer vector\n')
   printf('24-Mostrar la cantidad de números primos en el segundo vector\n')
   printf('25-Mostrar la cantidad de números primos en el tercer vector\n')
   printf('0-TERMINAR\n\n')

   op=input('Ingrese su opción:')
   select op
   case 1 then muestra_vector(a)
   case 2 then muestra_vector(b)
   case 3 then muestra_vector(c)
   case 4 then printf('\n La cantidad de números pares del primer vector es %i
\n', pares(a))
   case 5 then printf('\n La cantidad de números pares del primer vector es %i
\n', pares(b))
   case 6 then printf('\n La cantidad de números pares del primer vector es %i
\n', pares(c))
   case 7 then printf('\n El promedio de los valores del primer vector es %f
\n',promedio(a))
   case 8 then printf('\n El promedio de los valores del primer vector es %f
\n',promedio(b))
   case 9 then printf('\n El promedio de los valores del primer vector es %f
\n',promedio(c))
   case 10 then printf ('La suma es %i \n',suma_vectores(a,b))
   case 11 then printf ('La suma es %i \n',suma_vectores(a,c))
   case 12 then printf ('La suma es %i \n',suma_vectores(b,c))
   case 13 then bus=input(' \n Ingrese el valor que desea buscar:')
       printf('\n El %i esta %i veces \n',bus,busqueda(a))
   case 14 then bus=input(' \n Ingrese el valor que desea buscar:')
       printf('\n El %i esta %i  veces \n',bus,busqueda(b))
   case 15 then bus=input(' \n Ingrese el valor que desea buscar:')
       printf('\n El %i esta %i veces \n',bus,busqueda(c))
 case 16 then printf('\n La cantidad de valores que son divisibles por 3 es %i
\n',divisores(a))
 case 17 then printf('\n La cantidad de valores que son divisibles por 3 es %i
\n',divisores(b))
 case 18 then printf('\n La cantidad de valores que son divisibles por 3 es %i
\n',divisores(c))
 case 19 then printf('\n La cantidad de números negativos es %i\n',
negativo(a,b,c))
 case 20 then printf('\n Los divisores del primer vector son:')
     div_elementos(a)
case 21 then printf('\n Los divisores del segundo vector son:')
    div_elementos(b)
case 22 then printf('\n Los divisores del tercer vector son:')
   div_elementos(c)
```

```
case 23 then printf('\n La cantidad de primos del primer vector es %i
\n',can_pri(a))
case 24 then printf('\n La cantidad de primos del segundo vector es %i
\n',can_pri(b))
case 25 then printf('\n La cantidad de primos del tercer vector es %i
\n',can_pri(c))
case 0 then printf('\n Eligió terminar \n')
else printf('\n Opción no válida \n')
   end
end
```

Anexo 4: Tareas evaluadas desarrolladas por estudiantes

Anexo 4.1: Tarea evaluada 1

Estudiante 1

```
-->x=1
 x  =
    1.
-->y=2
 y  =
    2.
-->b=x+y
 b  =
    3.
-->c=x*y
 c  =
    2.
-->d=x/y
 d  =
    0.5
-->e=b^2+2*(c+d)^3
 e  =
    40.25
-->a=x^3*(2*x^2+3.56*x-
5.21)/((sqrt(3*x+2.3)))
 a  =
    0.1520303
-->%pi*5.2^2
 ans  =
    84.948665
-->2*%pi*5.2
 ans  =
    32.672564
-->%pi*9^2
 ans  =
    254.469
-->2*%pi*9
 ans  =
    56.548668
```

Estudiante 2

```
-->x=12;
-->y=3;
-->b=x+y
 b  =
    15.
-->c=x*y
 c  =
    36.
```

```
-->d=x/y
 d  =
    4.
-->e=(b^2)+2*((c+d)^3)
 e  =
    128225.
-->a=(x^3)*((2*(x^2))+(3.56*x)-
5.21)/(sqrt((3*x)+2.3))
 a  =
    90888.448
-->r=5.2;
-->%pi*(r^2)
 ans  =
    84.948665
-->p=2*%pi*r
 p  =
    32.672564
-->p=d*%pi
 p  =
    56.548668
-->a=%pi*(d/2)^2
 a  =
    254.469
```

Anexo 4.2: Tarea evaluada 2

Ejercicio 1

```
//ejercicio 1
-->a=6.3
a  =
    6.3
-->c=10.1
 c  =
    10.1
-->b=sqrt(c^2-a^2)
b  =
    7.8943017
-->anguloalfa=asin(a/c)
 alfa=                                  0.6735471
-->alfaengrados=(alfa*180)/3.14
alfaengrados=
38.610981
```

Ejercicio 2.

```
//ejercicio 2
-->a=10.5
 a  =
         10.5
-->b=7.3
 b  =
    7.3
-->c=sqrt(b^2+a^2)
 c  =
     12.788276
-->anguloalfa=acos(b/c)
 anguloalfa  =
    0.9632734
 -->alfaengrados=(alfa*180)/3.14   alfaengrados
=
55.219496
```

Ejercicio 3

```
//ejercicio 3
-->x=3
 x  =
     3.
 -->y=abs(sqrt((2*x)-(x+2)))+sqrt(x+3)
//función creada
 y  =
     3.4494897
```

```
 -->x=10
 x  =
     10.
 -->y=abs(sqrt((2*x)-(x+2)))+sqrt(x+3)
 y  =
     6.4339784
 -->x=20
 x  =
     20.
 -->y=abs(sqrt((2*x)-(x+2)))+sqrt(x+3)
 y  =
     9.0384722
 -->x=5
 x  =
     5.
 -->y=abs(sqrt((2*x)-(x+2)))+sqrt(x+3)
 y  =
     4.5604779
```

Ejercicio 4

```
//ejercicio 4
-->x=1
 x  =
     1.
 -->y=abs(sqrt(x^2-abs(x+2)))
y  =
  1.4142136
 -->x=2
 x  =
    2.
 -->y=abs(sqrt(x^2-abs(x+2)))
 y  =
    0.
 -->x=0
 x  =
    0.
 -->y=abs(sqrt(x^2-abs(x+2)))
 y  =
    1.4142136
 -->x=-1
 x  =
  - 1.
 -->y=abs(sqrt(x^2-abs(x+2)))
 y  =
  0.
 -->x=-2
 x  =
  - 2.
 -->y=abs(sqrt(x^2-abs(x+2)))
 y  =
    2.
```

Ejercicio 5

```
//ejercicio 5
-->a=30
 a  =
    30.
-->angrad=(30*3.14)/180
 angrad  =
    0.5233333
-->sin(angrad)
 ans  =
    0.4997701
-->cos(angrad)
 ans  =
    0.8661581
 -->acos(angrad)
 ans  =
    1.0200383
 -->asin(angrad)
 ans  =
    0.5507580
 -->tan(angrad)
 ans  =
    0.5769964
 -->atan(angrad)
 ans  =
    0.4821396
-->cotg(angrad)
 ans  =
    1.7331131
```

Anexo 4.3: Tarea evaluada 3

Estudiante 1

```
-->//Ejercicio 1
--> ejercicio (a)
-->z1=5.3+10*%i
 z1  =
    5.3 + 10.i
-->//Parte Real
-->a=real(z1)
 a  =
    5.3
-->//Parte Imaginaria
-->b=imag(z1)
 b  =
    10.
-->//Conjugado
-->c=conj(z1)
 c  =
    5.3 - 10.i
-->d=atan(imag(z1)/real(z1))
 d  =
    1.0834377
-->//Magnitud
-->e=abs(z1)
 e  =
    11.317685
-->//Ejercicio (b)
-->z2=4.3+5*%i
 z2  =
    4.3 + 5.i
-->//Ejercicio (c)
-->//Parte Real
-->f=real(z2)
 f  =
    4.3
-->//parte imaginaria
-->g=imag(z2)
 g  =
    5.
-->//conjugado
-->h=conj(z2)
 h  =
    4.3 - 5.i
-->//argumento
-->j=atan(g/f)
 j  =
    0.8605253
-->//magnitud
-->k=abs(z2)
 k  =
    6.5946948
```

```
-->//ejercicio (d)
-->a1=z1+z2
 a1  =
    9.6 + 15.i
-->a2=z1-z2
 a2  =
    1. + 5.i
-->a3=z2-z1
 a3  =
  - 1. - 5.i
-->a4=z1*z2
 a4  =
  - 27.21 + 69.5i
 -->a5=z1/z2
 a5  =
     1.6737181 + 0.3793976i
 -->a6=z2/z1
 a6  =
     0.5682723 - 0.1288157i
 -->a7=((a4-z1)/z2)+(2.2+8*%i)
 a7  =
     5.8262819 + 17.620602i

-->//Ejercicio (2)
->p1=poly([2,5,3],"x","c")
 p1  =
                2
    2 + 5x + 3x
-->roots(p1)
 ans  =
  - 1.
  - 0.6666667
-->q1=poly([3,4,6],"x","c")
 q1  =
                2
    3 + 4x + 6x
-->roots(q1)
 ans  =
  - 0.3333333 + 0.6236096i
  - 0.3333333 - 0.6236096i
-->p2=poly([1,2],"x")
 p2  =
              2
    2 - 3x + x
-->roots(p2)
 ans  =
    2.
    1.
-->q2=poly([3,6],"x")
 q2  =
                2
    18 - 9x + x
-->roots(q2)
 ans  =
```

```
    6.
    3.
-->b1=p1+q2
 b1  =
                2
    20 - 4x + 4x
-->b2=p1-p2
 b2  =
           2
    8x + 2x
-->b3=q1-p1
 b3  =
               2
    1 - x + 3x
 -->b4=q1*p2
 b4  =
               2     3    4
    6 - x + 3x - 14x + 6x
 -->b5=p1/p2
 b5  =
                2
    2 + 5x + 3x
    -----------
              2
    2 - 3x + x
 -->b6=q2/p1
 b6  =
                2
    18 - 9x + x
    -----------
               2
    2 + 5x + 3x
 -->b7=b1+p2+q1
 b7  =
                  2
    25 - 3x + 11x
```

Estudiante 2

```
-->//EJERCICIO N°1 LETRA A
-->Z1=5.3+10*%i
 Z1  =
    5.3 + 10.i
-->REAL=real(Z1)
 REAL  =
    5.3
-->IMAGINARIO=imag(Z1)
 IMAGINARIO  =
    10.
-->CONJUGADO=conj(Z1)
 CONJUGADO  =
    5.3 - 10.i
-->ARGUMENTO=IMAGINARIO/REAL
```

```
 ARGUMENTO  =
    1.8867925
-->MAGNITUD=abs(Z1)
 MAGNITUD  =
    11.317685

-->//EJERCICIO 1 LETRA B
 -->Z2=1+2*%i
 Z2  =
    1. + 2.i

->//EJERCICIO 1 LETRA c
-->R=real(Z2)
 R  =
    1.
-->I=imag(Z2)
 I  =
    2.
-->C=conj(Z2)
 C  =
    1. - 2.i
-->A=atan(I/R)
 A  =
    1.1071487
-->M=abs(Z2)
M
=
    2.236068

-->//EJERCICIO 1 LETRA D
 -->Z1+Z2
 ans  =
     6.3 + 12.i
 -->Z1-Z2
 ans  =
     4.3 + 8.i
 -->Z2-Z1
 ans  =
   - 4.3 - 8.i
 -->Z1*Z2
 ans  =
   - 14.7 + 20.6i
 -->Z1/Z2
 ans  =
     5.06 - 0.12i
 -->Z2/Z1
 ans  =
     0.1975174 + 0.0046842i
 -->((Z1*Z2)-Z1)/Z2 + (2.2+8*%i)
 ans  =
     2.44 + 18.12i

-->//EJERCICIO N°2
 -->P1=poly([1,3],"X","C")
```

```
P1  =
    1 + 3X
-->roots(P1)
ans  =
  - 0.3333333
-->Q1=poly([2,1],"X","C")
Q1  =
    2 + X
-->roots(Q1)
ans  =
  - 2.
-->P2=poly([0,1],"X")
P2  =
         2
 - X + X
-->Q2=poly([1,4],"X")
Q2  =
              2
   4 - 5X + X
-->P1+Q2
ans  =
              2
   5 - 2X + X
-->P1-P2
ans  =
              2
   1 + 4X - X
-->Q1-P1
ans  =
    1 - 2X
-->Q1*P2
ans  =
          2   3
 - 2X + X + X
-->P1/P2
ans  =
    1 + 3X
   ------
        2
 - X + X
-->Q2/P1
ans  =
              2
   4 - 5X + X
   ----------
     1 + 3X
-->P1+P2+Q1+Q2
ans  =
               2
   7 - 2X + 2X
```

Anexo 4.4: Tarea evaluada 4

Estudiante 1

Ejercicio 1

```
a=input('ingrese edad:');
if a<0 then
    printf('\nNO ES POSIBLE');
else
    if a<10 | a==10 then
     printf('\nNIÑO');
    else
      if a<14 | a==14 then
        printf('\nPUBER');
      else
        if a<18 | a==18 then
            printf('\nADOLESCENTE');
        else
            if a<25 | a==25 then
                printf('\nJOVEN');
            else
               if a<65 | a==65;
                printf('\nADULTO');
               else

printf('\nANCIANO');
                end
            end
        end
    end
end
end
```

Ejercicio 2

```
a=input('ingrese un numero A:');
b=input('ingrese un numero B:');
c=input('ingrese un numero C:');
if (a<0 | a==0) | (b<0 | b==0) | (c<0 | c==0)then
    printf('\nERROR');
else
    if (a^2+b^2==c^2) | (a^2+c^2==b^2) |
(b^2+c^2==a^2)then
        printf('\nES UN TRIO PITAGORICO');
    else
        printf('\nNO ES TRIO PITAGORICO');
    end
end
```

Ejercicio 3

```
a=input('ingrese sueldo:');
b=input('ingrese ventas:');
if (a<0 | a==0) | (b<0 | b==0) then
    printf('\nERROR');
else
    if b<4000000 then
        printf('\n %f',a);
    else
        if b<10000000 then
            a=a*1.03;
            printf('\n %f',a);
        else
            a=a*1.07;
            printf('\n %f',a);
        end
    end
end
```

Estudiante 2

Ejercicio 1

```
e=input('ingrese edad');
    if (e<10) | (e==10) then
        printf('niño');
    else
        if (e<14) | (e==14) then
            printf('puber');
        else
            if (e<18) | (e==18) then
                printf('adolecente');
            else
                if (e<25) | (e==25) then
                    printf('joven');
                else
                    printf('anciano');
                end
            end
        end
    end
```

Ejercicio 2

```
a=input('ingrese valor');
b=input('ingrese valor');
c=input('ingrese valor');
if (a^2+b^2==c^2) | (a^2+c^2==b^2) | (b^2+c^2==a^2)
then
    printf('forma terna pitagórica');
```

```
else
    printf('no forma terna pitagórica');
end
```

Ejercicio 3

```
sb=input('ingrese sueldo base');
v=input('ingrese ventas');
if sb>0 & (v>0 | v==0) then
    if v<4000000 then
        s=sb;
    else
        if v<10000000 then
            s=sb+v*0.03;
        else
            s=sb+v*0.07;
        end
    end
    printf('\n el sueldo mensual es %f',s);
else
    printf('error');
    end
```

Anexo 4.5: Tarea evaluada 5

Ejercicio 1

Estudiante 1

```
n=input('ingrese su nota');
s=n;
M=n;
m=n;
c=1;
while (c<5)
    n=input('ingrese su nota');
    s=s+n;
    if (n>M) then
        M=n;
    else
        if (n<m) then
            m=n;
        end
    end
    c=c+1;
end
p=s/5;
printf('\nSu promedio es: %f\n Su nota mas baja es: %f\n Su nota mas alta es:
%f',p,m,M);
```

Estudiante 2

```
n=input('ingrese una nota del alumno que este comprendida entre el uno y el
siete :');
a=n; b=n; s=n; c=2;
while c<=5
n1=input('ingrese una nota del alumno que este comprendida entre el uno y el
siete:');
s=s+n1;
if n1>a then
a=n1;
else
if n1<b then
b=n1;
end
end
c=c+1;
end
printf('el promedio del alumno es: %f, la nota más alta es: %f, la nota más
baja es: %f', (s/5),a,b);
```

Ejercicio 2

Estudiante 1

```
a=input('ingresar poblacion mayor de los paises (A):');
b=input('ingresar poblacion menor de los paises (B):');
if a>b & b>0 then
    c=0;
    while a>b
        a=(a*1.03);
        b=(b*1.06);
        c=c+1;
        printf('\n\n poblacion del pais A es:%f \n poblacion del pais B
es:%f',a,b);
    end
    printf('\n\n los años que tarda la poblacion (B) en superar a la poblacion
(A)es:%d',c);
else
    printf('\n error');
end
```

Estudiante 2

```
a=input('ingrese la poblacion menor obtenida en el primer año: ');
b=input('ingrese la poblacion mayor obtenida en el primer año: ');
c=1;
while (a<b)
    a=a*1.06;
    b=b*1.03;
    c=c+1;
printf('la poblacion en el año %d es %f,%f\n',c,a,b);
end
```

Ejercicio 3

Estudiante 1

```
d=0; b=0; s=0; i=0; c=1;
while c<=10
p=input('ingrese un valor númerico comprendido entre uno y siete para el
promedio del alumno:');
if p>=6.0 then
d=d+1;
else
if p>=5.0 then
b=b+1;
else
if p>=4.0 then
s=s+1;
else
i=i+1;
```

```
end
end
end
c=c+1;
end
printf('cantidad de destacados: %d, cantidad de bueno: %d, cantidad de
suficiente: %d y cantidad de insuficiente: %d', d,b,s,i);
```

Estudiante 2

```
c=1;
d=1;
b=0;
s=0;
i=0;
while (c<10) | (c==10)
    n=input('Ingrese promedio:');
    if n<40 then
        i=i+1;
    else
        if n<49 then
            s=s+1;
        else
            if n<59 then
                b=b+1;
            else
                d=d+1;
            end
        end
    end
    c=c+1;
end
printf('\n Los alumnos con nota destacada son: %i \n Los alumnos con nota buena
son: %i \n Los alumnos con nota suficiente son: %i \n Los alumnos con nota
insuficiente son: %i',d,b,s,i);
```

Ejercicio 4

Estudiante 1

```
//EJERCICIO N° 4
P=1
C=1
while C<10 | C==10
    P=P*C
    C=C+1
end
printf ('\n %f',P);
```

Estudiante 2

```
c=1;
s=2;
while s<=10
    c=c*s;
    s=s+1;
end
printf('\n La multiplicacion es %d',c);
```

Ejercicio 5

Estudiante 1

```
a=input('ingrese la base');
b=input('ingrese el exponente');
c=2;
x=a;
while (c<=b)
a=a*x;
c=c+1;
end
printf('la potencia es %d',a);
```

Estudiante 2

```
b=input('ingrese una base numerica '); e=input('ingrese un exponente
numerico');
c=1; p=1;
while c<=e
    p=p*b; c=c+1;
end
printf('el valor de la potencia es: %d', p);
```

Ejercicio 6

Estudiante 1

```
n=input('Ingrese número: ');
c=2;
f=1;
while (c<n) | (c==n)
    f=f*c;
    c=c+1;
end
printf('El factorial de %i es:%f',n,f);
```

Estudiante 2

```
n=input('ingrese un número:');
if n>=0 then
    if n==0 then
        printf('el factorial del número es: 1')
    else
        c=1; f=1;
        while c<=n
            f=f*c; c=c+1;
        end
    end
    printf('el factorial del número es: %d', f);
else
    printf('error');
end
```

Ejercicio 7

Estudiante 1

```
x=input('ingrese el valor de x en la serie: ');
n=input('ingrese el valor de n en la serie: ');
s=1;
c=1;
f=1;
while (c<n|c==n)
    b=c+x;
    j=1;
    p=1;
    while (j<c|j==c)
        p=p*b;
        j=j+1;
    end
    f=f*c;
    s=s+p/f;
    c=c+1;
end
printf('el resultado de la serie es: %f',s);
```

Estudiante 2

```
x=input('ingrese un valor'); n=input('ingrese un valor');
s=1; c=1; f=1;
while c<=n
    b=c+x; j=1; p=1;
    while j<=c
        p=p*b; j=j+1;
    end
        f=f*c; s=s+p/f; c=c+1;
    end
printf('el valor de la suma de la serie es: %f', s);
```

Anexo 4.6: Tarea evaluada 6

```
function v=llena_vector(v)
   for i=1:5
   v(i)=input('ingrese un entero')
   end
endfunction
function muestra_vector(v)
    for i=1:5
         printf('el valor en %i es %i \n',i, v(i))
   end
endfunction
function cp=pares(v)
   cp=0
   for i=1:5
     if modulo(v(i),2)==0 then
   cp=cp+1
   end
   end
endfunction
function p=promedio(v)
   s=0
   for i=1:5
   s=s+v(i)
   end
   p=s/5
endfunction
function suma=suma_vectores(v1, v2)
   for i=1:5
   suma(i)=v1(i)+v2(i)
   end
endfunction
function f=busqueda(v, bus)
   f=0
   for i=1:5
   if bus==v(i) then
   f=f+1
   end
   end
endfunction
function c=divisores(v)
   c=0
   for i=1:5
   if modulo(v(i),3)==0 then
   c=c+1
   end
   end
endfunction
function c=negativo(v1, v2, v3)
   c=0
   for i=1:5
       if v1(i)<0 then
           c=c+1
```

```
            if v2(i)<0 then
                c=c+1
                if v3(i)<0 then
                    c=c+1
                end
            end
        end
        end
 endfunction
function div_elementos(v)
   for i=1:5
       for j=1:v(i)
           if modulo (v(i),j)==0 then
               printf('el %i es divisible por %i\n',v(i),j)
           end
       end
   end
   endfunction
function cp=can_pri(v)
   cp=0
   for i=1:5
       cd=0
       for j=1:5
           if modulo(v(i),j)==0 then
               cd=cd+1
               end
           end
           if cd==2 then
               cp=cp+1
           end
       end
endfunction

a=[ ]
b=[ ]
c=[ ]
a= llena_vector(a)
b=llena_vector(b)
c=llena_vector(c)
op=1
while op~=0
   printf('1.........mostrar el primer vector \n')
   printf('2.........mostrar el segundo vector\n')
   printf('3.........mostrar el tercer vector \n')
   printf('4.........cantidad de elementos pares del primer vector\n')
   printf('5.........cantidad de elementos pares del segundo vector\n')
   printf('6.........cantidad de elementos pares del tercer vector\n')
   printf('7.........calcular y mostrar el promedio de los elementos del
primer vector\n')
   printf('8.........calcular y mostrar el promedio de los elementos del
segundo vector\n')
   printf('9.........calcular y mostrar el promedio de los elementos del
tercer vector\n')
   printf('10.........sumar vectores: primer vector y segundo vector\n')
```

```
   printf('11.........sumar vectores: primer vector y tercer vector\n')
   printf('12.........sumar vectores: segundo vector y tercer vector\n')
   printf('13.........busca un número determinado y devuelve la cantidad de
veces que se encuentra en el primer vector\n')
   printf('14.........busca un número determinado y devuelve la cantidad de
veces que se encuentra en el segundo vector\n')
   printf('15.........busca un número determinado y devuelve la cantidad de
veces que se encuentra en el tercer vector\n')
   printf('16.........mostrar la cantidad de valores que son divisibles por 3
en el primer vector \n')
   printf('17.........mostrar la cantidad de valores que son divisibles por 3
en el segundo vector \n')
   printf('18.........mostrar la cantidad de valores que son divisibles por 3
en el tercer vector \n')
   printf('19.........mostrar el total de valores negativos que existen entre
los tres vectores \n')
   printf('20.........buscar y muestrar los divisores de cada elemento del
primer vector\n')
   printf('21.........buscar y muestrar los divisores de cada elemento del
segundo vector\n')
   printf('22.........buscar y muestrar los divisores de cada elemento del
tercer vector\n')
   printf('23.........muestre  la cantidad de primos en el primer vector\n')
   printf('24.........muestre  la cantidad de primos en el segundo vector\n')
   printf('25.........muestre  la cantidad de primos en el tercer vector\n')
   printf('0..........para TERMINAR\n\n')
      op=input('ingrese su opción')
   select op
   case 1 then muestra_vector(a)
   case 2 then muestra_vector(b)
   case 3 then muestra_vector(c)
   case 4 then printf('la cantidad de números pares del primer vector es %i\n',
pares(a))
   case 5 then printf('la cantidad de números pares del primer vector es %i\n',
pares(b))
   case 6 then printf('la cantidad de números pares del primer vector es %i\n',
pares(c))
   case 7 then printf('el promedio de los valores del primer vector es
%f\n',promedio(a))
   case 8 then printf('el promedio de los valores del primer vector es
%f\n',promedio(b))
   case 9 then printf('el promedio de los valores del primer vector es
%f\n',promedio(c))
   case 10 then printf ('la suma es %i\n',suma_vectores(a,b))
   case 11 then printf ('la suma es %i\n',suma_vectores(a,c))
   case 12 then printf ('la suma es %i\n',suma_vectores(b,c))
   case 13 then bus=input('ingrese el valor que desea buscar')
       printf('el %i esta %i \n',bus,busqueda(a))
   case 14 then bus=input('ingrese el valor que desea buscar')
       printf('el %i esta %i \n',bus,busqueda(b))
   case 15 then bus=input('ingrese el valor que desea buscar')
       printf('el %i esta %i \n',bus,busqueda(c))
 case 16 then printf('la cantidad de valores que son divisibles por 3 es %i
\n',divisores(a))
```

```
 case 17 then printf('la cantidad de valores que son divisibles por 3 es %i
\n',divisores(b))
 case 18 then printf('la cantidad de valores que son divisibles por 3 es %i
\n',divisores(c))
 case 19 then printf('la cantidad de numeros negativos son %i\n',
negativo(a,b,c))
 case 20 then printf('los divisores del primer vector son')
     div_elementos(a)
case 21 then printf('los divisores del segundo vector son')
    div_elementos(b)
case 22 then printf('los divisores del tercer vector son')
   div_elementos(c)
case 23 then printf('la cantidad de primos del primer vector es
%i\n',can_pri(a))
case 24 then printf('la cantidad de primos del primer vector es
%i\n',can_pri(b))
case 25 then printf('la cantidad de primos del primer vector es
%i\n',can_pri(c))
case 0 then printf('eligio terminar')
else printf('opción no valida\n')
   end
end
```

Anexo 5: Resumen de ayudantías

Operaciones en Scilab

Operación	Símbolo
Suma	+
Resta	-
Multiplicación	*
División	/
Potencia	^
Raíz Cuadrada	sqrt()
Número Complejo	Parte entera + parte imaginaria * %i

Ejemplo

Operación	En Scilab
$\frac{a^2}{bc}+\sqrt{d}$	- -> (a^2/b*c)+sqrt(d)

FUNCIONES PREDEFINIDAS EN SCILAB

abs : valor absoluto ceil : parte entera superior fix : redondeo hacia cero (igual a int) int : redondeo hacia cero (igual a fix) floor : parte entera inferior max : máximo min : mínimo modulo : residuo entero rand : número aleatorio round : redondeo	sqrt : raíz cuadrada exp : función exponencial: e^x log : logaritmo natural log10 : logaritmo decimal log2 : logaritmo en base dos sin : seno cos : coseno acos : arcocoseno asin : arcoseno tan : tangente atan : arcotangente cotg : cotangente

Ejemplos

Operación	En Scilab
$\lvert -b \rvert + seno(30)$	- -> abs(-b)+sin(30)
Parte entera inferior	-->a=3.25 a = 3.25 -->floor(a) ans = 3.

Parte entera superior	-->ceil(a) ans = 4.
Redondeo hacia cero	-->int(a) ans = 3.
Redondeo	-->round(a) ans = 3.

NUMEROS COMPLEJOS

Numero complejo	En Scilab
Z=a+bi	- -> z=a+b*%i

z= a + b i

a: parte real

b: parte imaginaria

parte real	- -> re (z)
parte imaginaria	- -> im(z)
Conjugado	- -> conj(z)
Hallar magnitud	- -> Abs()
Hallar argumento	- -> arctan(im(z)/re(z))

POLINOMIOS

Pueden definirse de dos formas diferentes.

<u>Especificando raíces:</u>

```
Ingreso en Scilab
 -->p=poly([1,2],'x')
      p =
                            2
         2 - 3x + x
O también
-->p=poly([1,2],"x","roots")
 p  =
                     2
    2 - 3x + x
 Otra forma
-->p=poly([1,2],'x','r')
 p  =
                    2
    2 - 3x + x
```

Especificando los coeficientes:

```
Ingreso en Scilab
       -->q=poly([1,2],'x','c')
            q =
                 1 + 2x
O también
-->p=poly([1,2],"x","coeff")
 p  =
     1 + 2x
```

Obtener raíces:

```
--> roots(q)
```

VECTORES Y MATRICES

Notación	Mostrar Fila
Ve=[1 2 3] Ma=[1 2 3;4 5 6]	Ma(2,:)
Mostrar Elemento	Mostrar Columna
Ma(1,2)	Ma(:,2)

Implementación de algoritmos selectivos en SciNotes

si () entonces	if … then
Sino	else
fin si	end
Mientras	While
leer a	a=input (‘Ingrese dato’)
escribir	printf(‘texto’)
o	|
=	==
<-	=
y	&
salto de línea	\n
a MOD b	modulo(a,b)
%f	Real
%d	Entero
menor o igual	<=
mayor o igual	>=
Distinto	~=